IBM WebSphere MQ

Managed File Transfer

Version 7 Release 5

IBM

Contents

Figures

Tables

Product overview

This section provides introductory information that you can use to get started with WebSphere® MQ Managed File Transfer.

- "WebSphere MQ Managed File Transfer introduction"
- "WebSphere MQ Managed File Transfer product options" on page 4
- "WebSphere MQ Managed File Transfer topology overview" on page 10
- Basic file transfer using the scripts
- Two machine file transfer scenario using scripts
- Basic file transfer scenario
- Two machine file transfer scenario
- "What's new in Version 7.5?" on page 11

WebSphere MQ Managed File Transfer introduction

WebSphere MQ Managed File Transfer transfers files between systems in a managed and auditable way, regardless of file size or the operating systems used.

You can use WebSphere MQ Managed File Transfer to build a customized, scalable, and automated solution that enables you to manage, trust, and secure file transfers. WebSphere MQ Managed File Transfer eliminates costly redundancies, lowers maintenance costs, and maximizes your existing IT investments.

DEVELOPING APPLICATIONS
- RESTFUL API
- ANT SCRIPTING

ADMINISTERING
- WEBSPHERE MQ EXPLORER
- COMMAND-LINE TOOLS

DATABASE LOGGER
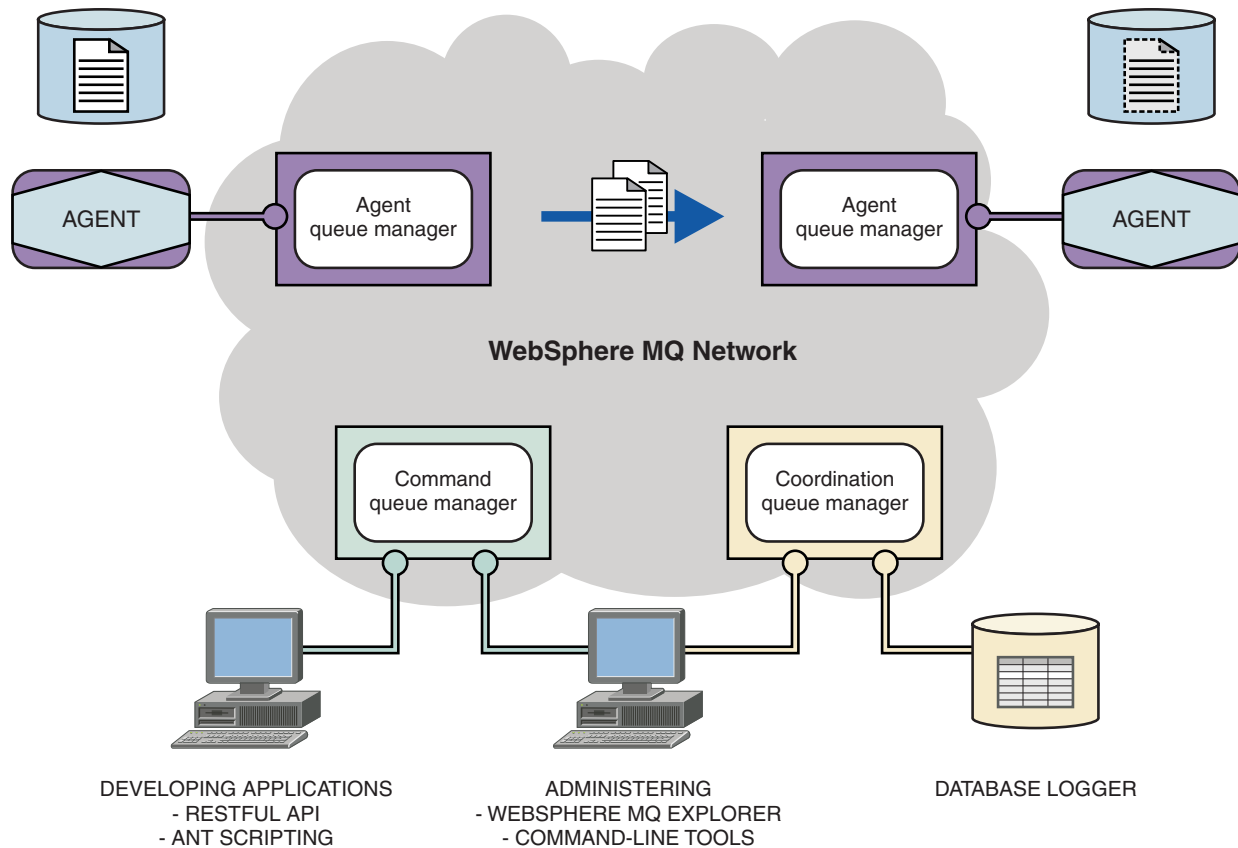
The diagram shows a simple WebSphere MQ Managed File Transfer topology. There are two agents, each connect to their own agent queue manager in a WebSphere MQ network. A file is transferred from the agent on the one side of the diagram, through the WebSphere MQ network, to the agent on the other side of the diagram. Also in the WebSphere MQ network are the coordination queue manager and a command queue manager. Applications and tools connect to these queue managers to configure, administer, operate, and log WebSphere MQ Managed File Transfer activity in the WebSphere MQ network.

WebSphere MQ Managed File Transfer can be installed as four different options, depending on your operating system and overall setup. These options are WebSphere MQ Managed File Transfer Agent, WebSphere MQ Managed File Transfer Logger, WebSphere MQ Managed File Transfer Service, or WebSphere MQ Managed File Transfer Tools For information, see "WebSphere MQ Managed File Transfer product options" on page 4.

You can use WebSphere MQ Managed File Transfer to perform the following tasks:
- Create managed file transfers
  - Create new file transfers from WebSphere MQ Explorer on Linux or Windows platforms.
  - Create new file transfers from the command line on all supported platforms.
  - Integrate file transfer function into the Apache Ant tool.
  - Write applications that control WebSphere MQ Managed File Transfer by putting messages on agent command queues.
  - Schedule file transfers to take place at a later time. You can also trigger scheduled file transfers based on a range of file system events, for example a new file being created.

- Continually monitor a resource, for example a directory, and when the contents of that resource meet some predefined condition, start a task. This task can be a file transfer, an Ant script, or a JCL job.
- Use the RESTful API provided by the WebSphere MQ Managed File Transfer Web Gateway to transfer files.
- Transfer files to and from IBM WebSphere MQ queues.
- Transfer files to and from FTP, FTPS, or SFTP servers.
- Transfer files to and from Connect:Direct® nodes.
- Transfer both text and binary files. Text files are automatically converted between the code pages and end-of-line conventions of the source and destination systems.
- Transfers can be secured, using the industry standards for Secure Socket Layer (SSL) based connections.
- View transfers in progress and log information about all transfers in your network
  - View the status of transfers in progress from WebSphere MQ Explorer on Linux or Windows platforms.
  - Check the status of completed transfers by using the WebSphere MQ Explorer on Linux or Windows platforms.
  - Use the WebSphere MQ Managed File Transfer database logger feature to save log messages to a DB2® or Oracle database.
  - Use the RESTful API provided by the WebSphere MQ Managed File Transfer Web Gateway to see information about all transfers in your network.

WebSphere MQ Managed File Transfer is built on IBM WebSphere MQ, which provides assured, once-only delivery of messages between applications. You can take advantage of various features of IBM WebSphere MQ. For example, you can use channel compression to compress the data that you send between agents over IBM WebSphere MQ channels and use SSL channels to secure the data that you send between agents. Files are transferred reliably and can tolerate the failure of the infrastructure over which the file transfer is carried out. If you experience a network outage, the file transfer restarts from where it left off when connectivity is restored.

By consolidating file transfer with your existing IBM WebSphere MQ network, you can avoid spending the resources required to maintain two separate infrastructures. If you are not already a IBM WebSphere MQ customer, by creating a IBM WebSphere MQ network to support WebSphere MQ Managed File Transfer you are building the backbone for a future SOA implementation. If you are already a IBM WebSphere MQ customer, WebSphere MQ Managed File Transfer can take advantage of your existing IBM WebSphere MQ infrastructure including IBM WebSphere MQ internet pass-thru and IBM® Integration Bus.

WebSphere MQ Managed File Transfer integrates with a number of other IBM products:

IBM Integration Bus

Process files that have been transferred by WebSphere MQ Managed File Transfer as part of an IBM Integration Bus flow. For more information, see "Working with IBM Integration Bus" on page 276.

IBM Sterling Connect:Direct

Transfer files to and from an existing Connect:Direct network by using the WebSphere MQ Managed File Transfer Connect:Direct bridge. For more information, see "The Connect:Direct bridge" on page 263.

IBM Tivoli® Composite Application Manager

IBM Tivoli Composite Application Manager provides an agent that you can use to monitor information that is published to the coordination queue manager.

Related concepts:

"WebSphere MQ Managed File Transfer product options"
WebSphere MQ Managed File Transfer can be installed as four different options, depending on your operating system and overall setup. These options are WebSphere MQ Managed File Transfer Agent, WebSphere MQ Managed File Transfer Logger, WebSphere MQ Managed File Transfer Service, or WebSphere MQ Managed File Transfer Tools.

"WebSphere MQ Managed File Transfer topology overview" on page 10

Related reference:

"How does WebSphere MQ Managed File Transfer work?" on page 399
WebSphere MQ Managed File Transfer interacts in a number of ways with WebSphere MQ. This topic describes how the two products interact.

WebSphere MQ Managed File Transfer product options

WebSphere MQ Managed File Transfer can be installed as four different options, depending on your operating system and overall setup. These options are WebSphere MQ Managed File Transfer Agent, WebSphere MQ Managed File Transfer Logger, WebSphere MQ Managed File Transfer Service, or WebSphere MQ Managed File Transfer Tools.

WebSphere MQ Managed File Transfer Agent
> The WebSphere MQ Managed File Transfer Agent install option installs a file transfer agent. A file transfer agent connects to a WebSphere MQ queue manager and transfers file data, as messages, to other file transfer agents. These must be installed either as part of the WebSphere MQ Managed File Transfer Agent or WebSphere MQ Managed File Transfer Service install options.
>
> The WebSphere MQ Managed File Transfer Agent install option can be installed on systems without the WebSphere MQ Server install option being present on the system. Some capabilities of the file transfer agent, installed as part of the WebSphere MQ Managed File Transfer Agent install are only available when the WebSphere MQ Managed File Transfer Agent install is installed on a system where the WebSphere MQ Server install option is installed. For example, the capability to perform protocol bridge configurations and operations.

WebSphere MQ Managed File Transfer Logger
> The WebSphere MQ Managed File Transfer Logger install option installs a file transfer logger. The file transfer logger connects to a WebSphere MQ queue manager, often the queue manager designated as the coordination queue manager, and logs file transfer audit related data to either a database or a file.
>
> The WebSphere MQ Managed File Transfer Logger install option must be installed on systems where the WebSphere MQ Server install option is already installed.

WebSphere MQ Managed File Transfer Service
> The WebSphere MQ Managed File Transfer Service install option installs a file transfer agent that has additional capabilities beyond those provided by the file transfer agent installed via the WebSphere MQ Managed File Transfer Agent install option. These additional capabilities are:
> - Create protocol bridge agents which are used to send and receive files with legacy FTP, FTPS or SFTP servers
> - Deploy the Web Gateway feature which provides RESTful interfaces for building web applications that transfer files
>
> The WebSphere MQ Managed File Transfer Service install option must be installed on systems where the WebSphere MQ Server install option is already installed.

WebSphere MQ Managed File Transfer Tools
> The WebSphere MQ Managed File Transfer Tools install option installs command line tools that are used to interact with file transfer agents. The tools allow you to start file transfers, schedule file transfers and create resource monitors from the command line.

The WebSphere MQ Managed File Transfer Tools install option can be installed and used on either a system where file transfer agents are installed or on a system where no file transfer agents are installed.

On UNIX platforms there is an additional WebSphere MQ Managed File Transfer Base install component. This component contains files common to all of the installation options. You must install the WebSphere MQ Managed File Transfer Base component before installing any of the Agent, Logger, Service, or Tools components.

For more information on the IBM WebSphere MQ components required for each product option on UNIX platforms, see the following topics:
- "Components required for each WebSphere MQ Managed File Transfer product option on HP-UX Systems"
- "Components required for each WebSphere MQ Managed File Transfer product option on Linux systems" on page 7
- "Components required for each WebSphere MQ Managed File Transfer product option on Solaris systems" on page 8
- "Components required for each WebSphere MQ Managed File Transfer product option on AIX systems" on page 9

Capabilities provided by the Service and Agent options

WebSphere MQ Managed File Transfer Service
- Make client or bindings mode connections to queue managers. When the file transfer agent and the queue manager are located on the same system, we recommend that you use the bindings mode connections.
- Transfer files to and from other WebSphere MQ Managed File Transfer agents.
- Transfer files to and from SFTP, FTP, or FTPS protocol servers.
- Transfer files to and from Connect:Direct nodes.
- Transfer files from HTTP clients through the Web Gateway.

Some capabilities are available on only a subset of supported platforms. For more information, see WebSphere MQ System Requirements.

WebSphere MQ Managed File Transfer Agent
- Make client or bindings mode connections to queue managers. When the file transfer agent and queue manager are located on the same system, we recommend that you use the bindings mode connections.
- Transfer files to and from other WebSphere MQ Managed File Transfer agents.
- Transfer files to and from Connect:Direct nodes.

Related concepts:
"WebSphere MQ Managed File Transfer introduction" on page 1
WebSphere MQ Managed File Transfer transfers files between systems in a managed and auditable way, regardless of file size or the operating systems used.
"WebSphere MQ Managed File Transfer topology overview" on page 10

Components required for each WebSphere MQ Managed File Transfer product option on HP-UX Systems

WebSphere MQ Managed File Transfer can be installed as four different options, depending on your operating system and overall setup. On HP-UX systems, these options are WebSphere MQ Managed File Transfer Agent, WebSphere MQ Managed File Transfer Logger, WebSphere MQ Managed File Transfer Service, and WebSphere MQ Managed File Transfer Tools, and each require specific components.

WebSphere MQ Managed File Transfer Agent

MQSERIES.MQM-RUNTIME

MQSERIES.MQM-JAVA

MQSERIES.MQM-JAVAJRE

MQSERIES.MQM-FTBASE

MQSERIES.MQM-FTAGENT

WebSphere MQ Managed File Transfer Logger

MQSERIES.MQM-RUNTIME

MQSERIES.MQM-SERVER

MQSERIES.MQM-JAVA

MQSERIES.MQM-JAVAJRE

MQSERIES.MQM-FTBASE

MQSERIES.MQM-FTLOGGER

WebSphere MQ Managed File Transfer Service

MQSERIES.MQM-RUNTIME

MQSERIES.MQM-SERVER

MQSERIES.MQM-JAVA

MQSERIES.MQM-JAVAJRE

MQSERIES.MQM-FTBASE

MQSERIES.MQM-FTAGENT

MQSERIES.MQM-FTSERVICE

WebSphere MQ Managed File Transfer Tools

MQSERIES.MQM-RUNTIME

MQSERIES.MQM-JAVA

MQSERIES.MQM-JAVAJRE

MQSERIES.MQM-FTBASE

MQSERIES.MQM-FTTOOLS

Components required for each WebSphere MQ Managed File Transfer product option on Linux systems

WebSphere MQ Managed File Transfer can be installed as four different options, depending on your operating system and overall setup. On Linux systems, these options are WebSphere MQ Managed File Transfer Agent, WebSphere MQ Managed File Transfer Logger, WebSphere MQ Managed File Transfer Service, and WebSphere MQ Managed File Transfer Tools, and each require specific components.

WebSphere MQ Managed File Transfer Agent

MQSeriesRuntime

MQSeriesJava

MQSeriesJRE

MQSeriesFTBase

MQSeriesFTAgent

WebSphere MQ Managed File Transfer Logger

MQSeriesRuntime

MQSeriesServer

MQSeriesJava

MQSeriesJRE

MQSeriesFTBase

MQSeriesFTLogger

WebSphere MQ Managed File Transfer Service

MQSeriesRuntime

MQSeriesServer

MQSeriesJava

MQSeriesJRE

MQSeriesFTBase

MQSeriesFTAgent

MQSeriesFTService

WebSphere MQ Managed File Transfer Tools

MQSeriesRuntime

MQSeriesJava

| MQSeriesJRE

| MQSeriesFTBase

| MQSeriesFTTools

Components required for each WebSphere MQ Managed File Transfer product option on Solaris systems

WebSphere MQ Managed File Transfer can be installed as four different options, depending on your operating system and overall setup. On Solaris systems, these options are WebSphere MQ Managed File Transfer Agent, WebSphere MQ Managed File Transfer Logger, WebSphere MQ Managed File Transfer Service, and WebSphere MQ Managed File Transfer Tools, and each require specific components.

WebSphere MQ Managed File Transfer Agent

| runtime

| java

| jre

| ftbase

| ftagent

WebSphere MQ Managed File Transfer Logger

| runtime

| server

| java

| jre

| ftbase

| ftlogger

WebSphere MQ Managed File Transfer Service

| runtime

| server

| java

| jre

| ftbase

| ftagent

| ftservice

WebSphere MQ Managed File Transfer Tools

runtime

java

jre

ftbase

fttools

Components required for each WebSphere MQ Managed File Transfer product option on AIX systems

WebSphere MQ Managed File Transfer can be installed as four different options, depending on your operating system and overall setup. On AIX® systems, these options are WebSphere MQ Managed File Transfer Agent, WebSphere MQ Managed File Transfer Logger, WebSphere MQ Managed File Transfer Service, and WebSphere MQ Managed File Transfer Tools, and each require specific components.

WebSphere MQ Managed File Transfer Agent

mqm.base.runtime

mqm.java.rte

mqm.jre.rte

mqm.ft.base

mqm.ft.agent

WebSphere MQ Managed File Transfer Logger

mqm.base.runtime

mqm.server.rte

mqm.java.rte

mqm.jre.rte

mqm.ft.base

mqm.ft.logger

WebSphere MQ Managed File Transfer Service

mqm.base.runtime

mqm.server.rte

mqm.java.rte

mqm.jre.rte

| mqm.ft.base

| mqm.ft.agent

| mqm.ft.service

| **WebSphere MQ Managed File Transfer Tools**

| mqm.base.runtime

| mqm.java.rte

| mqm.jre.rte

| mqm.ft.base

| mqm.ft.tools

WebSphere MQ Managed File Transfer topology overview

WebSphere MQ Managed File Transfer agents send and receive the files that are transferred. Each agent has its own set of queues on its associated queue manager and the agent is attached to its queue manager in either bindings or client mode. An agent can also use the coordination queue manager as its queue manager.

The coordination queue manager broadcasts audit and file transfer information. The coordination queue manager represents a single point for the collection of agent, transfer status, and transfer audit information. The coordination queue manager is not required to be available in order for transfers to take place. If the coordination queue manager temporarily becomes unavailable, transfers continue as normal. Audit and status messages are stored in the agent queue managers until the coordination queue manager became available, and can then be processed as normal.

Agents register with the coordination queue manager and publish their details to that queue manager. This agent information is used by the WebSphere MQ Managed File Transfer plug-in to enable the start of transfers from the WebSphere MQ Explorer. The agent information collected on the coordination queue manager is also used by the commands to display agent information and agent status.

Transfer status and transfer audit information is published on the coordination queue manager. The transfer status and transfer audit information is used by the WebSphere MQ Managed File Transfer plug-in to monitor the progress of transfers from the WebSphere MQ Explorer. The transfer audit information stored on the coordination queue manager can be retained to provide auditability.

You can use WebSphere MQ Managed File Transfer with either WebSphere MQ Version 6.0 or WebSphere MQ Version 7.0. However, you need access to one WebSphere MQ Version 7.0 or later, queue manager to act as your coordination queue manager.

The command queue manager is used to connect to the WebSphere MQ network and is the queue manager connected to when you issue WebSphere MQ Managed File Transfer commands.
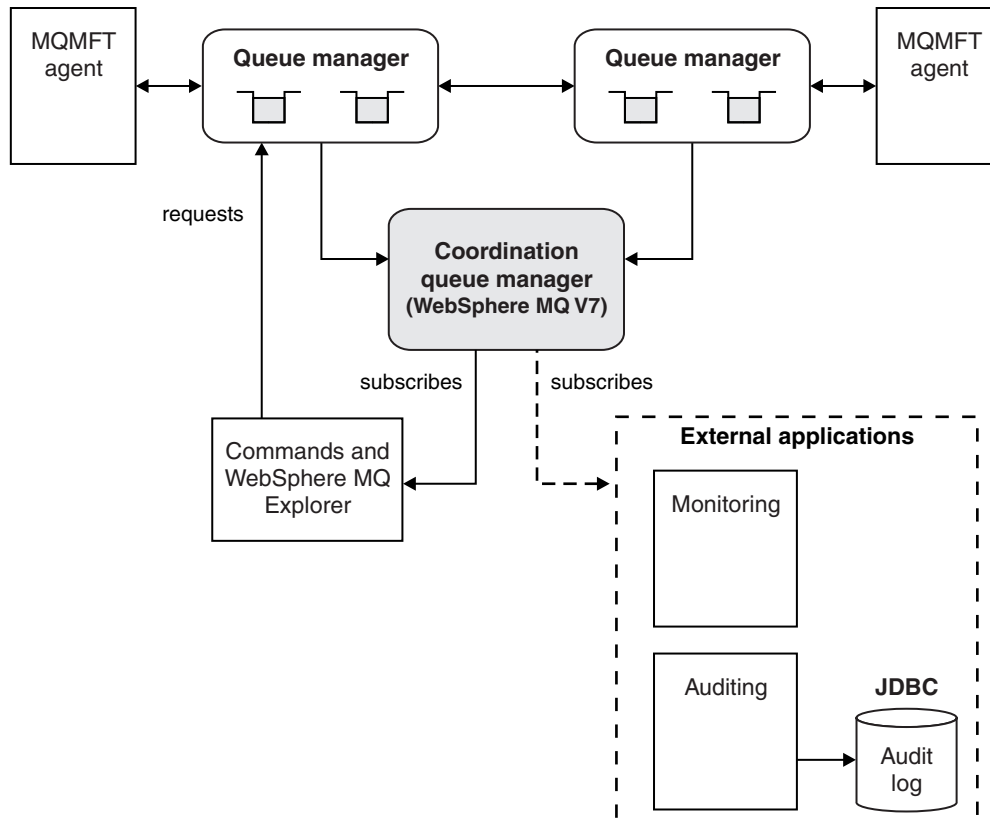
Related concepts:

"WebSphere MQ Managed File Transfer introduction" on page 1
WebSphere MQ Managed File Transfer transfers files between systems in a managed and auditable way, regardless of file size or the operating systems used.

Related reference:

"How does WebSphere MQ Managed File Transfer work?" on page 399
WebSphere MQ Managed File Transfer interacts in a number of ways with WebSphere MQ. This topic describes how the two products interact.

Related information:

Basic file transfer scenario

Two machine file transfer scenario

What's new in Version 7.5?

Learn about the main new functions in WebSphere MQ Managed File Transfer Version 7.5.

Changes to installation

WebSphere MQ Managed File Transfer is now installed as a component of IBM WebSphere MQ. For general information about installing IBM WebSphere MQ, see Installing and uninstalling and for more information about the individual Managed File Transfer components for each platform, see Choosing what to install.

Migration tools

To help you migrate agents, stand-alone database loggers, Web Gateways, and JEE database loggers from WebSphere MQ File Transfer Edition V7.0 to WebSphere MQ V7.5, use the following information:

- "fteMigrateAgent (migrate a WebSphere MQ File Transfer Edition V7.0 agent to WebSphere MQ V7.5" on page 502
- "fteMigrateLogger (migrate a WebSphere MQ File Transfer Edition V7.0 database logger to WebSphere MQ V7.5)" on page 505
- "fteMigrateConfigurationOptions (migrate a WebSphere MQ File Transfer Edition V7.0 configuration to WebSphere MQ V7.5)" on page 504
- "Migrating a WebSphere Application Server V7 Web Gateway from WebSphere MQ File Transfer Edition V7.0 to WebSphere MQ V7.5" on page 24
- "Migrating a Web Gateway on WebSphere Application Server Community Edition from WebSphere MQ File Transfer Edition V7.0 to WebSphere MQ V7.5" on page 25
- "Migrating a WebSphere Application Server V7 JEE database logger from WebSphere MQ File Transfer Edition V7.0 to WebSphere MQ V7.5" on page 26
- "Migrating a JEE database logger on WebSphere Application Server Community Edition from WebSphere MQ File Transfer Edition V7.0 to WebSphere MQ V7.5" on page 27

Changes to the configuration
- Some significant changes have been made to the configuration layout between WebSphere MQ Managed File Transfer and IBM WebSphere MQ, which are summarized in "Changes between WebSphere MQ File Transfer Edition V7.0.4 or earlier and WebSphere MQ V7.5" on page 19
- Several properties, relating to credentials have been moved or replaced. For more information, see "New property locations in MQMFT V7.5."

Enhancements to the logger
- An interface has been added for users to define the content and format of the information output for each log message processed by the file logger. Users can now customize the log files so they can be processed by a log file interpretation script. For more information, see "Installing and configuring the WebSphere MQ Managed File Transfer stand-alone file logger" on page 134.

Enhancements to security support
- FIPS and TLS-enabled channel support has been added to agent, command, and IBM WebSphere MQ Explorer connections to queue managers. For more information, see "FIPS support" on page 691.
- From WebSphere MQ V7.5, to successfully run a subset of the fte commands, you must run these commands as an administrator (member of the mqm group). For more information, see "Security changes" on page 20.

Enhancements to user exit routines
You can manage the properties of multiple protocol file servers by implementing the `com.ibm.wmqfte.exitroutine.api.ProtocolBridgePropertiesExit2` interface, which is an expansion of `com.ibm.wmqfte.exitroutine.api.ProtocolBridgePropertiesExit2` for Version 7.5. For more information, see "Looking up protocol file server properties by using exit classes (ProtocolBridgePropertiesExit2)" on page 253.

New property locations in MQMFT V7.5

Some properties have moved location in WebSphere MQ Managed File Transfer V7.5. These include credential and security properties.

In WebSphere MQ Managed File Transfer V7.5 changes have been made to the way that sensitive information is stored within configuration files. Several properties can no longer be used to specify user names and passwords in the coordination, command, agent and logger properties files. The new configuration model stores the sensitive information that was previously defined by these properties in separate credentials files. New properties have been introduced to allow the location of a credentials file to be specified.

To migrate a configuration from the old model to the new model, you must move sensitive information to a credentials file and you must specify the location of that credentials file using new properties. The following table shows how the old configuration model maps to the new model:

Moved properties

Table 1. Properties moved from property files to credentials file in WebSphere MQ Managed File Transfer V7.5

Old property name	Old file name	Where to specify location of credential file for this property	Where in credentials file to now specify this property
coordinationSslTrustStorePassword	coordination.properties	coordinationSslTrustStoreCredentialsFile in coordination.properties file	password attribute of <file> element in the MQMFTCredentials.xml file
coordinationSslKeyStorePassword	coordination.properties	coordinationSslKeyStoreCredentialsFile in coordination.properties file	password attribute of <file> element in the MQMFTCredentials.xml file
connectionSslTrustStorePassword	command.properties	connectionSslTrustStoreCredentialsFile in command.properties file	password attribute of <file> element in the MQMFTCredentials.xml file
connectionSslKeyStorePassword	command.properties	connectionSslKeyStoreCredentialsFile in command.properties file	password attribute of <file> element in the MQMFTCredentials.xml file
agentSslTrustStorePassword	agent.properties	agentSslTrustStoreCredentialsFile in agent.properties file	password attribute of <file> element in the MQMFTCredentials.xml file
agentSslKeyStorePassword	agent.properties	agentSslKeyStoreCredentialsFile in agent.properties file	password attribute of <file> element in the MQMFTCredentials.xml file
cdNodeTrustStorePassword	agent.properties	cdNodeTrustStoreCredentialsFile in agent.properties file	password attribute of <file> element in the MQMFTCredentials.xml file
cdNodeKeyStorePassword	agent.properties	cdNodeKeyStoreCredentialsFile in agent.properties file	password attribute of <file> element in the MQMFTCredentials.xml file
wmqfte.database.user	logger.properties	wmqfte.database.credentials.file in logger.properties file	user attribute of <logger> element in the MQMFTCredentials.xml file
wmqfte.database.password	logger.properties	wmqfte.database.credentials.file in logger.properties file	password attribute of <logger> element in the MQMFTCredentials.xml file

In addition, several attributes in ProtocolBridgeProperties.xml are no longer valid:

Old XML attribute	New credentials file location element	New XML file password location
trustStorePassword attribute on element <ftpsServer>	<credentialsFile> in ProtocolBridgeProperties.xml	trustStorePassword attribute on element <server> or <serverHost> in the ProtocolBridgeCredentials.xml file
keyStorePassword attribute on element <ftpsServer>	<credentialsFile> in ProtocolBridgeProperties.xml	keyStorePassword attribute on element <server> or <serverHost> in the ProtocolBridgeCredentials.xml file

In WebSphere MQ Managed File Transfer V7.5 to move the old property **coordinationSslTrustStorePassword**, which was stored in the coordination.properties file, complete the following steps:

1. move the value to the password attribute of the <file> element in the MQMFTCredentials.xml file.

2. update the **coordinationSslTrustStoreCredentialsFile** property in the coordination.properties file to specify the location of the credentials file.

What's new in Version 7.5.0.2?

►V 7.5.0.2

Learn about the main new functions in WebSphere MQ Managed File Transfer Version 7.5.0.2.

4690 OS support

You can install WebSphere MQ Managed File Transfer on a device that is running 4690 OS. For more information, see "Using WebSphere MQ Managed File Transfer in a retail environment" on page 35.

Enabling new function for the JEE database logger

A TRANSFER_ITEM_ATTRIBUTES database table is added for the JEE database logger to use for 4690 OS distribution support. For more information about distribution attributes for WebSphere MQ Managed File Transfer on 4690 OS, see File distribution attributes.. To use the new function for the JEE database logger in Version 7.5.0.2, you must install a new version of the JEE database logger. For more information, see "Installing the new JEE database logger function in Version 7.5.0.2" on page 15.

Enhancements to working with transfers

- You can now submit a large one file to one message transfer, up to a file size of 100 MB. To reduce memory usage for large one file to one message transfers, you are recommended to set the **-qs** parameter on the **fteCreateTransfer** command equal to the size of message being written. If you have a file larger than 100 MB, and you also specify the **-qs** parameter on the **fteCreateTransfer** command, the file is split into multiple messages.

  In the case of recovery of binary file to message transfers, if possible the transfer restarts from the point that the last checkpoint was written. In some cases however, binary transfers will be restarted from the beginning of the file. In the case of text transfer recovery, the transfer always restarts from the beginning of the file. Restarting from the beginning of the file can result in an incomplete group of messages on the destination queue. When the failed text transfer is restarted from the start of the file, a new group of messages is written.

- You can now publish transfer progress log messages for transfers that fail early. You can then use the information that is published about the transfer items in the failed transfer to resubmit that transfer.

- The commandMessagePriority property sets the priority of both internal messages and command messages for the **fteStopAgent**, **fteCancelTransfer**, **ftePingAgent**, and **fteSetAgentTraceLevel** commands. You can also use the commandMessagePriority property to set the priority of internal acknowledgement and acknowledgement-expected messages. You can set commandMessagePriority to a value to prioritize internal WebSphere MQ Managed File Transfer messages above new transfer requests, which can improve agent performance. For more information, see "The installation.properties file" on page 538.

- You can use the maxInlineFileSize property to set the maximum size of file that is included in the transfer request message for single file-to-file or file-to-message transfers. This might improve transfer performance. For more information, see "The agent.properties file" on page 548.

- You can use the enableMemoryAllocationChecking property to ensure that the agent checks whether there is sufficient memory available to run a transfer before the transfer is started. If there is insufficient memory available, the transfer is put into recovery, which prevents the agent from failing with an out-of-memory error. For more information, see "The agent.properties file" on page 548.

- Transfer log publications for file-to-message and message-to-file transfers now contain all the transfer request attributes.

Installing the new JEE database logger function in Version 7.5.0.2

▶ V 7.5.0.2

If you want to use the new function for the JEE database logger in Version 7.5.0.2, you must install a new EAR file.

Install one of the following Version 7.5.0.2 EAR files into your application server:
- Db2: `com.ibm.wmqfte.databaselogger.jee.7502func.ear`
- Oracle: `com.ibm.wmqfte.databaselogger.jee.oracle.7502func.ear`

WebSphere Application Server Community Edition

To install and deploy the new EAR file, complete steps 3 and 4 of Installing the WebSphere MQ Managed File Transfer JEE database logger with WebSphere Application Server Community Edition.

WebSphere Application Server Version 7

To install and deploy the new EAR file, complete step 9 of Installing the WebSphere MQ Managed File Transfer JEE database logger with WebSphere Application Server Version 7.0.

What's new in Version 7.5.0.1?

▶ V 7.5.0.1

Learn about the main new functions in WebSphere MQ Managed File Transfer Version 7.5.0.1.

New options to control line separators
The feature to support user-defined metadata transfer options allows you to specify additional transfer options that do not have a specific **fteCreateTransfer** parameter defined. For this release, three user-defined metadata transfer options are supported, which can control line separator generation for text transfers to and from record-oriented files, such as z/OS® data sets. For details about these options and how to specify them, see "Supported user-defined metadata keys" on page 566 and fteCreateTransfer -md parameter.

Agent diagnostic information
You can view diagnostic information about a local agent by specifying the **-d** parameter on the **fteShowAgentDetails** command. This information includes current transfers, scheduled transfers, monitors, and agent queue depths. You can use this information to determine the health and status of a local agent. For more information, see "fteShowAgentDetails (display WebSphere MQ Managed File Transfer agent details)" on page 522.

Web Gateway and cross-site request forgery (CSRF) protection
You can enable protection for the Web Gateway against CSRF attacks using the **CSRFProtection** initialization parameter. For more information, see "Optional security for the Web Gateway" on page 114.

Connect:Direct platform support on UNIX

Connect:Direct is now supported on AIX, HP-UX, and Solaris. For more information, see WebSphere MQ System Requirements.

What's new in WebSphere MQ File Transfer Edition?

Learn about the main new functions in previous versions of WebSphere MQ File Transfer Edition.

- What's new in V7.0.4.1?
- What's new in V7.0.4?
- What's new in V7.0.3?
- What's new in V7.0.2?
- What's new in V7.0.1?

Installing WebSphere MQ Managed File Transfer

This topic summarizes what you must do to install WebSphere MQ Managed File Transfer.

From Version 7.5, WebSphere MQ Managed File Transfer is installed as a component of IBM WebSphere MQ on UNIX platforms and Windows and is no longer installed as a separate product.

Product options

WebSphere MQ Managed File Transfer can be installed as four different options, depending on your operating system and overall setup. These options are WebSphere MQ Managed File Transfer Agent, WebSphere MQ Managed File Transfer Logger, WebSphere MQ Managed File Transfer Service, or WebSphere MQ Managed File Transfer Tools.

To decide which components to install, review the product options and topology information in the following topics:
- "WebSphere MQ Managed File Transfer product options" on page 4
- "WebSphere MQ Managed File Transfer topology overview" on page 10

How to install

For an overview of IBM WebSphere MQ installation on UNIX platforms and Windows, see Installing and uninstalling.

For information about which specific Managed File Transfer components to install for your platform, see Choosing what to install.

Related concepts:
"WebSphere MQ Managed File Transfer product options" on page 4
WebSphere MQ Managed File Transfer can be installed as four different options, depending on your operating system and overall setup. These options are WebSphere MQ Managed File Transfer Agent, WebSphere MQ Managed File Transfer Logger, WebSphere MQ Managed File Transfer Service, or WebSphere MQ Managed File Transfer Tools.
"WebSphere MQ Managed File Transfer topology overview" on page 10

Related reference:
"Installed command sets" on page 401
The following table shows which commands are installed with each component.

Changes between WebSphere MQ File Transfer Edition V7.0.4 or earlier and WebSphere MQ V7.5

If you are planning to move from WebSphere MQ File Transfer Edition V7.0.4, or earlier to WebSphere MQ V7.5, review the following information that summarizes the changes between versions.

Configuration changes

The configuration layout directly after installation in V7.5 is different to the configuration layout directly after installation in WebSphere MQ File Transfer Edition V7.0.4, or earlier.

For example, the diagram shows the configuration layout directly after installation firstly as it was in WebSphere MQ File Transfer Edition V7.0.4.1 and then as it is in WebSphere MQ V7.5.

Websphere MQ Managed File Transfer V7.0.4.1

```
var
  └──IBM
      └──WMQFTE
          └──config
              ├──wmqfte.properties
              ├──COORDQM1
              │   ├──command.properties
              │   ├──coordination.properties
              │   ├──databaselogger.properties
              │   └──agents
              │       ├──AGENT1
              │       │   ├──agent.properties
              │       │   └──exits
              │       └──AGENT2
              │           └── _ _
              └──COORDQM2
                  └──agents
                      └──AGENT3
                          └── _ _
```

WebSphere MQ Managed File Transfer configuration in WebSphere MQ V7.5

```
var
  └──mqm
      └──mqft
          ├──config
          │   ├──COORDQM1
          │   │   ├──command.properties
          │   │   ├──coordination.properties
          │   │   ├──agents
          │   │   │   └──AGENT1
          │   │   │       ├──agent.properties
          │   │   │       └──exits
          │   │   │   └──AGENT2
          │   │   │       └── _ _
          │   │   └──loggers
          │   │       └──LOGGER1
          │   │           └──logger.properties
          │   └──COORDQM2
          │       └──agents
          │           └──AGENT3
          │               └── _ _
          ├──installations
          │   ├──INSTALLATION1
          │   │   └──installation.properties
          │   └──INSTALLATION2
          │       └──installation.properties
          ├──ipc
          │   └── _ _
          └──logs
              ├──COORDQM1
              │   ├──agents
              │   │   ├──AGENT1
              │   │   │   └── _ _
              │   │   └──AGENT2
              │   │       └── _ _
              │   └──loggers
              │       └──LOGGER1
              │           └── _ _
              └──COORDQM2
                  └──agents
                      └──AGENT3
                          └── _ _
```

WebSphere MQ File Transfer Edition V7.0.4, or earlier file name and default location	V7.5 equivalent file name and default location

Default configuration directory location (wmqfte_configuration_directory):
 	
• UNIX systems: `/var/IBM/WMQFTE/config`
• Linux systems: `/var/ibm/WMQFTE/config`
• Windows: `C:\Documents and Settings\All Users\Application Data\IBM\WMQFTE\config`
	Default configuration directory location and content:
	
	Information that was previously in the WebSphere MQ File Transfer Edition configuration directory is split over four separate subdirectories: `config`, `installations`, `ipc`, and `logs`. The default product root directories (MQ_DATA_PATH) are as follows:
	
	• UNIX systems: `/var/mqm`
	• Linux systems: `/var/mqm`
	• Windows: the location of the configuration directory depends on the location of your primary IBM WebSphere MQ installation. The default locations for primary installations are as follows:
	
	   – 32-bit: `C:\Program Files\IBM\WebSphere MQ`
	   – 64-bit: `C:\Program Files (x86)\IBM\WebSphere MQ`
	
	The configuration subdirectories are as follows:
	
	• The *MQ_DATA_PATH*`/mqft/config` directory contains the parts of the configuration that are read-only for Managed File Transfer processes. For example, `agent.properties` and `command.properties`.
	• The *MQ_DATA_PATH*`/mqft/installations` directory contains configuration information for each installation. The content of this directory is equivalent to the content of the `wmqfte.properties` file.
	• The *MQ_DATA_PATH*`/mqft/ipc` directory contains IPC resources used internally to communicate between the Managed File Transfer components. Applicable to UNIX and Linux systems only.
	• The *MQ_DATA_PATH*`/mqft/logs` directory contains the parts of the configuration that are written by Managed File Transfer processes. For example, trace information and log files.

`wmqfte.properties` The `wmqfte.properties` file sets properties that apply to the entire WebSphere MQ File Transfer Edition installation. The default location is *wmqfte_configuration_directory*	`installation.properties` The `installation.properties` file is a renamed and relocated equivalent of the `wmqfte.properties` file. On UNIX and Linux systems, the default location is *MQ_DATA_PATH*`/mqft/`*installations*`/installation_name` On Windows, the default location is *MQ_DATA_PATH*`\mqft\installations\`*installation_name*
`databaselogger.properties`. This file contains property information for the stand-alone database logger. The default location is *wmqfte_configuration_directory/ coordination_qmgr_name*	`logger.properties` This file now incorporates property information for stand-alone file loggers, stand-alone database loggers, and JEEE database loggers.The default location is *MQ_DATA_PATH*`/mqft/config/`*coordination_qmgr_name*`/ loggers/`*logger_name*.

Security changes

For WebSphere MQ V7.5, only users who are administrators (members of the mqm group) can run the following list of **fte** commands:

- "fteChangeDefaultConfigurationOptions (change the default configuration options)" on page 424
- "fteCreateAgent (create a WebSphere MQ Managed File Transfer agent)" on page 429
- "fteCreateBridgeAgent (create and configure WebSphere MQ Managed File Transfer protocol bridge agent)" on page 432
- "fteCreateCDAgent (create a Connect:Direct bridge agent)" on page 437
- "fteCreateLogger (create a WebSphere MQ Managed File Transfer logger)" on page 441
- "fteDeleteAgent (delete a WebSphere MQ Managed File Transfer agent)" on page 485
- "fteDeleteLogger (delete a WebSphere MQ Managed File Transfer logger)" on page 487
- "fteMigrateAgent (migrate a WebSphere MQ File Transfer Edition V7.0 agent to WebSphere MQ V7.5" on page 502
- "fteMigrateConfigurationOptions (migrate a WebSphere MQ File Transfer Edition V7.0 configuration to WebSphere MQ V7.5)" on page 504
- "fteMigrateLogger (migrate a WebSphere MQ File Transfer Edition V7.0 database logger to WebSphere MQ V7.5)" on page 505
- "fteModifyAgent (modify a WebSphere MQ Managed File Transfer agent)" on page 507
- "fteModifyLogger (run a WebSphere MQ Managed File Transfer logging application as a Windows service)" on page 509
- "fteSetupCommands (create the command.properties file)" on page 519
- "fteSetupCoordination (set up coordination details)" on page 520

For WebSphere MQ V7.5, only the user that the agent process is running under can run the command "fteSetAgentTraceLevel (set WebSphere MQ Managed File Transfer agent trace level)" on page 345.

Migrating a WebSphere MQ File Transfer Edition V7.0 agent to WebSphere MQ V7.5

Use the `fteMigrateAgent` command to migrate a WebSphere MQ File Transfer Edition V7.0 agent to WebSphere MQ V7.5. If an agent is configured to run as a Windows service, you must complete the optional steps in this task.

Before you begin

Identify instances of the WebSphere MQ File Transfer Edition agent installed as part of the WebSphere MQ File Transfer Edition .

About this task

To migrate to MQ V7.5, first stop the agent, then migrate the queue manager installation to MQ V7.5 and additionally select the File Transfer components. When the queue manager has been migrated, you can migrate the agent's configuration to the MQ V7.5 installation using the `fteMigrateAgent` command. Start the migrated agent using the MQ V7.5 queue manager.

If the agent is connecting to its queue manager using MQ bindings mode and the queue manager is at Version 7.0.1.6 or later, you can alternatively perform a side-by-side migration by using multiple installations and the steps listed below. If you do not wish to perform a side-by-side migration, instead migrate the queue manager and agent using the steps as described above.

If the agent is connecting to its queue manager using MQ bindings mode and the queue manager is at Version 7.0.1.5 or earlier, you must either migrate this installation to 7.0.1.6, to permit a side-by-side migration, or you must migrate this installation directly to MQ V7.5.

If the agent is connecting to its queue manager across a network as an MQ client, you can migrate the agent to MQ V7.5 by completing step 7 only.

If the agent is configured as a Windows service you must run the command with the **-f** parameter. For more information, see "fteMigrateAgent (migrate a WebSphere MQ File Transfer Edition V7.0 agent to WebSphere MQ V7.5" on page 502.

Procedure

1. Install MQ V7.5 alongside the existing queue manager installation, selecting the MQ Server and File Transfer components.
2. Stop the WMQFTE v7.0.x agent.
3. Move the queue manager to the MQ V7.5 installation.
4. Use the **fteMigrateAgent** command to migrate the agent configuration from the WMQFTE V7.0.x installation to the agent capability integrated with MQ V7.5.
5. Optional: If the WMQFTE v7.0.x agent is configured as a Windows service, complete the following optional steps:
   a. Use the **fteModifyAgent** command to reconfigure the agent so that it is no longer a Windows service.
   b. Use the **fteModifyAgent** command from the MQ V7.5 installation to configure the MQ V7.5 version of the agent to be a Windows service.
6. Start the MQ V7.5agent.
7. Optional: To migrate an agent connecting to its queue manager across a network as a client:
   a. Install MQ V7.5 onto the system
   b. Use the **fteMigrateAgent** command to migrate the agent configuration from the WMQFTE v7.0.x installation to the agent capability integrated with MQ V7.5.
   c. Stop the FTE v7.0.x agent.
   d. If the WMQFTE v7.0.x agent is configured as a Windows service, use the **fteModifyAgent** command from the FTE v7.0.x install to re-configure the agent so that it is not run as a Windows service
   e. If the WMQFTE v7.0.x agent is configured as a Windows service, use the **fteModifyAgent** command from the MQ V7.5 install to configure the MQ V7.5 agent as a Windows service.
   f. Start the MQ V7.5 agent.

Migrating a WebSphere MQ File Transfer Edition V7.0 database logger to WebSphere MQ V7.5

Use the **fteMigrateLogger** command to migrate a stand-alone WebSphere MQ File Transfer Edition V7.0 database logger to V7.5. If you have configured the database logger as a Windows service, you must complete extra migration steps.

Before you begin

Identify instances of the stand-alone database logger. You must update database logger instances before other parts of the Managed File Transfer network so that these instances can correctly process the latest versions of the transfer log messages they receive.

About this task

To migrate directly to V7.5, first stop the logger and then migrate this installation to the required version. When this migration is complete, use the **fteMigrateLogger** command to migrate the database logger configuration to required version of MQ.

If the queue manager is MQ 7.0.1.6 or later, you can alternatively perform a side-by-side migration by using multiple installations as detailed in the following steps. If you do not wish to perform a side-by-side migration, instead migrate the queue manager and logger using the steps as described above.

If the database logger is connected to a MQ V7.0.1.5 or earlier queue manager, you must first either migrate this installation to 7.0.1.6, to permit a side-by-side migration, or you must migrate this installation directly to MQ V7.5, and additionally select the File Transfer components.

If the database logger is configured as a Windows service you must run the **fteMigrateLogger** command with the **-f** parameter. For more information, see "fteMigrateLogger (migrate a WebSphere MQ File Transfer Edition V7.0 database logger to WebSphere MQ V7.5)" on page 505.

Procedure

1. Install WebSphere MQ V7.5, alongside the existing WebSphere MQ File Transfer Edition V7.0 installation, selecting the MQ Server and Managed File Transfer Service components.
2. Stop the WebSphere MQ File Transfer Edition V7.0 database logger.
3. Move the queue manager to the MQ V7.5 installation.
4. Use the **fteMigrateLogger** command to migrate the database logger configuration from the V7.0 installation to the WebSphere MQ V7.5 database logger.
5. Create the tables needed for the database logger to start by running the sql scripts in `MQ_INSTALLATION_PATH`/mqft/sql. There is one script for each increase in product level, run all that are appropriate in order. You must run the upgrade sql scripts in version order, starting with their current level of WebSphere MQ File Transfer Edition or Managed File Transfer. The available scripts, where *** can be Db2, Oracle are as follows:
   - `ftelog_tables_***_701-702.sql`
   - `ftelog_tables_***_702-703.sql`
   - `ftelog_tables_***_703-704.sql`
   - `ftelog_tables_***_704-750.sql`
   - `ftelog_tables_***_750-7502.sql`
6. Optional: If the WebSphere MQ File Transfer Edition V7.0 database logger was configured as a Windows service, complete the following steps:
   a. Reconfigure the database logger so that it is no longer a Windows service using the fteModifyDatabaseLogger command.
   b. Reconfigure the WebSphere MQ Managed File Transfer V7.5 logger so that it is a Windows service using the **fteModifyLogger** command.
7. Start the WebSphere MQ Managed File Transfer V7.5 database logger.

Results

The database logger has now been migrated from WebSphere MQ File Transfer Edition V7.0 to WebSphere MQ V7.5.

Migrating a WebSphere Application Server V7 Web Gateway from WebSphere MQ File Transfer Edition V7.0 to WebSphere MQ V7.5

To migrate the Web Gateway on WebSphere Application Server V7 from WebSphere MQ File Transfer Edition V7.0 to WebSphere MQ V7.5, complete the following steps:

Before you begin
- Make a note of several of your existing settings before migrating.
- Open the WebSphere Application Server console.

About this task

Procedure
1. Open the WebSphere Application Server console.
2. Click **Applications** > **Application Types** > **Enterprise Applications**. Locate the WebSphere MQ File Transfer Edition Web Gateway application in the list of applications. If the Web Gateway application is not already stopped, select the application and click **Stop**.
3. Make a note of the configuration settings that you previously set up for the Web Gateway. You will need these later in step 7 on page 25.

    a. Click **Enterprise Applications** > **WebSphere MQ File Transfer Edition Web Gateway** > **Initialize parameters for servlets** and make a note of the settings for the following servlet parameters:
    - webGatewayName
    - agentName
    - coordinationQMgr
    - fileSpaceRoot
    - tempFileUploadDir
    - maxTempFileUploadSpace
    - defaultMQMDUserID

    b. Click **Enterprise Applications** > **WebSphere MQ File Transfer Edition Web Gateway** > **Security role to user/group mapping** and make a note of all entries in the pane.

    c. Click **Enterprise Applications** > **WebSphere MQ File Transfer Edition Web Gateway** > **Context Root For Web Modules** and make a note of all entries in the pane.

    d. Click **Enterprise Applications** > **WebSphere MQ File Transfer Edition Web Gateway** > **Message Driven Bean listener bindings** > **Target Resource JNDI Name** and make a note of the activation specification.

    For example, `jms/WMQFTEActivationSpec`

    e. Click **Enterprise Applications** > **WebSphere MQ File Transfer Edition Web Gateway** > **Resource references - queue connection factory** and make a note of the queue connection factory and the data sources used.

    For example,

    ```
    jms/WMQFTEWebAgentConnectionFactory
    DataSource: jdbc/wmqfte-filespace, jdbc/wmqfte-database values
    ```

4. Uninstall the WebSphere MQ File Transfer Edition Web Gateway application by clicking **Applications** > **Application Types** > **Enterprise Applications**. Select the Web Gateway application and click **Uninstall**.

5. Optional: if you are using multiple installations to migrate to WebSphere MQ V7.5 and the native library path is different, change the path by clicking **Resources** > **JMS providers** > **WebSphere MQ messaging provider**

    For example, if the native library path was: `C:\Program Files\IBM\WebSphere MQ\java\lib`, change the path to: `C:\Program Files\IBM\`*New MQ Installation Location*`\java\lib`

6. Optional: if you are using multiple installations to migrate to WebSphere MQ V7.5, you must associate the queue manager with the new installation using the setmqm command.

7. Reinstall the Web Gateway application, using the information in "Configuring the Web Gateway" on page 165 and the information you recorded earlier in step 3 on page 24.

8. Start the new application by clicking **Applications** > **Application Types** > **Enterprise Applications**. Select the Web Gateway application and click **Start**.

9. Run the IVT tool to verify that Web Gateway application still works. For more information, see "Verifying your Web Gateway installation" on page 187.

Migrating a Web Gateway on WebSphere Application Server Community Edition from WebSphere MQ File Transfer Edition V7.0 to WebSphere MQ V7.5

To migrate a Web Gateway on WebSphere Application Server Community Edition from WebSphere MQ File Transfer Edition V7.0 to WebSphere MQ V7.5, complete the following steps:

About this task

Procedure

1. Open the WebSphere Application Server CE console.

2. Click **Applications** > **Application EARs**. Click **Stop** for the WebSphere MQ Managed File Transfer Web Gateway application.

3. Click **Uninstall** then click **OK** on the **Message from webpage** window.

4. Click **Applications** > **J2EE Connectors** and identify the resource adapter. For example, `default/wmq.jmsra.rar/1328267508796/rar`. Click **Stop** for the resource adapter.

5. Click **Uninstall** then click **OK** on the **Message from webpage** window.

6. Shut down and restart the server.

7. Optional: if you are using multiple installations to migrate to WebSphere MQ V7.5, you must associate the queue manager with the new installation using the setmqm command.

8. Click **Applications** > **Deployer** > **Initialize application**.

9. Optional: if you are using multiple installations to migrate to WebSphere MQ V7.5, set the archive to point to the `wmq.jmsra.rar` file in the new installation, for example, *MQ_INSTALLATION_PATH* `/java/lib/jca/wmq.jmsra.rar`

10. Set the plan to point to the previously used plan file. You do not need to change plan file unless you are changing the queue manager. Ensure that you select the **Start application after installation** check box.

11. Click **Install**. The following messages are displayed:

    ```
    The application was successfully deployed.
    The application was successfully started.
    ```

    If these messages are not displayed, check the server logs for further information. For example, if you see the following exception, restart the server:

    ```
    java.lang.UnsatisfiedLinkError: mqjbnd (Library is already loaded in another ClassLoader)
    ```

12. Before redeploying the Web Gateway, you must update the EAR file. For more information, see the following sections in "Preparing to deploy the Web Gateway with WebSphere Application Server Community Edition" on page 167. The values that you need to enter are the same values in the old xml files that were used for the Web Gateway before migration.

    • "Updating the web.xml file" on page 173

    • "Updating the openejb-jar.xml to configure the Web Gateway to use file spaces" on page 175 (required only if you are using file spaces)

- "Update the deployment plan" on page 176 (required only if you must deploy the Web Gateway for a non-default environment or you are using your own security realm)
- "Update the deployment plan for the administrative console" on page 177 (required only if you want to deploy the Web Gateway administrative console with a non-default context root)

13. Deploy the Web Gateway application. For more information, see "Deploying the Web Gateway with WebSphere Application Server Community Edition" on page 183.

14. Run the IVT application to check the Web Gateway application is correctly installed. For more information, see "Verifying your Web Gateway installation" on page 187.

Migrating a WebSphere Application Server V7 JEE database logger from WebSphere MQ File Transfer Edition V7.0 to WebSphere MQ V7.5

To migrate a JEE database logger on WebSphere Application Server V7 from WebSphere MQ File Transfer Edition V7.0 to WebSphere MQ V7.5, complete the following steps:

About this task

Procedure

1. Open the WebSphere Application Server console.

2. Click **Applications** > **Application Types** > **Enterprise Applications**. Locate the WebSphere MQ File Transfer Edition database logger application in the list of applications. If the database logger application is not already stopped, select the application and click **Stop**.

3. Make a note of the configuration settings that you previously set up for the JEE database logger. You will need these later in step 7.

   a. If you originally made changes from the default settings for EJB modules while installing the database logger (see step 9 on page 158 for more information), click **Enterprise Applications** > **WebSphere MQ File Transfer Edition database logger** > **Environment entries for EJB modules** and make a note of the settings in the pane.

   b. Click **Enterprise Applications** > **WebSphere MQ File Transfer Edition database logger** > **Message Driven Bean listener bindings** and make a note of the activation specification used, the **Target Resource JNDI name** and the **Destination JNDI name**.

   c. Click **Enterprise Applications** > **WebSphere MQ File Transfer Edition database logger** > **Resource references** and make a note of the reject queue connection factory details.

   d. Click **Enterprise Applications** > **WebSphere MQ File Transfer Edition database logger** > **Resource environment entry references** and make a note of the reject queue details.

4. Uninstall the WebSphere MQ File Transfer Edition database logger application by clicking **Applications** > **Application Types** > **Enterprise Applications**. Select the database logger application and click **Uninstall**.

5. Optional: if you are using multiple installations to migrate to WebSphere MQ V7.5, and the native library path is different, change the path by clicking **Resources** > **JMS providers** > **WebSphere MQ messaging provider**

   For example, if the native library path was: `C:\Program Files\IBM\WebSphere MQ\java\lib`, change the path to: `C:\Program Files\IBM\`*New MQ Installation Location*`\java\lib`

6. Optional: if you are using multiple installations to migrate to WebSphere MQ V7.5, you must associate the queue manager with the new installation using the setmqm command.

7. Reinstall the database logger application, using the information in "Installing the WebSphere MQ Managed File Transfer JEE database logger with WebSphere Application Server Version 7.0" on page 155 and the information you recorded earlier in step 3.

8. Start the new database logger by clicking **Applications** > **Application Types** > **Enterprise Applications**. Select the database logger application and click **Start**.

9. To verify the migration, check the database to ensure that entries are being written.

Migrating a JEE database logger on WebSphere Application Server Community Edition from WebSphere MQ File Transfer Edition V7.0 to WebSphere MQ V7.5

To migrate a JEE database logger on WebSphere Application Server Community Edition from WebSphere MQ File Transfer Edition V7.0 to WebSphere MQ V7.5, complete the following steps:

About this task

Procedure

1. Open the WebSphere Application Server CE console.
2. Click **Applications** > **Application EARs**. Click **Stop** for the WebSphere MQ File Transfer Edition database logger application.
3. Click **Uninstall** then click **OK** on the **Message from webpage** window.
4. Click **Applications** > **J2EE Connectors** and identify the resource adapter. For example, `default/wmq.jmsra.rar/1328267508796/rar`. Click **Stop** for the resource adapter.
5. Click **Uninstall** then click **OK** on the **Message from webpage** window.
6. Shut down and restart the server.
7. Optional: if you are using multiple installations to migrate to WebSphere MQ V7.5, you must associate the queue manager with the new installation using the setmqm command.
8. Click **Applications** > **Deployer** > **Install application**. The **Install Application** window appears.
9. Optional: if you are using multiple installations to migrate to WebSphere MQ V7.5, set the archive to point to the `wmq.jmsra.rar` file in the new installation, for example, *MQ_INSTALLATION_PATH* `/java/lib/jca/wmq.jmsra.rar`
10. Set the plan to point to the previously used plan file. You do not need to change the plan file unless you are changing the queue manager. Ensure that you select the **Start application after installation** check box.
11. Click **install**. The following messages are displayed:

    ```
    The application was successfully deployed.
    The application was successfully started.
    ```

    If these messages are not displayed, check the server logs for further information. For example, if you see the following exception, restart the server:

    ```
    java.lang.UnsatisfiedLinkError: mqjbnd (Library is already loaded in another ClassLoader)
    ```
12. Update the EAR file for the JEE database logger supplied with WebSphere MQ V7.5. For more information, see step 3 of Installing the WebSphere MQ Managed File Transfer JEE database logger with WebSphere Application Server Community Edition.
13. Deploy the EAR file by completing step 4 of Installing the WebSphere MQ Managed File Transfer JEE database logger with WebSphere Application Server Community Edition.

Migrating the database tables on Db2 on z/OS

If your database is Db2 on a z/OS system, you must complete the following steps to migrate between different versions of WebSphere MQ File Transfer Edition or WebSphere MQ Managed File Transfer, V7.0.3 to V7.0.4, and V7.0.4 to V8.0.0. You may also use these migration steps to enable use of the BIGINT data type in your database tables. �switchV 7.5.0.1 The BIGINT data type is available in WebSphere MQ File Transfer Edition V7.0.4 Fix Pack 3, or WebSphere MQ Managed File Transfer V7.5 Fix Pack 1 or later.

About this task

To enable use of BIGINT data types, you must be using Db2 V9.1 or later. INTEGER data types are used for fields which denote the sizes of files that are transferred and the table ID associated with each

transfer. If you want to log transfers with file sizes greater than 2 GB, or if you want to store more than 2,147,483,648 individual transfers in your database you must use the BIGINT SQL file.

Procedure

1. If you have not already stopped your database logger, stop your database logger using the **fteStopDatabaseLogger** command.

2. Back up your log database with the tools that are provided by Db2.

3. Create a table space. This table space must have a page size of at least 8 KB and an associated buffer pool with a page size of at least 8 KB. Give your new table space a name. For example, FTENEWSPACE.

4. Create the new tables. Change the schema name in the file from FTELOG to a schema name that is different to the name of your existing schema. If you want to use the BIGINT datatype instead of the INTEGER datatype use the `ftelog_tables_zos_bigint.sql`. Otherwise, use the `ftelog_tables_zos.sql` file.

5. Migrate the data from your old schema to your new schema. To do this migration edit the `ftelog_tables_zos`*old-new*`.sql` file, where *old* is the version of the database tables that exist and *new* is the version of the database tables that you are migrating to. Change FTESRC to the name of your existing schema. Change FTEDEST to the name of the schema that you created in the previous step.

6. Change the database logger properties to refer to the new database schema.

   • If you are using the stand-alone database logger, edit the `logger.properties` file to include the following line:

     `wmqfte.database.schema=`*schema_name*

     In this example, *schema_name* is the name of the schema that you created in step 3.

   • If you are using the JEE database logger, you must change the schema name in the database logger EAR file. For more information, see "Changing the schema name in your Java Platform, Enterprise Edition database logger" on page 154.

7. In tables with generated ID columns, set the ID generators to begin from a value one higher than the existing highest ID value. The following tables have generated ID columns:

   • AUTH_EVENT
   • CALL
   • CALL_ARGUMENT
   • CALL_RESULT
   • FILE_SPACE_ENTRY
   • METADATA
   • MONITOR_ACTION
   • MONITOR_EXIT_RESULT
   • MONITOR_METADATA
   • SCHEDULE
   • SCHEDULE_ACTION
   • SCHEDULE_ITEM
   • SCHEDULE_SPEC
   • TRANSFER_CALLS
   • TRANSFER_CD_NODE
   • TRANSFER_CORRELATOR
   • TRANSFER_EVENT
   • TRANSFER_EXIT
   • TRANSFER_ITEM
   • TRANSFER_ITEM_ATTRIBUTES

- TRANSFER_STATS
- TRIGGER_CONDITION

To set the generated IDs of these tables to the correct value perform the following steps for each table:

a. Determine the maximum ID value in the existing data. You can find this value by running this SQL statement:

```
SELECT MAX(ID) FROM schema_name.table_name
```

The value that is returned from this command is the maximum existing ID in the specified table.

b. Alter the table to set the ID generator to begin from a new value that is 1 higher than the value returned by the previous step. You can set this value by running the following SQL statement:

```
ALTER TABLE schema_name.table_name ALTER COLUMN ID RESTART WITH value
```

In these statements, *schema_name* is the name of the schema that you created in step 3.

Related tasks:

"Migrating from the stand-alone database logger to the JEE database logger" on page 164
You can migrate from the stand-alone database logger to the JEE database logger. You must stop the stand-alone database logger and install the JEE database logger. To avoid losing or duplicating log entries you must stop messages being published to the SYSTEM.FTE topic before stopping the stand-alone database logger, and restart it after you have installed the JEE database logger. Back up your database before migration. .

"Increasing the page size of the log database on Db2 on Windows, UNIX or Linux"
If your database is Db2 on a Windows, UNIX or Linux system, and you created your log database with a page size of less than 8 KB, you must increase the page size of the database before migrating to the V7.0.3 or later tables.

Increasing the page size of the log database on Db2 on Windows, UNIX or Linux

If your database is Db2 on a Windows, UNIX or Linux system, and you created your log database with a page size of less than 8 KB, you must increase the page size of the database before migrating to the V7.0.3 or later tables.

Procedure

1. If you have not already stopped your database logger, stop your database logger using the **fteStopDatabaseLogger** command.

2. Back up your log database using the tools provided by Db2.

3. Use the Db2 **export** command to transfer the data from your log database tables to files on disk.

   Note: You must specify large object files for tables that include large objects. Those tables are CALL_RESULT and METADATA.

4. Drop your log database tables.

5. Create a table space with a page size of at least 8 KB and with an associated buffer pool with a page size of at least 8 KB. Give your new table space a name. For example, FTE8KSPACE.

6. Edit the `ftelog_tables_db2.sql` file so that the commands create tables in the new table space. In the `ftelog_tables_db2.sql` file, change all occurrences of the text IN "USERSPACE1" to IN "*new_tablespace_name*". For example, change IN "USERSPACE1" to IN "FTE8KSPACE".

7. Run the SQL commands in the `ftelog_tables_db2.sql` file against your database.

8. Use the Db2 **load** command to transfer the exported data into the new tables.

   Note:

- **Map the column names based on the column names found in the input file**. Ensure that the input column names and target column names match up in those tables that have changed their structure.
- You must specify the IDENTITY OVERRIDE behavior on the identity column of all tables, except for MONITOR and TRANSFER. Specifying this behavior ensures that the row IDs are not regenerated during the load operation.

9. Run the Db2 `set integrity` command with integrity status values of **immediate** and **checked**, against the following tables in the order given:
   - CALL_ARGUMENT
   - MONITOR
   - MONITOR_ACTION
   - MONITOR_EXIT_RESULT
   - MONITOR_METADATA
   - SCHEDULE_ACTION
   - SCHEDULE
   - SCHEDULE_ITEM
   - TRANSFER
   - TRANSFER_CALLS
   - TRANSFER_EVENT
   - TRANSFER_ITEM
   - TRANSFER_STATS
   - TRIGGER_CONDITION

10. In tables with generated ID columns, set the ID generators to begin from a value one higher than the existing highest ID value. The following tables have generated ID columns:
   - AUTH_EVENT
   - CALL
   - CALL_ARGUMENT
   - CALL_RESULT
   - FILE_SPACE_ENTRY
   - METADATA
   - MONITOR_ACTION
   - MONITOR_EXIT_RESULT
   - MONITOR_METADATA
   - SCHEDULE
   - SCHEDULE_ACTION
   - SCHEDULE_ITEM
   - SCHEDULE_SPEC
   - TRANSFER_CALLS
   - TRANSFER_CD_NODE
   - TRANSFER_CORRELATOR
   - TRANSFER_EVENT
   - TRANSFER_EXIT
   - TRANSFER_ITEM
   - TRANSFER_ITEM_ATTRIBUTES
   - TRANSFER_STATS
   - TRIGGER_CONDITION

To set the generated IDs of these tables to the correct value perform the following steps for each table:

a. Determine the maximum ID value in the existing data. You can find this value by running this SQL statement:

```
SELECT MAX(ID) FROM FTELOG.table_name
```

The value returned from this command is the maximum existing ID in the specified table.

b. Alter the table to set the ID generator to begin from a new value that is 1 higher than the value returned by the previous step. You can set this value by running the following SQL statement:

```
ALTER TABLE FTELOG.table_name ALTER COLUMN ID RESTART WITH value
```

Related tasks:

"Migrating from the stand-alone database logger to the JEE database logger" on page 164
You can migrate from the stand-alone database logger to the JEE database logger. You must stop the stand-alone database logger and install the JEE database logger. To avoid losing or duplicating log entries you must stop messages being published to the SYSTEM.FTE topic before stopping the stand-alone database logger, and restart it after you have installed the JEE database logger. Back up your database before migration. .

Migrating the database tables on DB2 on z/OS

If your database is Db2 on a z/OS system, you must complete the following steps to migrate between different versions of WebSphere MQ File Transfer Edition or WebSphere MQ Managed File Transfer, V7.0.3 to V7.0.4, and V7.0.4 to V8.0.0. You may also use these migration steps to enable use of the BIGINT data type in your database tables. ▶ **V 7.5.0.1** The BIGINT data type is available in WebSphere MQ File Transfer Edition V7.0.4 Fix Pack 3, or WebSphere MQ Managed File Transfer V7.5 Fix Pack 1 or later.

About this task

To enable use of BIGINT data types, you must be using Db2 V9.1 or later. INTEGER data types are used for fields which denote the sizes of files that are transferred and the table ID associated with each transfer. If you want to log transfers with file sizes greater than 2 GB, or if you want to store more than 2,147,483,648 individual transfers in your database you must use the BIGINT SQL file.

Procedure

1. If you have not already stopped your database logger, stop your database logger using the **fteStopDatabaseLogger** command.
2. Back up your log database using the tools provided by Db2.
3. Use the Db2 **export** command to transfer the data from your log database tables to files on disk.

   Note: You must specify large object files for tables that include large objects. Those tables are CALL_RESULT and METADATA.

4. Drop your log database tables.
5. Create a table space with a page size of at least 8 KB and with an associated buffer pool with a page size of at least 8 KB. Change the schema name in the file from FTELOG to a schema name that is different to the name of your existing schema. If you want to use the BIGINT datatype instead of the INTEGER datatype use the ftelog_tables_zos_bigint.sql. Otherwise, use the ftelog_tables_zos.sql file.
6. Edit the ftelog_tables_zos.sql file so that the commands create tables in the new table space.
7. Run the SQL commands in the ftelog_tables_zos.sql file against your database.
8. Use the Db2 **load** command to transfer the exported data into the new tables.

   Note:

- **Map the column names based on the column names found in the input file**. Ensure that the input column names and target column names match up in those tables that have changed their structure.

- You must specify the IDENTITY OVERRIDE behavior on the identity column of all tables, except for MONITOR and TRANSFER. Specifying this behavior ensures that the row IDs are not regenerated during the load operation.

9. Run the Db2 `set integrity` command with integrity status values of **immediate** and **checked**, against the following tables in the order given:
   - CALL_ARGUMENT
   - MONITOR
   - MONITOR_ACTION
   - MONITOR_EXIT_RESULT
   - MONITOR_METADATA
   - SCHEDULE_ACTION
   - SCHEDULE
   - SCHEDULE_ITEM
   - TRANSFER
   - TRANSFER_CALLS
   - TRANSFER_EVENT
   - TRANSFER_ITEM
   - TRANSFER_STATS
   - TRIGGER_CONDITION

10. In tables with generated ID columns, set the ID generators to begin from a value one higher than the existing highest ID value. The following tables have generated ID columns:
    - AUTH_EVENT
    - CALL
    - CALL_ARGUMENT
    - CALL_RESULT
    - METADATA
    - MONITOR_ACTION
    - MONITOR_EXIT_RESULT
    - MONITOR_METADATA
    - SCHEDULE
    - SCHEDULE_ACTION
    - SCHEDULE_ITEM
    - SCHEDULE_SPEC
    - TRANSFER_CALLS
    - TRANSFER_EVENT
    - TRANSFER_ITEM
    - TRANSFER_STATS
    - TRIGGER_CONDITION

    To set the generated IDs of these tables to the correct value perform the following steps for each table:

    a. Determine the maximum ID value in the existing data. You can find this value by running this SQL statement:
    ```
    SELECT MAX(ID) FROM FTELOG.table_name
    ```

The value returned from this command is the maximum existing ID in the specified table.

b. Alter the table to set the ID generator to begin from a new value that is 1 higher than the value returned by the previous step. You can set this value by running the following SQL statement:

```
ALTER TABLE FTELOG.table_name ALTER COLUMN ID RESTART WITH value
```

Using WebSphere MQ Managed File Transfer in a retail environment

You can install WebSphere MQ Managed File Transfer on a device that is running an IBM 4690 operating system. To complete this type of installation, you must first install WebSphere MQ Managed File Transfer on a non-4690 machine. You then collect installation and configuration files on the non-4690 machine and transfer them to the IBM 4690 machine. You then use these files to install WebSphere MQ Managed File Transfer on the IBM 4690 machine.

How to install

For an overview of IBM WebSphere MQ installation, see Installing and uninstalling.

For information about which specific MQMFT components to install for your platform, see Choosing what to install.

For information about preparing to install MQMFT in a retail environment, see "Preparing to install WebSphere MQ Managed File Transfer on an IBM 4690 system" on page 61.

Related reference:
"Restrictions when running on a 4690 OS system" on page 85
There are a number of restrictions and unsupported functions when you run WebSphere MQ Managed File Transfer on a 4690 OS system in a retail environment.

Scenarios in a retail environment

This section provides scenarios on how to use the IBM WebSphere MQ Version 7.5.0.2 support for the IBM 4690 operating system.

Subtopics
- "1. Get started with file transfers using a 4690 OS in store" on page 36

  This scenario explains how to get started with WebSphere MQ Managed File Transfer on the 4690 OS platform. This scenario helps explain the special considerations when you deploy a WebSphere MQ Managed File Transfer agent to the 4690 OS platform.
- "2. Transferring files from head office to a 4690 OS system in store" on page 47

  Creates a two computer topology representative of one computer at a head-office site, and one a 4690 OS store controller at a retail store. Learn how to create the definitions and authorizations that are required to transfer files from the head-office site to a 4690 OS system that is deployed at the retail store.
- "3. Transferring files from a 4690 OS system in store to head office" on page 53

  Covers creating the definitions that are required to automatically transfer files that are created on the 4690 OS system to the computer at head office. Find out how the WebSphere MQ Managed File Transfer concept of resource monitoring can be used to automatically transfer any file that is created inside a particular directory.

1. Get started with file transfers using a 4690 OS in store

You can use this scenario to help you get started with WebSphere MQ Managed File Transfer on the 4690 OS.

Transferring files with control, reliability, and an audit trail, can be a fundamental requirement for exchanging data between retail stores and a retail head office. This scenario provides you with a basic understanding of how to use WebSphere MQ Managed File Transfer to exchange data between a computer in your head office and a 4690 OS store controller that is deployed in a retail store. Scenarios in later topics demonstrate different patterns of interaction. These different patterns are required to show how the Managed File Transfer component can address real business problems that are encountered by retailers.

To work through this scenario, you should have a basic understanding of IBM WebSphere MQ and WebSphere MQ Managed File Transfer. Specifically, you should be familiar with the following ideas:
* Concept of a queue manager
* Basic configuration and administration of IBM WebSphere MQ
* Concept of a IBM WebSphere MQ agent
* Basic configuration and administration of WebSphere MQ Managed File Transfer

For more information about the WebSphere MQ Managed File Transfer capability, see "WebSphere MQ Managed File Transfer introduction" on page 1.

In this scenario, IBM WebSphere MQ can be used to initiate and track a transfer of a file from an 4690 OS back to the same system. The transfer in this scenario helps you understand scenarios in later topics, such as transferring files from the head-office computer to a 4690 OS in-store, and transferring files from an in-store 4690 OS to a head-office site.

Example file transfer topology

This and subsequent scenarios are based around a hub and spoke topology. This diagram shows conceptual hub and spoke topology that comprises the hub (H) and multiple spokes (S).
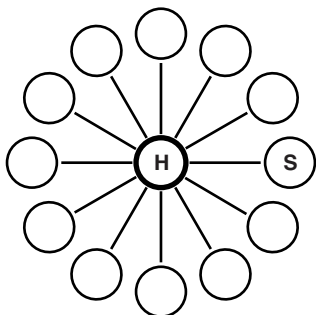
A hub and spoke topology is representative of the file transfer requirements of many retailers. In this case, the hub corresponds to a central head office site and the spokes correspond to the stores operated by the retailer. Often software configuration, deployment, and administration would take place at the head-office site.

In this scenario you can complete the following tasks:
* Plan the solution.
* Configure IBM WebSphere MQ for file transfers on a 4690 OS.
* Create a configuration for an agent that is running on a 4690 OS.

- Deploy an agent to a 4690 OS.
- Verify the scenario by transferring a file.

Related concepts:

"Using WebSphere MQ Managed File Transfer in a retail environment" on page 35
You can install WebSphere MQ Managed File Transfer on a device that is running an IBM 4690 operating system. To complete this type of installation, you must first install WebSphere MQ Managed File Transfer on a non-4690 machine. You then collect installation and configuration files on the non-4690 machine and transfer them to the IBM 4690 machine. You then use these files to install WebSphere MQ Managed File Transfer on the IBM 4690 machine.

"Plan the solution"
To use WebSphere MQ Managed File Transfer on the 4690 OS, you must have a configuration that includes at least 2 computers.

"Deploy an agent to a 4690 OS system" on page 46
Implementing the solution that is described by this scenario requires the deployment of a WebSphere MQ Managed File Transfer agent on a 4690 OS. The agent is started by configuring it as a 4690 OS background application.

"Verify the scenario by transferring a file" on page 46
Verify the topology that is built in this scenario by transferring a file from the 4690 OS system (COMPUTER2) back to the host 4690 OS system. The task assumes that you have a Windows system. For a UNIX system, substitute appropriate paths and commands.

Related tasks:

"Configuring WebSphere MQ for file transfers" on page 42
Configure IBM WebSphere MQ for file transfers by issuing commands to build the topology for the basic file transfer scenario.

"Creating a configuration for an agent on a 4690 OS system" on page 44
Implementing the solution that is described by this scenario requires the creation of a configuration bundle. A configuration bundle packages together all the configuration that is required for a 4690 OS WebSphere MQ Managed File Transfer agent.

Plan the solution

To use WebSphere MQ Managed File Transfer on the 4690 OS, you must have a configuration that includes at least 2 computers.

A minimal configuration includes a WebSphere MQ Managed File Transfer agent on a 4690 OS and requires two computers:

- A computer that is deployed at the head-office site.
- A 4690 OS store controller computer that is deployed in a retail store.
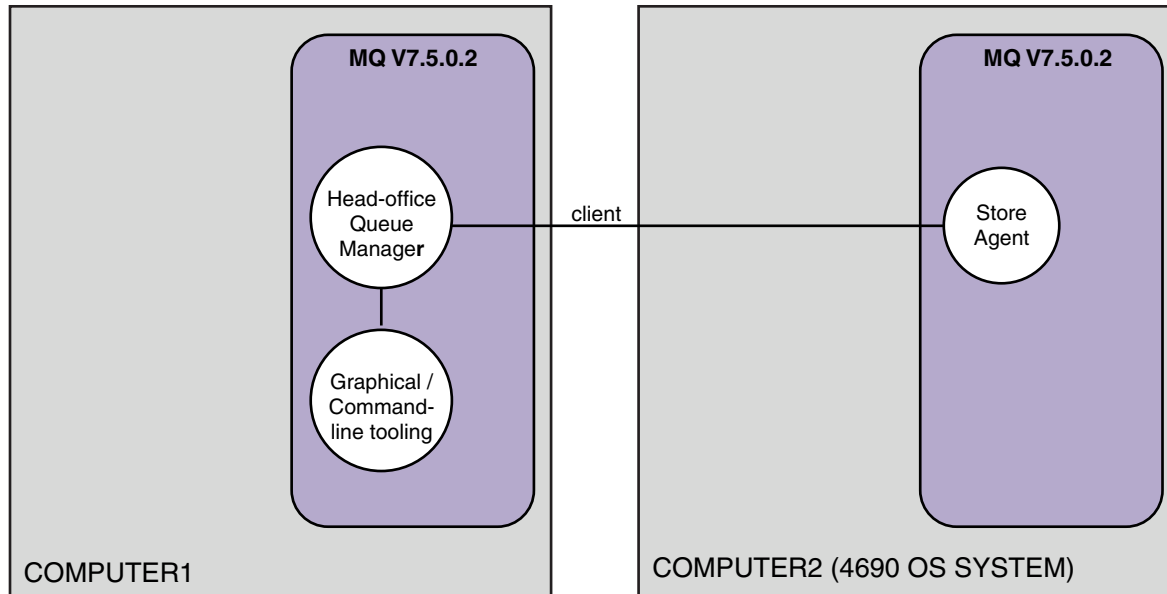
This diagram shows the head office and store topology that spans two computers. COMPUTER1 is the head-office computer. COMPUTER2 is the 4690 OS store controller, which would typically be in a retail store. The WebSphere MQ Managed File Transfer agent on COMPUTER2 connects to the queue manager on COMPUTER1 over a client connection.

The computer that is deployed at the head-office site is used to complete the following activities:
1. Create a configuration for the WebSphere MQ Managed File Transfer agent that is running on the 4690 OS system.
2. Run a IBM WebSphere MQ queue manager that provides connectivity for the WebSphere MQ Managed File Transfer components.
3. Run the graphical or command-line tool that is used to configure and administer managed file transfer operations.

For simplicity, this scenario uses a single computer to complete all of the activities carried out at the head-office site. You must consider whether your deployment would benefit from using more than one computer to complete these activities.

The 4690 OS computer, which is deployed at the retail store, runs the 4690 OS WebSphere MQ Managed File Transfer agent component. In this scenario, a file is transferred from the retail store to the head office, and then back to the retail store. Although not representative of a real use case, transferring a file validates that this scenario is correctly configured, and acts as the foundation for subsequent scenarios that exchange file data between the head-office computer and the 4690 OS computer at the retail store.
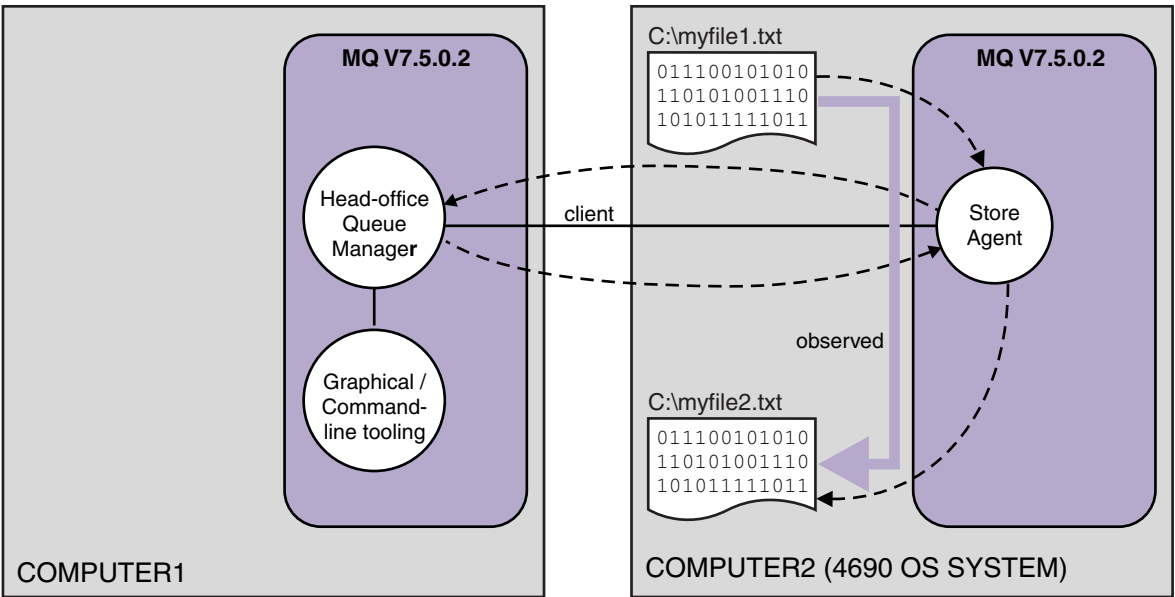
This diagram shows the file transfer route that is demonstrated by this scenario. In this example, assume that COMPUTER1 is a Windows computer and COMPUTER2 is a 4690 OS store controller. You might want to use an alternative platform or architecture for COMPUTER1; for a full list of supported platforms, see http://www.ibm.com/support/docview.wss?uid=swg27006467

The scenario assumes that you have a Windows system. For a UNIX system, substitute appropriate paths and commands, and ensure that you have read and write access to all relevant directories. The scenario also assumes that you have a basic understanding of what a queue manager does.

Understand the security model

This scenario, and scenarios in later topics, create a file transfer topology with the following security characteristics:

- Access to IBM WebSphere MQ is partitioned between four roles:
  1. The IBM WebSphere MQ administrator who configures IBM WebSphere MQ and creates the configuration that is required to complete managed file transfer operations.
  2. FTHOFFS who start, stop, and interact with the agent deployed at head office. This is described in the "2. Transferring files from head office to a 4690 OS system in store" on page 47 scenario.
  3. FTSTORES who represent, to the queue manager at head office, the role that starts, stops, and interacts with agents deployed at a retail store.
  4. FTUSERS who initiate file transfers.

     For more information about configuring access, see "Authorities for resources specific to WebSphere MQ Managed File Transfer" on page 404.
- All connections from retail store based agents are mapped to a single user FTSTORES at the hub.
- Weak IP-based authentication is used to authenticate the agent that is deployed at the retail store.

For clarity, the security model in the scenario is not fully secured. You must understand your own topology needs and security threats before you deploy a similar topology in production. You must therefore consider whether to address the following potential vulnerabilities:

- Any user can impersonate any other. Consider finer granularity in the object access model for file transfer resources.

- Any agent can impersonate any other. Consider stronger authentication, for example, TLS/SSL and finer granularity in the object access model for file transfer resources.
- The interface between file system versus IBM WebSphere MQ security is not considered. Consider implementing file sandboxing, and understand the impact of permissions of the agent's configuration files.
- The interface between the agent and the operating system is not described. Consider implementing file sandboxing. For more information about sandboxing, see Sandboxes.

For more information about security, IBM WebSphere MQ, and file transfers, see What to do next in "Verify the scenario by transferring a file" on page 46.

Prerequisites and licenses

You need the following items to complete this scenario:
- A test computer that satisfies the hardware and operating system prerequisites for IBM WebSphere MQ, for details, see http://www.ibm.com/support/docview.wss?uid=swg27006467, with no existing installation of IBM WebSphere MQ or IBM WebSphere MQ data.
- An 4690 OS store controller that satisfies the hardware and operating system prerequisites for WebSphere MQ Managed File Transfer.
- IBM WebSphere MQ version 7.5.
- IBM WebSphere MQ Fix Pack 7.5.0.2. You can download this fix pack from http://www.ibm.com/support/docview.wss?uid=swg27038184.

Prepare the head-office computer

Ensure your test computer satisfies the requirements for installation of IBM WebSphere MQ version 7.5. For more information, see Checking requirements.

Install a queue manager on the head-office computer

Install IBM WebSphere MQ version 7.5 server with the following components:
- Server
- WebSphere MQ Explorer
- WebSphere MQ Managed File Transfer Agent
- WebSphere MQ Managed File Transfer Command Line tools

For details on which components to install, see Choosing what to install.

Decide how you want to administer IBM WebSphere MQ. You can administer IBM WebSphere MQ by:
- Setting up an appropriate environment by using the **setmqenv** command. For more information, see setmqenv.
- Calling fully qualified IBM WebSphere MQ administrative commands.

The scenario assumes that you are using a clean computer with no previous installations of IBM WebSphere MQ or WebSphere MQ Managed File Transfer installed. If not, you must determine whether coexistence is supported or adjust the installation mechanism and configuration of environments. For details, see Multiple installations.

Prepare the 4690 OS system

Ensure your test 4690 OS store controller satisfies the requirements for installation of WebSphere MQ Managed File Transfer version 7.5.0.2, see Checking requirements.

Install an agent on the 4690 OS system

Install WebSphere MQ Managed File Transfer version 7.5.0.2 to the 4690 OS store controller. For more information, see "Installing IBM WebSphere MQ Managed File Transfer on 4690 OS" on page 62.

Create your groups and users

The security model that is used in this scenario, assumes that you create the following groups and users on the head-office computer:

- Groups
  - mqm

    Created when IBM WebSphere MQ is installed. Members of this group can administer IBM WebSphere MQ and its resources.
  - FTHOFFS

    You must create this group. Members of this group start, stop, and interact with the agent deployed at head office. For more information, see "2. Transferring files from head office to a 4690 OS system in store" on page 47.
  - FTSTORES

    You must create this group. This group is used by the queue manager at head office, to represent the group of users who start, stop, and interact with the agent deployed in the retail store.
  - FTUSERS

    You must create this group. Members of this group can initiate file transfers.
- Users
  - mqmAdmin

    IBM WebSphere MQ administrator. You must create this user:
    - On Windows, this user must be both a member of the mqm group and a Windows administrator to be able to define a WebSphere MQ File Transfer Edition agent that is run as a Windows service
    - On other platforms, this user only needs to be a member of the mqm group.
  - ftuser

    You must create this user, and make it a member of the FTUSERS group. To avoid the potential for administrative level security access to the queue manager, do not add this user to the mqm group, or make this user a Windows administrator.
  - fthoff

    You must create this user, and make it a member of the FTHOFFS group. To avoid the potential for administrative level security access to the queue manager, do not add this user to the mqm group, or make this user a Windows administrator.

    On Windows, this user is used to run the IBM WebSphere MQ Managed File Transfer agent process as a Windows service. The user requires:
    - The account has a password set
    - The account has Log on as a service authority, see: "Guidance for running an agent or logger as a Windows service" on page 366.
  - ftstore

    You must create this user, and make it a member of the FTSTORES group. To stop this user unintentionally having administrative level security access to the queue manager, do not add this user to the mqm group, or make this user a Windows administrator.

Configuring WebSphere MQ for file transfers

Configure IBM WebSphere MQ for file transfers by issuing commands to build the topology for the basic file transfer scenario.

About this task

In this task, you complete the following activities:
- Create a queue manager on the head-office computer.
- Set up the WebSphere MQ object definitions that are required to allow an agent, running on a 4690 OS system, to connect to the queue manager.
- Create the WebSphere MQ object definitions that are required for a basic WebSphere MQ Managed File Transfer topology.
- Apply a basic security model to the topology.

The task assumes that you have a Windows system. For a UNIX system, substitute appropriate paths and commands.

Complete these steps from the user `mqmAdmin`, in the WebSphere MQ bin directory, *`MQ_INSTALL_ROOT`*`\bin`.

Procedure

1. Create a queue manager named `hoffQM`.

   `crtmqm hoffQM`

2. Start the queue manager.

   `strmqm hoffQM`

3. Start the MQSC interface for `hoffQM`.

   `runmqsc hoffQM`

4. Create a channel to handle incoming connections from the WebSphere MQ Managed File Transfer agent that is running on the 4690 OS Store Controller system.

   `DEFINE CHANNEL(FTE.AGENT.SVRCONN) CHLTYPE(SVRCONN)`

5. Create a channel authentication record to allow a connection from the 4690 OS Store Controller system into queue manager `hoffQM`. The channel authentication record restricts which IP addresses can connect to the queue manager and assigns the identity of user ftstore to the connection. This command must be run for each in-store agent.

   `SET CHLAUTH('FTE.AGENT.SVRCONN') TYPE(ADDRESSMAP) ADDRESS('`*`IP address of 4690 OS system`*`') USERSRC(MAP) MCAUSER('ftstor`

   For more information about channel authentication records, see Channel authentication records. The goal of this scenario is not to lock down and secure the topology, but to demonstrate a basic file transfer. The implemented security model supports this demonstration, but you must understand your own security threats and take appropriate actions where necessary. For discussions of options to consider, see What to do next in "Verify the scenario by transferring a file" on page 46.

6. Identify a free port that can be used for network communications with the queue manager that is running on the head-office system, for example 1414. Define a listener LISTENER1 to use this free port.

   `DEFINE LISTENER(LISTENER1) TRPTYPE(TCP) CONTROL(QMGR) PORT(1414)`

7. Start the listener `LISTENER1`.

   `START LISTENER(LISTENER1)`

8. End the MQSC interface for queue manager `hoffQM`.

   `end`

9. Configure queue manager `hoffQM` as the coordination queue manager for the WebSphere MQ Managed File Transfer topology.

   `fteSetupCoordination -coordinationQMgr hoffQM`

For more information about the coordination queue manager role, see "WebSphere MQ Managed File Transfer topology overview" on page 10.

10. Use the MQSC interface to define the WebSphere MQ objects that are required to make queue manager `hoffQM` a coordination queue manager.

   ```
   runmqsc hoffQM < ..\mqft\config\hoffQM\hoffQM.mqsc
   ```

11. Configure queue manager `hoffQM` as the command queue manager for the WebSphere MQ Managed File Transfer topology.

   ```
   fteSetupCommands -connectionQMgr hoffQM
   ```

12. Create the object definitions that are required for an agent, SAGENT, on queue manager `hoffQM`. This command must be run for each in-store agent.

   ```
   fteDefine -t agent SAGENT | runmqsc hoffQM
   ```

   The **fteDefine** command was introduced in the WebSphere MQ 7.5.0.2 fix pack. If you cannot locate this command in your WebSphere MQ installation, check the service level of the installation by using the **dspmqver** command. For more information about the **fteDefine** command, see "fteDefine (generate configuration scripts)" on page 483.

13. Ensure that the FTSTORES and FTUSERS groups have appropriate access to the WebSphere MQ objects required to complete file transfer operations. You can tailor this configuration to suite your own security requirements.

   ```
   setmqaut -m hoffQM -t qmgr -g FTSTORES +connect +inq +setid +altusr
   setmqaut -m hoffQM -n SYSTEM.FTE -t queue -g FTSTORES +get +put
   setmqaut -m hoffQM -n SYSTEM.FTE -t topic -g FTSTORES +pub +sub
   setmqaut -m hoffQM -n SYSTEM.DEFAULT.MODEL.QUEUE -t queue -g FTSTORES +browse +dsp +get +put
   setmqaut -m hoffQM -t qmgr -g FTUSERS +connect
   setmqaut -m hoffQM -n SYSTEM.FTE -t topic -g FTUSERS +sub
   setmqaut -m hoffQM -n SYSTEM.DEFAULT.MODEL.QUEUE -t queue -g FTUSERS +browse +dsp +get +put
   ```

   The following commands must be run for each in-store agent.

   ```
   setmqaut -m hoffQM -n SYSTEM.FTE.COMMAND.SAGENT -t queue -g FTSTORES +browse +get +put +setid
   setmqaut -m hoffQM -n SYSTEM.FTE.DATA.SAGENT -t queue -g FTSTORES +get +put
   setmqaut -m hoffQM -n SYSTEM.FTE.EVENT.SAGENT -t queue -g FTSTORES +browse +get +put
   setmqaut -m hoffQM -n SYSTEM.FTE.REPLY.SAGENT -t queue -g FTSTORES +browse +get +put
   setmqaut -m hoffQM -n SYSTEM.FTE.STATE.SAGENT -t queue -g FTSTORES +browse +get +inq +put
   setmqaut -m hoffQM -n SYSTEM.FTE.COMMAND.SAGENT -t queue -g FTUSERS +put
   ```

   For more information about the **setmqaut** command, see setmqaut. For more information about granting authority to groups, see "Group authorities for resources specific to WebSphere MQ Managed File Transfer" on page 405.

14. Set up the authorization that is required to allow a file to be sent from the 4690 OS system, back to the 4690 OS system. These commands need to be run against only one in-store agent so you can verify the installation. These authorizations are removed at the end of scenario 2.

   ```
   setmqaut -m hoffQM -n SYSTEM.FTE.AUTHTRN1.SAGENT -t queue -g FTUSERS +browse +put
   setmqaut -m hoffQM -n SYSTEM.FTE.AUTHAGT1.SAGENT -t queue -g FTSTORES +browse +put
   ```

   For more information about granting authority for specific WebSphere MQ Managed File Transfer actions, see "User authorities on WebSphere MQ Managed File Transfer actions" on page 409.

Related concepts:

"1. Get started with file transfers using a 4690 OS in store" on page 36
You can use this scenario to help you get started with WebSphere MQ Managed File Transfer on the 4690 OS.

"Using WebSphere MQ Managed File Transfer in a retail environment" on page 35
You can install WebSphere MQ Managed File Transfer on a device that is running an IBM 4690 operating system. To complete this type of installation, you must first install WebSphere MQ Managed File Transfer on a non-4690 machine. You then collect installation and configuration files on the non-4690 machine and transfer them to the IBM 4690 machine. You then use these files to install WebSphere MQ Managed File Transfer on the IBM 4690 machine.

Creating a configuration for an agent on a 4690 OS system

Implementing the solution that is described by this scenario requires the creation of a configuration bundle. A configuration bundle packages together all the configuration that is required for a 4690 OS WebSphere MQ Managed File Transfer agent.

About this task

Configuring 4690 OS Managed File Transfer agents is not the same as configuring agents on Windows or UNIX.

Configuring a 4690 OS agent is a two-step task:

1. A configuration bundle is created by using the **fteBundleConfiguration** utility that is supplied with WebSphere MQ Managed File Transfer. The configuration bundle packages together all of the information that is required to configure a 4690 OS agent.

2. The configuration bundle is transferred to the 4690 OS store controller where it is deployed to the WebSphere MQ Managed File Transfer installation.

The reasons for choosing this style of configuration include the following:

- Centralizes the WebSphere MQ Managed File Transfer knowledge that is required to configure a network of 4690 OS agents. For example, in a retail scenario, these skills might be concentrated at the head-office site, with little or no IT skills present at individual retail stores.
- Provides a mechanism by which one configuration bundle can be deployed to many 4690 OS systems. This method reduces the chance of inconsistencies between the configuration that is used on different systems. For more information, see "Verify the scenario by transferring a file" on page 46.
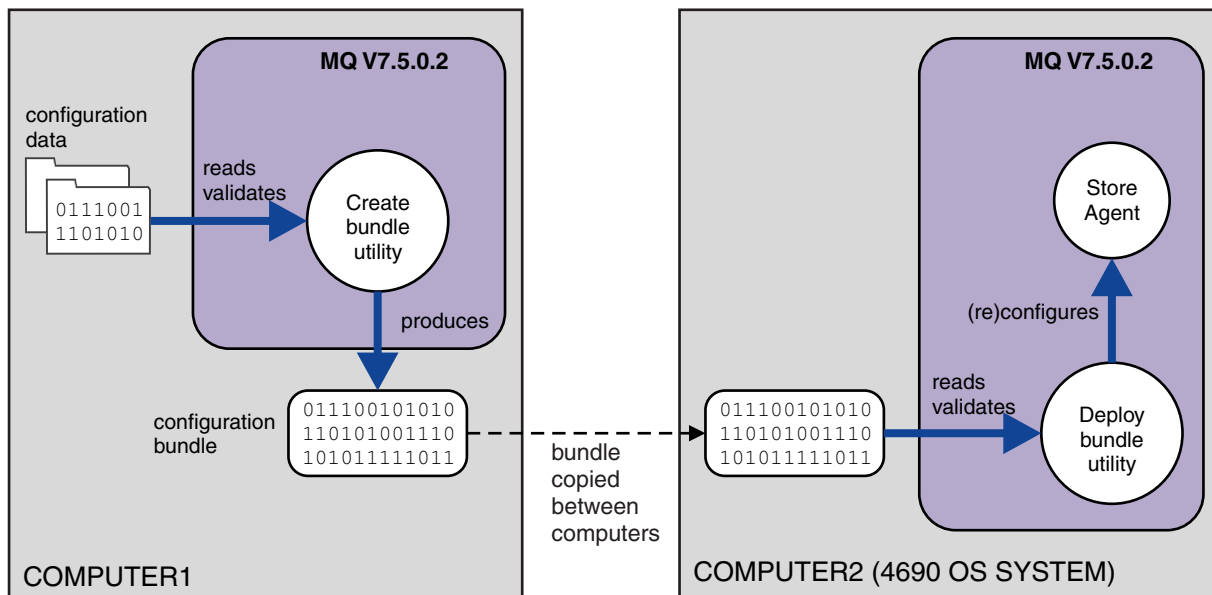
This diagram shows how the customer using this scenario creates and deploys a configuration bundle. To achieve this scenario, the following steps were completed:

1. A set of configuration data is created on COMPUTER1. For a retailer, this work typically takes place at one central site such as the head office. In this scenario, the configuration data is based on one of the examples that are provided as part of WebSphere MQ Managed File Transfer.

2. The **fteBundleConfiguration** utility is used on COMPUTER1 to read and validate the configuration data. As its output, the **fteBundleConfiguration** utility produces a single configuration bundle file.

3. The configuration bundle file is copied to the file system of COMPUTER2, which is a 4690 OS store controller system.
4. The **ftecfg** command is used to configure or reconfigure the WebSphere MQ Managed File Transfer installation on the 4690 OS system.

The task assumes that you have a Windows system. For a UNIX system, substitute appropriate paths and commands, and ensure that you have read and write access to all relevant directories. The scenario also assumes that you have a basic understanding of what a queue manager does.

As the user mqmAdmin, on COMPUTER1:

Procedure

1. Create a temporary directory to use when you manipulate the configuration data.
   ```
   mkdir %TEMP%\4690cfg
   ```
2. Unpack the sample configuration bundle, by using the **fteBundleConfiguration** command.
   ```
   fteBundleConfiguration -x MQ_INSTALL_PATH\mqft\samples\4690\basic.zip %TEMP%\4690cfg
   ```
3. Change directory to the temporary directory
   ```
   cd %TEMP%\4690cfg
   ```
4. Edit the `coordination.properties` file so that it contains the following property:
   ```
   coordinationQMgr=hoffQM
   ```
5. Rename the name directory to reflect the agent's name.
   ```
   move name SAGENT
   ```
6. Edit the `agent.properties` file (located inside the `agents\names` directory) so that it contains the following six properties:
   ```
   agentName=SAGENT
   agentQMgr=hoffQM
   agentQMgrHost=host or ip address of COMPUTER1
   agentQMgrPort=port number MQ is configured to listen on
   agentQMgrChannel=FTE.AGENT.SVRCONN
   authorityChecking=true
   ```
7. Create a configuration bundle by using the **fteBundleConfiguration** command.
   ```
   fteBundleConfiguration s1cfg.zip %TEMP%\4690cfg
   ```
8. Copy the configuration bundle to the 4690 OS system by using the mechanism that you normally use for transferring files to 4690 OS. Ensure that the file is written to the root of the `C:\` drive as `C:\S1CFG.ZIP`. The configuration bundle contains binary data. Ensure that the configuration bundle is transferred as a binary file if you are using the FTP protocol to transfer the data.
9. Optional: Remove the temporary directory that was used to manipulate the configuration data.
   ```
   %TEMP% rmdir /s 4690cfg
   ```

Related concepts:

"1. Get started with file transfers using a 4690 OS in store" on page 36
You can use this scenario to help you get started with WebSphere MQ Managed File Transfer on the 4690 OS.

"Using WebSphere MQ Managed File Transfer in a retail environment" on page 35
You can install WebSphere MQ Managed File Transfer on a device that is running an IBM 4690 operating system. To complete this type of installation, you must first install WebSphere MQ Managed File Transfer on a non-4690 machine. You then collect installation and configuration files on the non-4690 machine and transfer them to the IBM 4690 machine. You then use these files to install WebSphere MQ Managed File Transfer on the IBM 4690 machine.

Deploy an agent to a 4690 OS system

Implementing the solution that is described by this scenario requires the deployment of a WebSphere MQ Managed File Transfer agent on a 4690 OS. The agent is started by configuring it as a 4690 OS background application.

Procedure

Follow these instructions on COMPUTER2 from the WebSphere MQ Managed File Transfer bin directory `f:\adxetc\mft75\bin`.

1. Use the **ftecfg** command to create or replace the WebSphere MQ Managed File Transfer configuration on the 4690 OS system.

   `ftecfg C:\S1CFG.ZIP`

   For more information about the **ftecfg** command, see "ftecfg (creates a WebSphere MQ Managed File Transfer configuration on an IBM 4690 system)" on page 93. For more information about the configuration process, see "Creating a WebSphere MQ Managed File Transfer configuration on an IBM 4690 system" on page 68.

2. Create a 4690 OS background task to start the agent. For more information about starting an agent, see "Starting an agent on a 4690 OS system" on page 82.

3. Re-IPL the 4690 OS system to start the agent.

Verify the scenario by transferring a file

Verify the topology that is built in this scenario by transferring a file from the 4690 OS system (COMPUTER2) back to the host 4690 OS system. The task assumes that you have a Windows system. For a UNIX system, substitute appropriate paths and commands.

On COMPUTER2 (the 4690 OS system at the retail store):

1. Create a source file and a target directory:
   - Create a sample file to transfer, for example, `C:\START\MYFILE.TXT`
   - Create a directory, or identify an existing directory, to transfer this file to. For example, `C:\END\`

On COMPUTER1 (the system at head office):

1. As the user *ftuser*, enter the **fteCreateTransfer** command to initiate the transfer of your file from `C:\START\MYFILE.TXT` to `C:\END\MYFILE.TXT`:

   `fteCreateTransfer -sa SAGENT -sm hoffQM -sd delete -da SAGENT -dm hoffQM -w -dd C:\END\ C:\START\MYFILE.TXT`

   - `-sa SAGENT` defines the source agent (that is, the agent from which the file is transferred) to be SAGENT.
   - `-sm hoffQM` defines the queue manager to which the source agent, SAGENT, connects.
   - `-sd delete` specifies that the source file is deleted after a successful transfer.
   - `-da SAGENT` defines the destination agent (that is, the agent to which the file is transferred) to be SAGENT.
   - `-dm hoffQM` defines the queue manager to which the destination agent, SAGENT, connects.
   - `-w` requests the **fteCreateTransfer** command to wait for confirmation of its success.
   - `-dd C:\END\` defines the destination directory to be `C:\END\`.
   - `C:\START\`*MYFILE.TXT* defines the file to transfer.

On COMPUTER2 (the 4690 OS system at the retail store):

1. Check that the sample file is successfully moved between directories:
   - Check that the sample file is no longer present in the source directory, for example, `C:\START`
   - Check that the sample file is present in the destination directory, for example, `C:\END\MYFILE.TXT`

What to do next

- Extend your topology to include transfers between an agent at the head-office system and the retail store agent that is created in this scenario. For more information, see "2. Transferring files from head office to a 4690 OS system in store."

- Extend the security model that is used in this scenario, by reading about "Sandboxes" on page 103,"Authorities for resources specific to WebSphere MQ Managed File Transfer" on page 404, or "User authorities on WebSphere MQ Managed File Transfer actions" on page 409.

- Secure your environment further. Your own requirements might mandate a different access model to the one used in this scenario. For more information, see Securing WebSphere MQ File Transfer Edition V7, which can be found at: `http://www.ibm.com/developerworks/websphere/library/techarticles/0902_wyatt/0902_wyatt.html`.

- Creating a single configuration bundle that can be deployed to many 4690 OS systems by automatic substitution of agent name or agent configuration values. For more information, see "Customizing agent names in a 4690 OS configuration bundle" on page 69, and "Customizing agent properties in a 4690 OS configuration bundle" on page 71.

2. Transferring files from head office to a 4690 OS system in store

This scenario explains how you can use WebSphere MQ Managed File Transfer to send file data from a head office site to a 4690 OS store controller system in a retail store.

This scenario builds on the previous scenario, "1. Get started with file transfers using a 4690 OS in store" on page 36. Ensure that you have already completed the previous scenario, before you start this scenario.
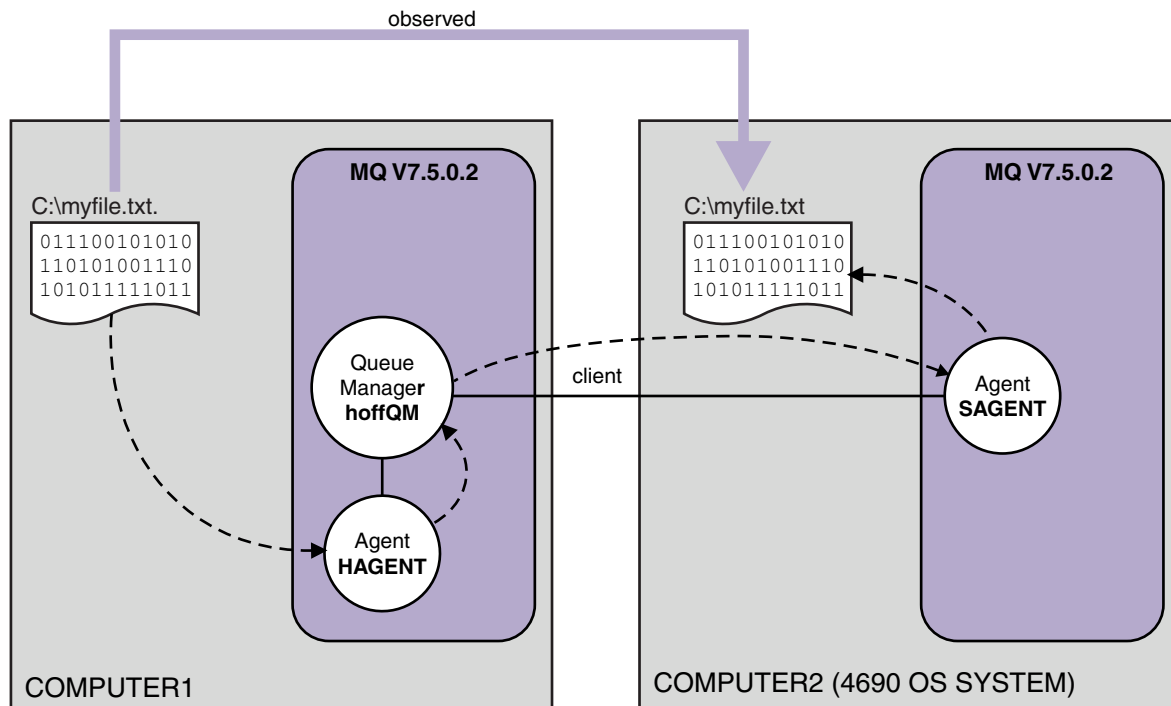
This diagram shows the head office and store topology that spans two computers. COMPUTER1 is the head-office computer where queue manager `hoffQM` was created in the previous scenario. COMPUTER2 is the 4690 OS store controller, which is typically in a retail store. The WebSphere MQ Managed File Transfer agent, SAGENT was created on COMPUTER2 in the previous scenario. This scenario creates agent HAGENT on COMPUTER1 and transfers a file from COMPUTER1 to COMPUTER2 as indicated on the diagram.

To work through this scenario, you must have a basic understanding of IBM WebSphere MQ and WebSphere MQ Managed File Transfer. Specifically, you must be familiar with the following concepts:

- Concept of a queue manager
- Concept of a IBM WebSphere MQ agent
- Basic configuration and administration of WebSphere MQ Managed File Transfer

For more information about the WebSphere MQ Managed File Transfer capability, see "WebSphere MQ Managed File Transfer introduction" on page 1.

Configuring file transfers at head office

This scenario extends the first scenario to include file transfers from head office to an in-store 4690 OS system.
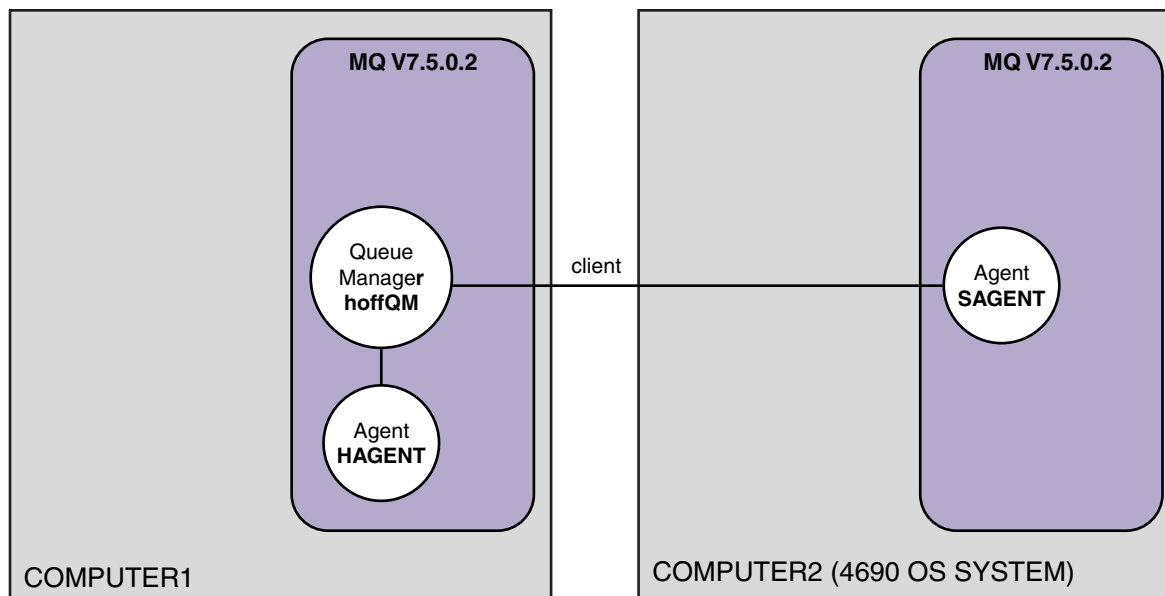
This diagram shows the topology that is created in this scenario. The queue manager `hoffQM` on COMPUTER1 and agent SAGENT on COMPUTER2 were created as part of the previous scenario. When you have completed this scenario, you can complete the following tasks:

- Define a WebSphere MQ Managed File Transfer agent on the system at head office
- Start the agent on the system at head office
- Update the authorities that members of the FTUSERS group have, so that only file transfers from the head-office site to the retail store site are permitted

The task assumes that you have a Windows system. For a UNIX system, substitute appropriate paths and commands.

Prerequisites

To complete this scenario, you need the following items:

- COMPUTER1, a computer that is deployed at the head-office site.
- COMPUTER2, a 4690 OS store controller that is deployed at the retail store site.

Both computers must have a working configuration that you created as part of "1. Get started with file transfers using a 4690 OS in store" on page 36.

Procedure

As the user mqmAdmin, from the IBM WebSphere MQ `bin` directory (*MQ_INSTALL_ROOT*\bin) on COMPUTER1, complete the following steps:

1. Create an MQMFT agent, called HAGENT.

   ```
   fteCreateAgent -agentName HAGENT -agentQMgr hoffQM -s -su fthoff -sp password for fthoff
   ```

   The agent is created so that it is started as a Windows service. This means that the agent continues to run, under the `fthoff` user account, even if the user that starts the agent logs off the system. For more information about running an agent as a Windows service, see "Starting an agent as a Windows service" on page 201. If you are using a UNIX system as COMPUTER1 (the head-office computer), see "Starting an agent at UNIX system startup" on page 203.

2. Edit the configuration for the agent HAGENT to enable checking of user authorities for file transfer actions. In the `agent.properties` file in `..\mqft\config\hoffQM\agents\HAGENT\agent.properties`, add the following entry:

   ```
   authorityChecking=true
   ```

   For more information, see "User authorities on WebSphere MQ Managed File Transfer actions" on page 409.

3. Use the MQSC interface to define the IBM WebSphere MQ objects that are required by agent HAGENT on queue manager hoffQM.

   ```
   runmqsc hoffQM < ..\mqft\config\hoffQM\agents\HAGENT\HAGENT_create.mqsc
   ```

4. Ensure that the FTHOFFS, FTSTORES, and FTUSER groups have appropriate access to the IBM WebSphere MQ objects that belong to agent HAGENT. You might need to tailor this configuration to suit your own security requirements.

   ```
   setmqaut -m hoffQM -t qmgr -g FTHOFFS +connect +inq +setid +altusr
   setmqaut -m hoffQM -n SYSTEM.FTE -t queue -g FTHOFFS +get +put
   setmqaut -m hoffQM -n SYSTEM.FTE.COMMAND.HAGENT -t queue -g FTHOFFS +browse +get +put +setid
   setmqaut -m hoffQM -n SYSTEM.FTE.DATA.HAGENT -t queue -g FTHOFFS +get +put
   setmqaut -m hoffQM -n SYSTEM.FTE.EVENT.HAGENT -t queue -g FTHOFFS +browse +get +put
   setmqaut -m hoffQM -n SYSTEM.FTE.REPLY.HAGENT -t queue -g FTHOFFS +browse +get +put
   setmqaut -m hoffQM -n SYSTEM.FTE.STATE.HAGENT -t queue -g FTHOFFS +browse +get +inq +put
   setmqaut -m hoffQM -n SYSTEM.FTE -t topic -g FTHOFFS +pub +sub
   setmqaut -m hoffQM -n SYSTEM.DEFAULT.MODEL.QUEUE -t queue -g FTHOFFS +browse +dsp +get +put
   setmqaut -m hoffQM -n SYSTEM.FTE.COMMAND.HAGENT -t queue -g FTSTORES +put
   setmqaut -m hoffQM -n SYSTEM.FTE.DATA.HAGENT -t queue -g FTSTORES +put
   setmqaut -m hoffQM -n SYSTEM.FTE.DATA.SAGENT -t queue -g FTHOFFS +put
   setmqaut -m hoffQM -n SYSTEM.FTE.REPLY.HAGENT -t queue -g FTSTORES +put
   setmqaut -m hoffQM -n SYSTEM.FTE.COMMAND.HAGENT -t queue -g FTUSERS +put
   ```

   The following commands must be run for each in-store agent.

   ```
   setmqaut -m hoffQM -n SYSTEM.FTE.COMMAND.SAGENT -t queue -g FTHOFFS +put
   setmqaut -m hoffQM -n SYSTEM.FTE.DATA.SAGENT -t queue -g FTHOFFS +put
   setmqaut -m hoffQM -n SYSTEM.FTE.REPLY.SAGENT -t queue -g FTHOFFS +put
   ```

5. Remove authorization for members of the FTUSERS group so that those members are not able to transfer files from the 4690 OS system back to itself.

   ```
   setmqaut -m hoffQM -n SYSTEM.FTE.AUTHTRN1.SAGENT -t queue -g FTUSERS -browse -put
   setmqaut -m hoffQM -n SYSTEM.FTE.AUTHAGT1.SAGENT -t queue -g FTSTORES -browse -put
   ```

6. Authorize members of the FTUSERS group to be able to transfer files from agent HAGENT to agent SAGENT.

   ```
   setmqaut -m hoffQM -n SYSTEM.FTE.AUTHTRN1.HAGENT -t queue -g FTUSERS +browse
   setmqaut -m hoffQM -n SYSTEM.FTE.AUTHAGT1.HAGENT -t queue -g FTSTORES +browse
   ```

   The following commands must be run for each in-store agent.

   ```
   setmqaut -m hoffQM -n SYSTEM.FTE.AUTHTRN1.SAGENT -t queue -g FTUSERS +put
   setmqaut -m hoffQM -n SYSTEM.FTE.AUTHAGT1.SAGENT -t queue -g FTHOFFS +put
   ```

7. Start the agent HAGENT.

```
fteStartAgent HAGENT
```

Related concepts:

"1. Get started with file transfers using a 4690 OS in store" on page 36
You can use this scenario to help you get started with WebSphere MQ Managed File Transfer on the 4690 OS.

"Using WebSphere MQ Managed File Transfer in a retail environment" on page 35
You can install WebSphere MQ Managed File Transfer on a device that is running an IBM 4690 operating system. To complete this type of installation, you must first install WebSphere MQ Managed File Transfer on a non-4690 machine. You then collect installation and configuration files on the non-4690 machine and transfer them to the IBM 4690 machine. You then use these files to install WebSphere MQ Managed File Transfer on the IBM 4690 machine.

Transferring a file by using the command line

You can use the command-line interfaces, provided with WebSphere MQ Managed File Transfer, to transfer a file from the head-office system to a 4690 OS store controller system in a retail store.

The task assumes that you have a Windows system. For a UNIX system, substitute appropriate paths and commands.

1. Create or identify a suitable file on the computer at head office that can be used for the transfer. For example, `C:\start\myfile.txt`.
2. Create or identify a suitable directory that the file can be copied into on the 4690 OS store controller. For example, `C:\END\`.
3. As user `ftuser` on COMPUTER1, enter the following **fteCreateTransfer** command to start the transfer of your file from `C:\start\myfile.txt` (on COMPUTER1) to `C:\END\MYFILE.TXT` (on COMPUTER2):

   ```
   fteCreateTransfer -sa HAGENT -sm hoffQM -da SAGENT -dm hoffQM -w -dd C:\END\
   C:\start\myfile.txt
   ```

   For more information about this command, see "**fteCreateTransfer** (create new file transfer)" on page 461.
4. Confirm that the file has been copied to the 4690 OS store controller (COMPUTER2).

Related concepts:

"1. Get started with file transfers using a 4690 OS in store" on page 36
You can use this scenario to help you get started with WebSphere MQ Managed File Transfer on the 4690 OS.

"Using WebSphere MQ Managed File Transfer in a retail environment" on page 35
You can install WebSphere MQ Managed File Transfer on a device that is running an IBM 4690 operating system. To complete this type of installation, you must first install WebSphere MQ Managed File Transfer on a non-4690 machine. You then collect installation and configuration files on the non-4690 machine and transfer them to the IBM 4690 machine. You then use these files to install WebSphere MQ Managed File Transfer on the IBM 4690 machine.

Transferring a file by using WebSphere MQ Explorer

You can use WebSphere MQ Explorer to transfer a file from the head-office system to a 4690 OS store controller system in a retail store.

The task assumes that you have a Windows system. For a UNIX system, substitute appropriate paths and commands.

1. Identify or create a source file and a destination directory.
   a. Identify or create a sample file on COMPUTER1 (the computer at head office). For example: `C:\start\myfile.txt`
   b. Identify or create a directory on COMPUTER2 (the 4690 OS store controller). For example: `C:\END\`

2. As user ftuser, start WebSphere MQ Explorer on COMPUTER1. Either start the program from the **Start** menu (or equivalent), or run the **strmqcfg** command. For more information, see Launching WebSphere MQ Explorer.

3. Expand **Managed File Transfer** in the WebSphere MQ Explorer Navigator, right-click hoffQM and select **New Transfer** to start the New Transfer Wizard.
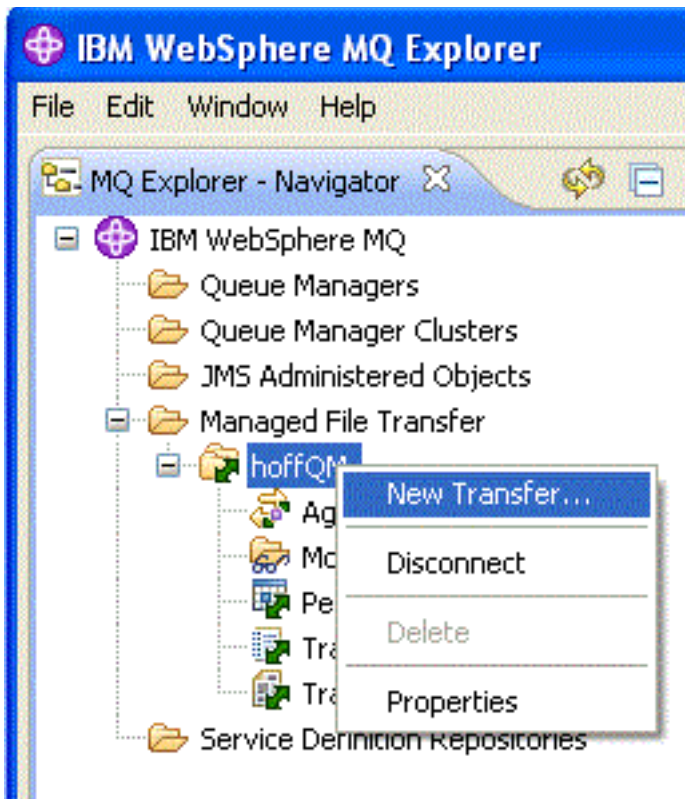
4. Select HAGENT as the source agent and SAGENT as the destination agent. Click **Next**.
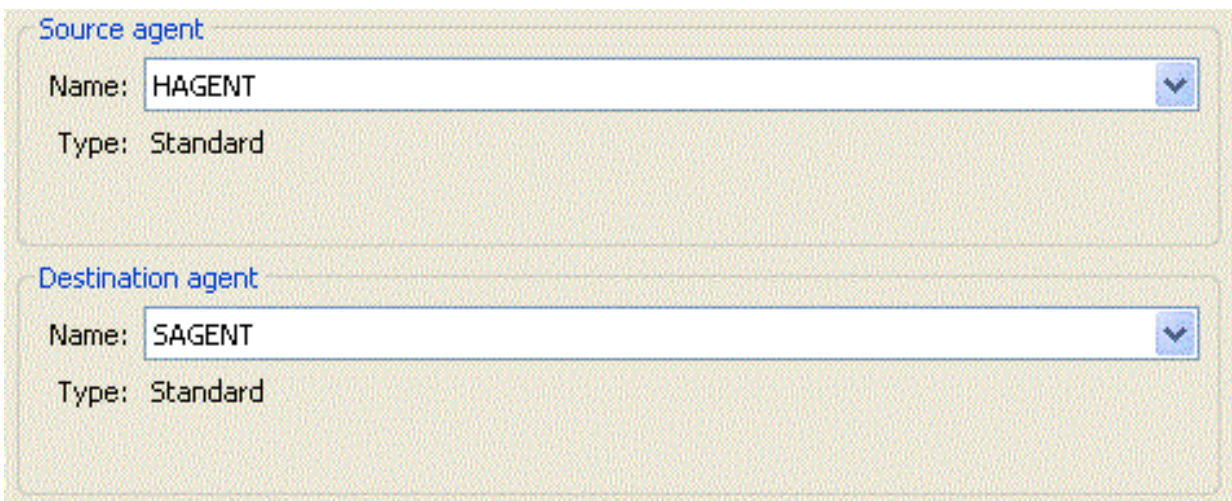
5. Click **Add**.
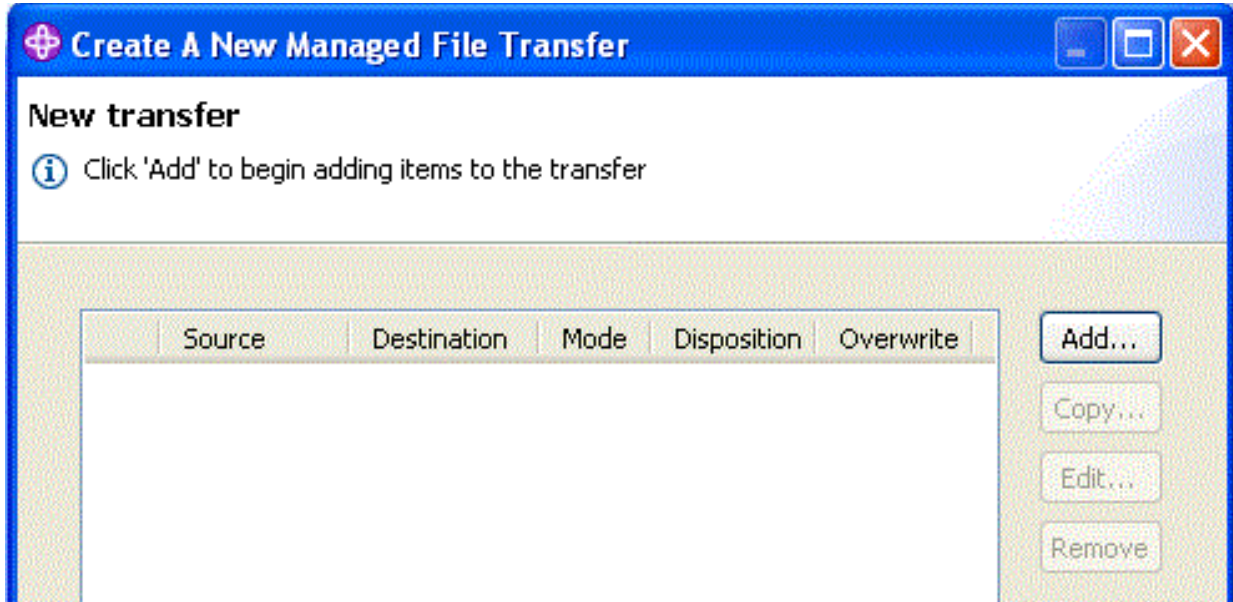
6. Complete the following steps on the Add a transfer item panel:

   a. For the source, enter `C:\start\myfile.txt` in the **File name** field.

   b. For the destination, select **Directory** from the **Type** list.

   c. For the destination, enter `C:\END\` in the **Directory** field. Click **OK**.
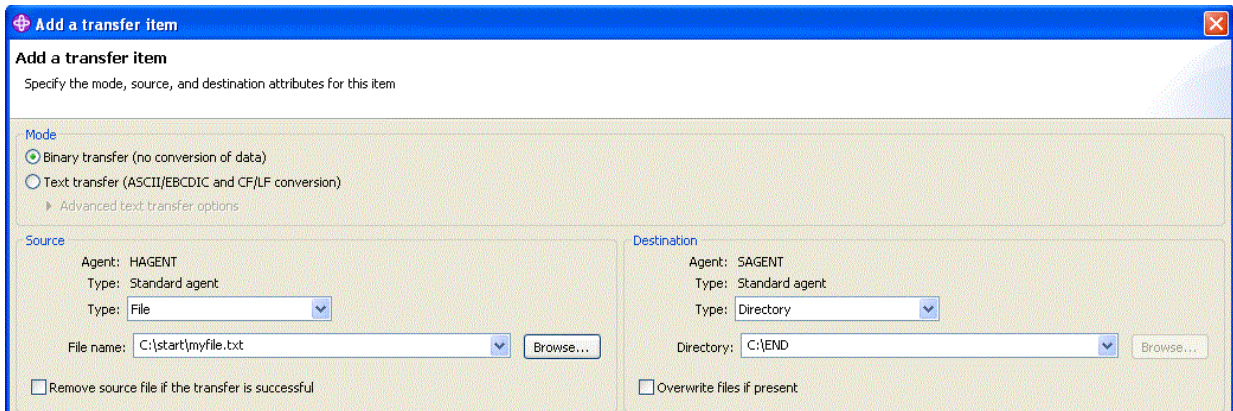
7. Click **Finish**. The transfer starts.

8. You can view the progress of the transfer in the Transfer Log window. This view is displayed by clicking **Transfer Log** in the Navigator.
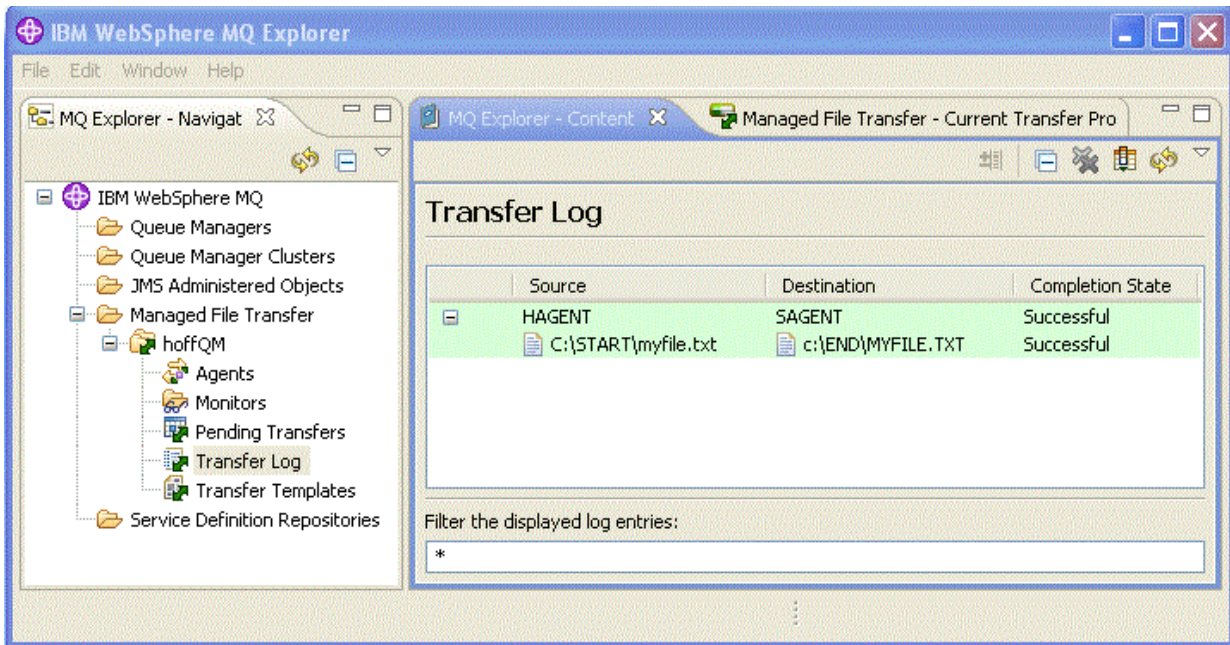
9. You can also manually check the file system of COMPUTER2 to confirm that the new file exists, for example: `C:\END\MYFILE.TXT`.

What to do next

You can extend your topology to automatically transfer files to the head-office system when they are created in a retail store. For more information, see "3. Transferring files from a 4690 OS system in store to head office."

You can secure your environment further. Your own requirements might mandate a different access model to the one used in this scenario. For more information about best practices in this area, see Securing WebSphere MQ File Transfer Edition V7.

Related concepts:

"1. Get started with file transfers using a 4690 OS in store" on page 36
You can use this scenario to help you get started with WebSphere MQ Managed File Transfer on the 4690 OS.

"Using WebSphere MQ Managed File Transfer in a retail environment" on page 35
You can install WebSphere MQ Managed File Transfer on a device that is running an IBM 4690 operating system. To complete this type of installation, you must first install WebSphere MQ Managed File Transfer on a non-4690 machine. You then collect installation and configuration files on the non-4690 machine and transfer them to the IBM 4690 machine. You then use these files to install WebSphere MQ Managed File Transfer on the IBM 4690 machine.

3. Transferring files from a 4690 OS system in store to head office

This scenario explains how to use WebSphere MQ Managed File Transfer to send file data from a 4690 operating system store controller that is situated in a retail store to a head-office site.

This scenario builds on the previous two scenarios. For more information, see "1. Get started with file transfers using a 4690 OS in store" on page 36 and "2. Transferring files from head office to a 4690 OS system in store" on page 47. These two previous scenarios must be completed before you start on this scenario.
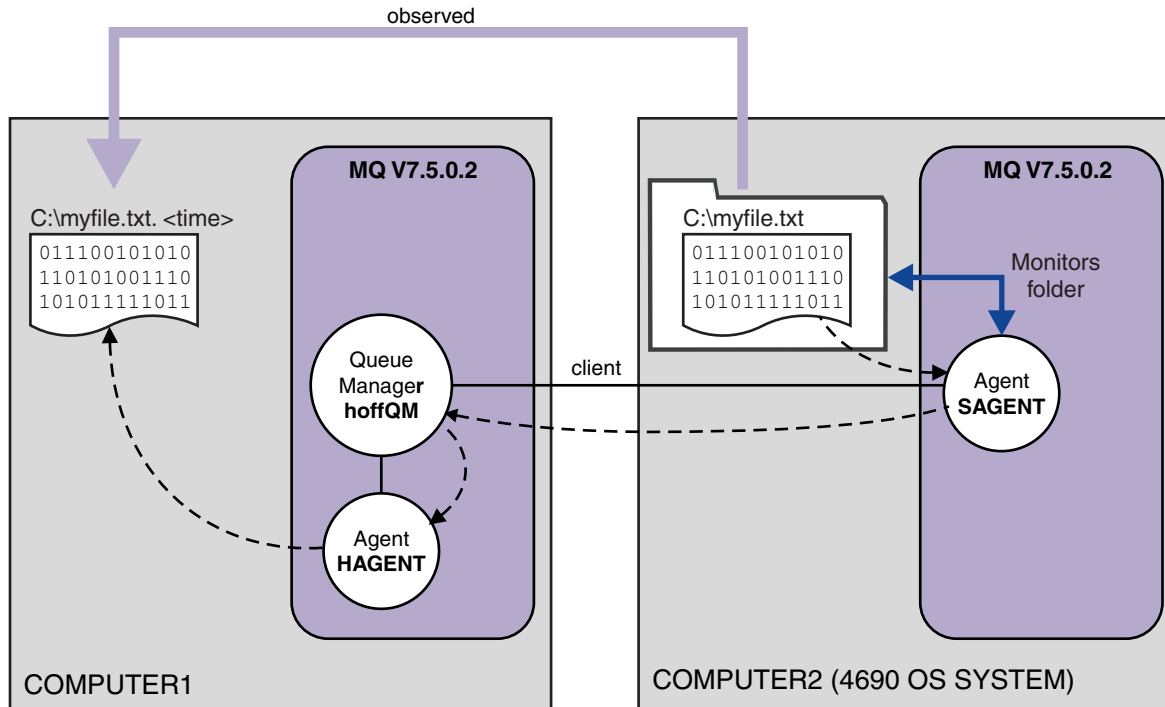
This diagram shows the file transfer operation that is configured as part of this scenario. The two agents, HAGENT and SAGENT, and the queue manager hoffQM, were configured as part of the earlier scenarios.

In this scenario, you configure agent SAGENT on COMPUTER2 to monitor a directory on the file system of the 4690 OS computer. When files arrive in this directory, they are transferred to a directory on COMPUTER1 and given a modified file name that includes a unique time stamp.

To complete this scenario, you need a basic understanding of IBM WebSphere MQ and WebSphere MQ Managed File Transfer. Specifically, basic configuration and administration of both IBM WebSphere MQ and Managed File Transfer, the concept of a queue manager, and the concept of a Managed File Transfer agent. For more information about Managed File Transfer capability, see "WebSphere MQ Managed File Transfer introduction" on page 1.

Configuring the agent on 4690 OS to monitor a directory

A common file transfer requirement is for WebSphere MQ Managed File Transfer to monitor a directory and transfer any files that are found to another system. Often the files must be given a unique name when they arrive at the destination system to avoid the possibility of a duplicate file name. Duplicate file names may cause for example, a file to be overwritten before it can be processed.

Before you begin

Both computers must have a working configuration that you created as part of "1. Get started with file transfers using a 4690 OS in store" on page 36.
- COMPUTER1, a computer that is deployed at head office.
- COMPUTER2, a 4690 OS store controller that is deployed at the retail store.

About this task

In this scenario, you complete the following tasks:
- Update the authorities of the FTUSERS group, so that members of that group can define resource monitors to agent SAGENT, and transfer files from SAGENT to HAGENT.

- Create the definitions that are required so that agent SAGENT can monitor a directory on the file system of COMPUTER2, the 4690 OS store controller. Files arriving in this directory are transferred to COMPUTER1, at head office.

The task assumes that you have a Windows system. For a UNIX system, substitute appropriate paths and commands.

All tasks and commands that are run against SAGENT must be run for each in-store agent.

Procedure

1. Identify or create source and target directories.
   a. Identify or create a directory on COMPUTER2, the 4690 OS store controller. For example, `C:\MONITOR\`
   b. Identify or create a directory on COMPUTER1, the computer at head office. For example, `C:\end\`
2. As user mqmAdmin, ensure that members of the FTUSERS group have the appropriate authority to define resource monitors on agent SAGENT.
   ```
   setmqaut -m hoffQM -n SYSTEM.FTE.AUTHMON1.SAGENT -t queue -g FTUSERS +browse
   ```
3. As user mqmAdmin, ensure that the groups FTHOFFS, FTSTORES, and FTUSERS have the appropriate authorizations so that files can be transferred from agent SAGENT to agent HAGENT.
   ```
   setmqaut -m hoffQM -n SYSTEM.FTE.AUTHTRN1.SAGENT -t queue -g FTUSERS +browse
   setmqaut -m hoffQM -n SYSTEM.FTE.AUTHTRN1.HAGENT -t queue -g FTUSERS +put
   setmqaut -m hoffQM -n SYSTEM.FTE.AUTHAGT1.SAGENT -t queue -g FTHOFFS +browse
   setmqaut -m hoffQM -n SYSTEM.FTE.AUTHAGT1.HAGENT -t queue -g FTSTORES +put
   ```
4. As user ftuser start WebSphere MQ Explorer on COMPUTER1. Start the program from the **Start** menu (or equivalent), or run the **strmqcfg** command. For more information, see Launching WebSphere MQ Explorer.
5. Click **Managed File Transfer** in the WebSphere MQ Explorer navigation view, right-click on Monitors under `hoffQM`, and select **New Monitor** to start the New Monitor wizard.
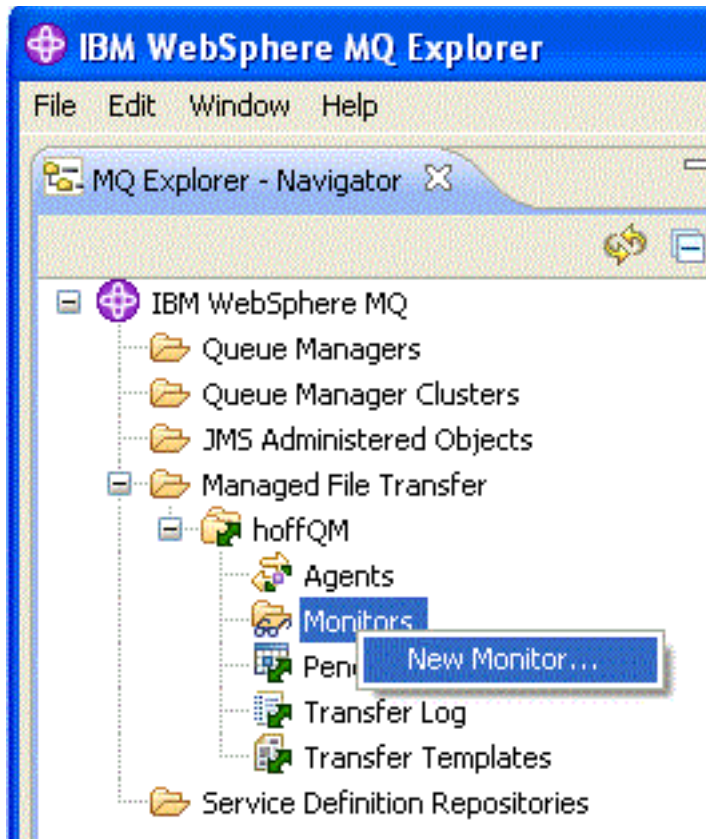
6. In the **Monitor name** field, enter a monitor name. Select SAGENT in the **Name** list in the **Source agent** pane. Select HAGENT in the **Name** list inside the **Destination agent** pane. Click **Next**.

7. Enter `C:\MONITOR\` in the **Directory** field. From the **Poll interval** list, select **10** and **seconds**. Click **Next**.

8. Click **Next** on the next page of the wizard to accept the default values for the Trigger condition.
9. Click **Add** to specify the files that to be transferred.
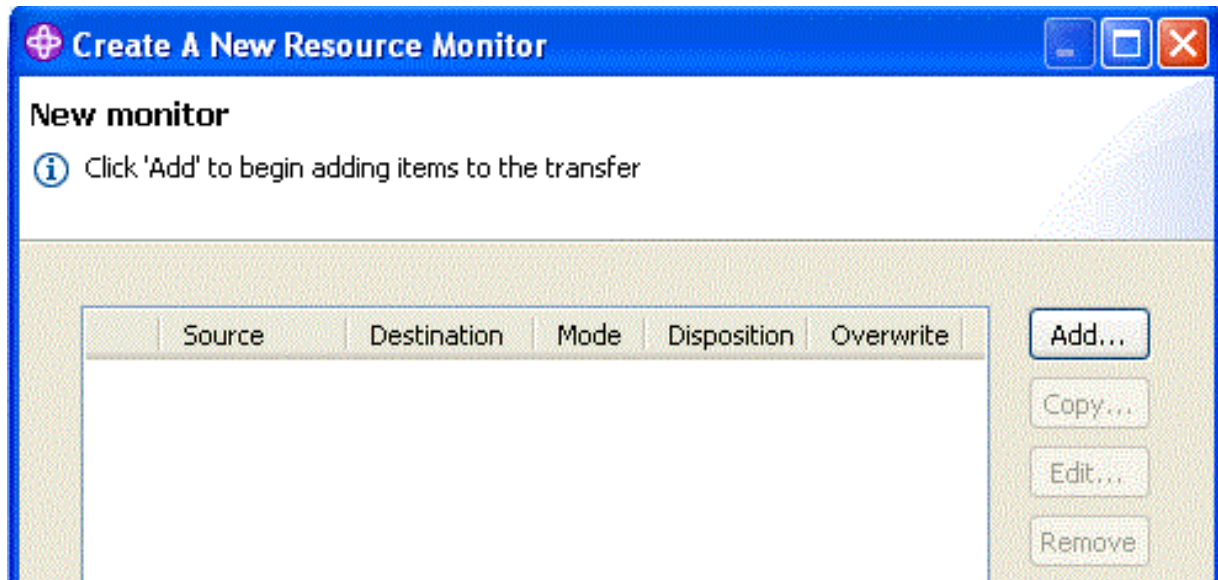
10. Enter `${FilePath}` in the **File name** field inside the **Source** pane. Enter `C:\end\`
`${FileName}.${CurrentTimeStampUTC}` in the **File name** field inside the **Destination** pane. These
values include variables that are substituted, at the point the transfer takes place, for information
such as the file name matched by the resource monitor. For more information about variable
substitution, see: "Customizing tasks with variable substitution" on page 225. Click **OK** to complete
the dialog.
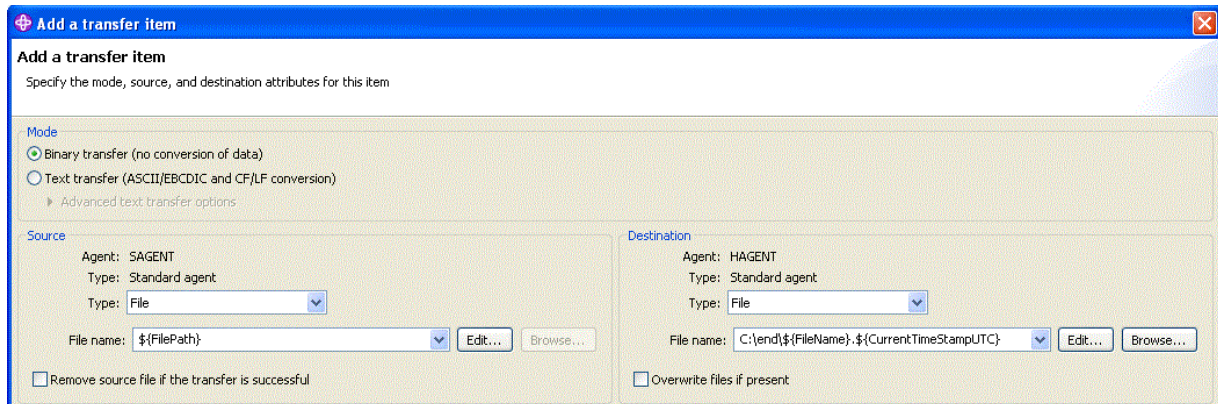
11. Click **Finish** to complete the definition of the resource monitor.
12. To inspect the resource monitors, that are defined to WebSphere MQ Managed File Transfer, click
Managed File Transfer in the WebSphere MQ Explorer navigation view, then right-click on Monitors
under hoffQM.
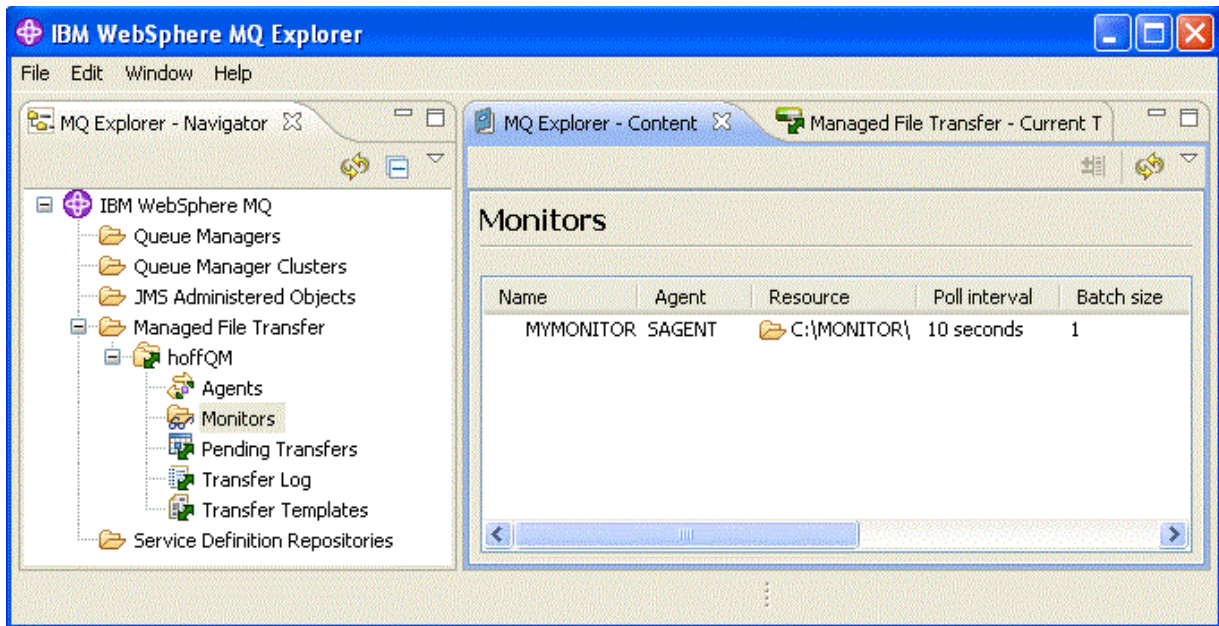
Related concepts:

"1. Get started with file transfers using a 4690 OS in store" on page 36
You can use this scenario to help you get started with WebSphere MQ Managed File Transfer on the 4690 OS.

"2. Transferring files from head office to a 4690 OS system in store" on page 47
This scenario explains how you can use WebSphere MQ Managed File Transfer to send file data from a head office site to a 4690 OS store controller system in a retail store.

"3. Transferring files from a 4690 OS system in store to head office" on page 53
This scenario explains how to use WebSphere MQ Managed File Transfer to send file data from a 4690 operating system store controller that is situated in a retail store to a head-office site.

Verify the scenario by transferring a file from 4690 OS to head office

Use this scenario to demonstrate using a WebSphere MQ Managed File Transfer resource monitor to transfer a file from the 4690 OS store controller in a retail store to a computer at a head-office site.

In the previous topic, "Configuring the agent on 4690 OS to monitor a directory" on page 54, you configured the retail store agent to monitor a directory. When files are placed into this directory, they are transferred to the head-office computer. These steps verify that your topology is working correctly by creating a file in the monitored directory, and checking that it is transferred to the head-office computer.

1. Create a file, for example `C:\MONITOR\MYFILE.TXT` in the directory that is being monitored on COMPUTER2.

   ```
   echo A big hello to everyone at head-office > C:\MONITOR\MYFILE.TXT
   ```

   The agent on the 4690 OS system (SAGENT) automatically transfers this file to the head-office computer.
2. To view information about the file transfer operation, started by creating a file in the monitored directory, click **Managed File Transfer** in the WebSphere MQ Explorer navigation view, expand hoffQM, and select **Transfer Log**.
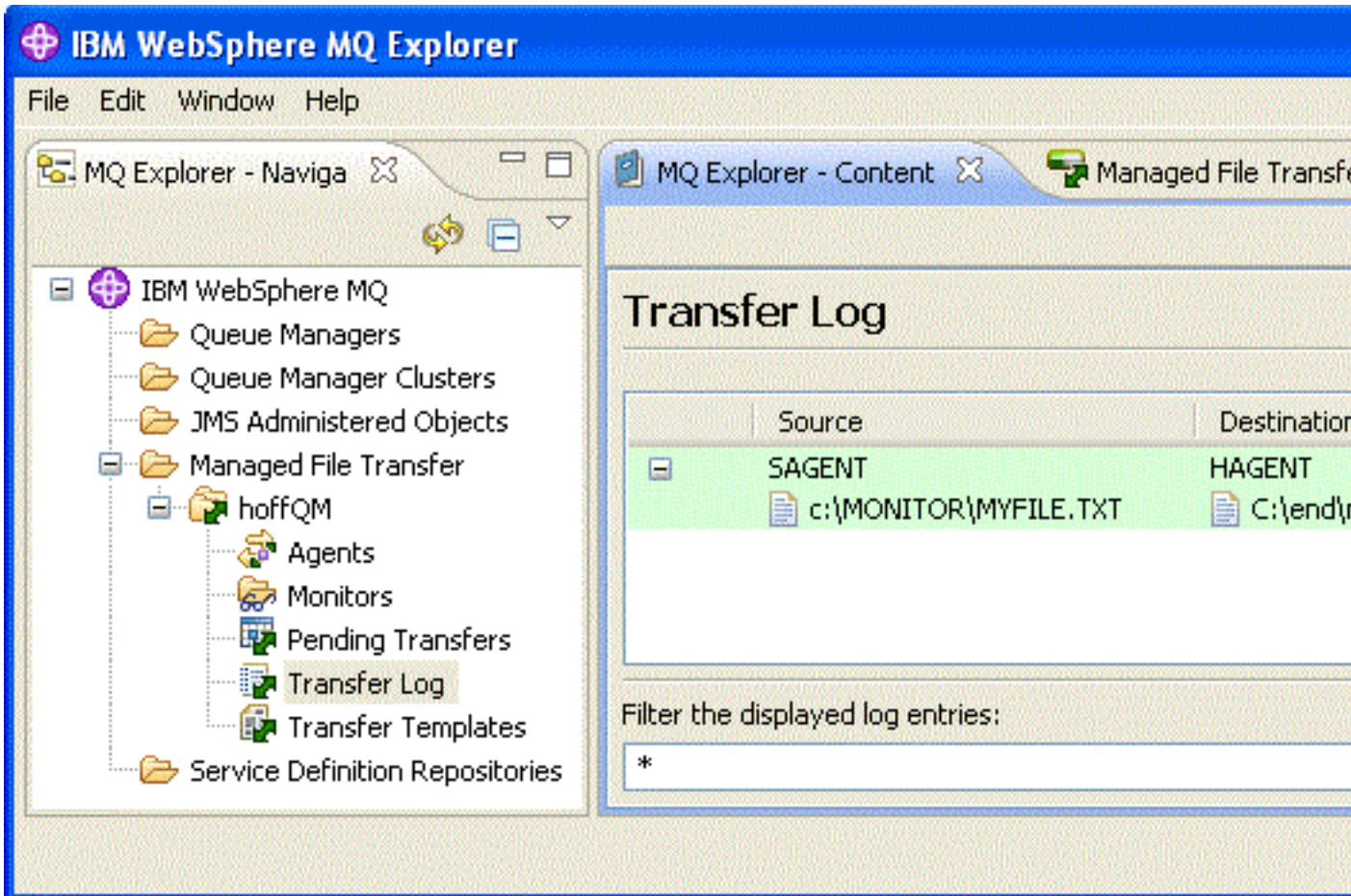
3. You can also inspect the file system of COMPUTER1 manually to confirm that the new file exists, for example: `C:\end\myfile.txt.20130307103836255`

What to do next

- Learn about creating resource monitor definitions from the command line, see "Resource monitoring" on page 215.
- Find out how WebSphere MQ Managed File Transfer can log managed file transfer activities to a database or a file system, see "Configuring a WebSphere MQ Managed File Transfer logger" on page 133.
- Read about the more general capabilities of WebSphere MQ Managed File Transfer, see "WebSphere MQ Managed File Transfer introduction" on page 1.

Preparing to install WebSphere MQ Managed File Transfer on an IBM 4690 system

You can install WebSphere MQ Managed File Transfer on a device that is running an IBM 4690 operating system. You complete this type of installation in two stages. The first stage is done on a non-4690 machine and involves collecting the configuration files, which are needed after installation, and transferring the configuration files and the installation `.zip` file to the IBM 4690 machine. The second stage uses the installation `.zip` file to install WebSphere MQ Managed File Transfer on the IBM 4690 machine.

Stage 1. Collect the installation and configuration files

Complete this stage on a machine that is not running an IBM 4690 operating system. Collect the files necessary for the installation and configuration and transfer them to the IBM 4690 machine.

Complete the following steps:

1. Create a .zip file that contains the configuration you want to use as part of your installation, for example config.zip. For more information about creating this configuration, see: "Creating a WebSphere MQ Managed File Transfer configuration on an IBM 4690 system" on page 68.

2. Ensure you have the installation .zip file: MQMFT75.ZIP.

   The MQMFT75.ZIP installation file for WebSphere MQ Managed File Transfer V7.5.0.2 is supplied either on a separate DVD specifically for 4690 OS or as a separate eImage, specifically for 4690 OS. You can download the eImage from the Passport Advantage® website at http://www.ibm.com/software/howtobuy/passportadvantage/pao_customers.htm.

3. By using FTP, or another method, transfer the MQMFT75.ZIP file and your config.zip file to the root directory of the IBM 4690 system f: drive.

Stage 2. Install WebSphere MQ Managed File Transfer

Complete this stage on the IBM 4690 system using the steps in "Installing IBM WebSphere MQ Managed File Transfer on 4690 OS."

Related concepts:

"Configuring WebSphere MQ Managed File Transfer in a master-backup 4690 OS controller setup" on page 80
You can configure agents in a master-backup 4690 OS controller setup to provide fault tolerance. Agents with the same name can be configured to run on multiple controllers in a retail environment. However, only one of the agents can run at any one time. This configuration can be combined with applications that are configured to stop and start in different circumstances.

"Configuring multiple WebSphere MQ Managed File Transfer agents in a 4690 OS controller setup" on page 81
You can configure multiple agents in a single store controller environment, or in an environment where there are multiple store controllers for a store.

Installing IBM WebSphere MQ Managed File Transfer on 4690 OS

Use the MQMFT75.ZIP file to install WebSphere MQ Managed File Transfer on 4690 OS.

Before you begin

- Ensure that you have created a WebSphere MQ Managed File Transfer configuration on a non-4690 system that you can use after installing MQMFT. For more information, see "Creating a WebSphere MQ Managed File Transfer configuration on an IBM 4690 system" on page 68

- The MQMFT75.ZIP installation file for WebSphere MQ Managed File Transfer V7.5.0.2 is supplied either on a separate DVD specifically for 4690 OS or as a separate eImage, specifically for 4690 OS. You can download the eImage from the Passport Advantage website at http://www.ibm.com/software/howtobuy/passportadvantage/pao_customers.htm.

About this task

Procedure

1. Transfer the installation .zip file MQMFT75.ZIP to the root directory of the F: drive on the 4690 OS store controller.

2. Log on and go to **Command Mode** (option **7** on the **SYSTEM MAIN MENU**).

3. Check whether WebSphere MQ Managed File Transfer is already installed. Request a directory listing of the F: drive to see whether the directory structure, created by the installation process, exists. Enter the following command:

```
dir f:\adxetc\mft75
```

If the response is similar to the following output, MQMFT is not installed:

```
Volume in drive vf: has no label
Directory of vf:adxetc/


        0 Files    6753900 KB free
```

If the response is similar to the following output, MQMFT is installed:

```
Volume in drive vf: has no label
Directory of vf:adxetc/mft75/
11-06-2012  11:00a  <DIR>     .
11-06-2012  11:00a  <DIR>     bin
11-06-2012  10:58a  <DIR>     ..
11-06-2012  11:00a  <DIR>     mqft
        4 Files    6715292 KB free
```

If the product is already installed, no further installation steps are required.

4. Switch to the root of the F: drive by entering the following commands:

```
f:
cd \
```

5. Extract the product installation files by running the following command:

```
adxnszzl –xo f:\MQMFT75.ZIP
```

A successful extraction of the product files produces output similar to the following:

```
ADXNSZZL - Version 3.3.0 - May 19 2011 17:44:03
  tool code Copyright (c) 2003,2011 IBM - All Rights Reserved
  zlib code Copyright (c) 1995-2002 Jean-loup Gailly and Mark Adler

Archive: f:/MQMFT75.ZIP
Extracting: f:/adxetc/mft75/bin (0 bytes)...Done
Inflating : f:/adxetc/mft75/bin/fteRAS.bat (974 bytes)...Done
Inflating : f:/adxetc/mft75/bin/ftecfg.bat (993 bytes)...Done
Inflating : f:/adxetc/mft75/bin/ftediag.bat (988 bytes)...Done
Extracting: f:/adxetc/mft75/mqft (0 bytes)...Done
Extracting: f:/adxetc/mft75/mqft/lib (0 bytes)...Done
Inflating : f:/adxetc/mft75/mqft/lib/agenttype.properties (32 bytes)...Done
Inflating : f:/adxetc/mft75/mqft/lib/com.ibm.wmqfte.agent.jar (1682543 bytes)...Done
Inflating : f:/adxetc/mft75/mqft/lib/com.ibm.wmqfte.bootstrap.jar (33376 bytes)...Done
Inflating : f:/adxetc/mft75/mqft/lib/com.ibm.wmqfte.cmdline.jar (1556790 bytes)...Done
Inflating : f:/adxetc/mft75/mqft/lib/com.ibm.wmqfte.common.jar (5371185 bytes)...Done
Inflating : f:/adxetc/mft75/mqft/lib/com.ibm.wmqfte.daemon.jar (75261 bytes)...Done
Inflating : f:/adxetc/mft75/mqft/lib/com.ibm.wmqfte.embedded.agent.jar (190744 bytes)...Done
Inflating : f:/adxetc/mft75/mqft/lib/com.ibm.wmqfte.exitroutines.api.jar (241582 bytes)...Done
Inflating : f:/adxetc/mft75/mqft/lib/com.ibm.wmqfte.native.jni.jar (3466 bytes)...Done
Inflating : f:/adxetc/mft75/mqft/lib/commons-beanutils.jar (188671 bytes)...Done
Inflating : f:/adxetc/mft75/mqft/lib/commons-digester-1.8.jar (143602 bytes)...Done
Inflating : f:/adxetc/mft75/mqft/lib/commons-io-1.4.jar (109043 bytes)...Done
Inflating : f:/adxetc/mft75/mqft/lib/commons-lang-2.4.jar (261809 bytes)...Done
Inflating : f:/adxetc/mft75/mqft/lib/commons-logging-1.1.1.jar (60841 bytes)...Done
Inflating : f:/adxetc/mft75/mqft/lib/commons-net-2.0.jar (197316 bytes)...Done
Inflating : f:/adxetc/mft75/mqft/lib/logging.properties (802 bytes)...Done
Inflating : f:/adxetc/mft75/mqft/lib/template.pc (374 bytes)...Done
Inflating : f:/adxetc/mft75/mqft/lib/template.rsp (452 bytes)...Done
Inflating : f:/adxetc/mft75/mqft/lib/com.ibm.mq.headers.jar (269365 bytes)...Done
Inflating : f:/adxetc/mft75/mqft/lib/com.ibm.mq.jar (429548 bytes)...Done
Inflating : f:/adxetc/mft75/mqft/lib/com.ibm.mq.jmqi.jar (2754010 bytes)...Done
Inflating : f:/adxetc/mft75/mqft/lib/com.ibm.mqjms.jar (3053199 bytes)...Done
```

An unsuccessful extraction of product files produces output that has a last line similar to the following:

```
ADXNSZZL — Exiting with rc=0x1F
```

If the extraction is not successful, use the following steps to troubleshoot the problem:

a. Ensure that the command that you used to extract the product files has been entered correctly.

b. Check that the MQMFT75.ZIP file has been correctly copied to F:\ and that the file name is entirely in uppercase letters.

c. Copy the MQMFT75.ZIP file to the 4690 OS system again. If you used the FTP protocol to transfer this file, ensure that it is transferred as a binary file.

d. If none of these steps resolve the problem, contact the IBM support center.

6. Validate that the product files were successfully extracted by running the following command:

```
dir f:\adxetc\mft75
```

If the response is similar to the following output, the MQMFT agent is installed:

```
Volume in drive vf: has no label
Directory of vf:adxetc/mft75/
11-06-2012  11:00a  <DIR>     .
11-06-2012  11:00a  <DIR>     bin
11-06-2012  10:58a  <DIR>     ..
11-06-2012  11:00a  <DIR>     mqft
        4 Files    6715292 KB free
```

If the response is similar to the following output, the MQMFT agent is not installed.

```
Volume in drive vf: has no label
Directory of vf:adxetc
        0 Files    6753900 KB free
```

If the product does not appear to be installed, carefully try each of the preceding steps again. If this repeating these steps does not resolve the problem, contact the IBM support center.

7. Read the product license and choose to accept or reject the license. Run the following commands:

```
cd \adxetc\mft75\bin
ftelap
```

This command displays the license for MQMFT and prompts you to either accept or decline the terms of the license. For more information about the **ftelap** command, see "ftelap (accept the license agreement during WebSphere MQ Managed File Transfer installation)" on page 92.

If you choose not to accept the license, complete the steps for removing the MQMFT program files from the 4690 OS system at "Uninstalling WebSphere MQ Managed File Transfer from a 4690 system" on page 67.

8. Check that the product installed correctly. When the installation is complete, the product is installed in the f:\adxetc\mft75 directory.

What to do next

When you have completed the installation, you must supply 4690 OS with a configuration. For more information, see "ftecfg (creates a WebSphere MQ Managed File Transfer configuration on an IBM 4690 system)" on page 93.

Installing a fix pack for IBM WebSphere MQ Managed File Transfer on 4690 OS

Use the *FIXPACK*.ZIP file to apply a fix pack to a WebSphere MQ Managed File Transfer installation on 4690 OS.

Before you begin

- Ensure that WebSphere MQ Managed File Transfer is installed and configured on the 4690 OS system that you are applying the fix pack to.

About this task

Procedure

1. Transfer the fix pack `.zip` file *FIXPACK*.ZIP to the root directory of the `f:` drive on the 4690 OS store controller.
2. Stop any process controller and agent processes that are running. Complete this task by accessing the background application menu, stopping all the process controller applications, then, stopping all the agent applications. Stop the process controller applications before the agent applications because they restart the agent applications. Process controller applications can be identified by parameter lists that start with: `@f:/adxetc/mft75/` and end with a `.pc` file suffix. Agent applications can be identified by parameter lists that start with: `@f:/adxetc/mft75/` and end with a `.rsp` file suffix.
3. Create a backup of the current installation. This backup can be used to roll back the application of the fix pack if a problem is encountered. Enter the following command:

   ```
   dir f:
   cd \adxetc
   adxnszzl -r -c MFTBACKUP.ZIP mft75\*
   ```
4. Apply the fix pack by unpacking the contents of *FIXPACK*.ZIP. Enter the following command:

   ```
   f:
   cd \
   adxnszzl -xo FIXPACK.ZIP
   ```
5. IPL the store controller.
6. Validate that the agent started correctly by checking the agent's log files. Enter the following command:

   ```
   f:
   cd \adxetc\mft75\mqft\logs\coord_qm\agents\agent_name\logs\
   type output0.log
   ```
7.

   Note: If it is necessary to roll back the fix pack application. Complete the following tasks:

   a. Stop any process controller and agent processes that are running. As in step 2.

   b. Unpack the backup that is created in step 3. Enter the following command:

   ```
   f:
   cd \adxetc
   adxnszzl -xo MFTBACKUP.ZIP
   ```

   c. IPL the store controller.

Related tasks:

"Installing IBM WebSphere MQ Managed File Transfer on 4690 OS" on page 62
Use the `MQMFT75.ZIP` file to install WebSphere MQ Managed File Transfer on 4690 OS.

"Creating a WebSphere MQ Managed File Transfer configuration on an IBM 4690 system" on page 68
To create or replace a WebSphere MQ Managed File Transfer configuration on an IBM 4690 system, you first create the configuration on a non-4690 platform. You then copy the configuration to the IBM 4690 system in a compressed file and run the **ftecfg** command to deploy the configuration to the IBM 4690 system.

Example installation script for IBM 4690 on WebSphere MQ Managed File Transfer

If you want to install WebSphere MQ Managed File Transfer on multiple 4690 systems, you can write an installation script to help. You can use the following example as a starting point to write your own installation script.

```
ECHO OFF


ECHO +--------------------------------------------------------------------+
ECHO +          IBM WebSphere MQ Managed File Transfer V7.5 Installation        +
ECHO +--------------------------------------------------------------------+
ECHO +--------------------------------------------------------------------+ > F:/install.log
ECHO +          IBM WebSphere MQ Managed File Transfer V7.5 Installation        + >> F:/install.log
ECHO +--------------------------------------------------------------------+ >> F:/install.log


REM Extract the contents of the product zip file into the F:/adxetc/mft75
REM directory.
ECHO + Installing MQMFT...                                                 +
ECHO + Installing MQMFT...                                                 + >> F:/install.log
adxnszzl -xo F:/MQMFT75.ZIP >> F:/install.log >>* F:/install.log
IF NOT ERRORLEVEL 0 GOTO EXTRACTFAIL


REM If you want to automatically accept the product license as part of
REM your installation, you can do so by using the supplied ftelap tool.
REM ********************************************************************
REM * NOTE: Uncomment the following line only after you have read the *
REM *       product license and have fully accepted its terms.  *
REM ********************************************************************
REM COMMAND -C F:/adxetc/mft75/bin/ftelap -accept >> F:/install.log >>* F:/install.log


REM After the product files are extracted and the license is accepted, the
REM ftecfg command can be run to lay down a configuration and generate
REM all the files necessary to start an agent.
COMMAND -C F:/adxetc/mft75/bin/ftecfg F:/MFT75CFG.zip >> F:/install.log >>* F:/install.log
IF NOT ERRORLEVEL 0 GOTO CFGFAIL


REM Copy the install log file to the product directory
COPY F:/install.log F:/adxetc/mft75/install.log
DEL F:/install.log

ECHO +--------------------------------------------------------------------+
ECHO + Installation complete.                                             +
ECHO + Product files can be found in the f:/adxetc/mft75 directory        +
ECHO + Installation log written to file: f:/adxetc/mft75/install.log      +
ECHO +--------------------------------------------------------------------+
ECHO +--------------------------------------------------------------------+ >> F:/adxetc/mft75/install.log
ECHO + Installation complete.                                             + >> F:/adxetc/mft75/install.log
ECHO + Product files can be found in the f:/adxetc/mft75 directory        + >> F:/adxetc/mft75/install.log
ECHO + Installation log written to file: f:/adxetc/mft75/install.log      + >> F:/adxetc/mft75/install.log
ECHO +--------------------------------------------------------------------+ >> F:/adxetc/mft75/install.log
```

```
       GOTO END


REM Log that the extract of the product files failed
:EXTRACTFAIL
ECHO + ERROR: Problem occurred extracting install files. See previous entries +
ECHO +          in the install log for more details.                          +
ECHO +----------------------------------------------------------------------+
ECHO + ERROR: Problem occurred extracting install files. See previous entries + >> F:/fteinst.log
ECHO +          in the install log for more details.                          + >> F:/fteinst.log
ECHO +----------------------------------------------------------------------+ >> F:/fteinst.log
GOTO FAIL


REM Log that the configuration command failed
:CFGFAIL
ECHO + ERROR: Problem occurred processing the supplied configuration zip     +
ECHO +          file. See previous entries in the install log for more details. +
ECHO +----------------------------------------------------------------------+
ECHO + ERROR: Problem occurred processing the supplied configuration zip     + >> F:/install.log
ECHO +          file. See previous entries in the install log for more details. + >> F:/install.log
ECHO +----------------------------------------------------------------------+ >> F:/install.log
GOTO FAIL


:FAIL
ECHO +----------------------------------------------------------------------+
ECHO + Installation failed, exiting.                                        +
ECHO + Installation log written to file: f:/install.log                    +
ECHO +----------------------------------------------------------------------+
ECHO +----------------------------------------------------------------------+ >> F:/install.log
ECHO + Installation failed, exiting.                                        + >> F:/install.log
ECHO + Installation log written to file: f:/install.log                    + >> F:/install.log
ECHO +----------------------------------------------------------------------+ >> F:/install.log
GOTO END


:END
```

Related tasks:

"Installing IBM WebSphere MQ Managed File Transfer on 4690 OS" on page 62
Use the `MQMFT75.ZIP` file to install WebSphere MQ Managed File Transfer on 4690 OS.

Uninstalling WebSphere MQ Managed File Transfer from a 4690 system

To uninstall the WebSphere MQ Managed File Transfer agent from a 4690 store controller, complete the following steps:

Procedure

1. Remove any existing WebSphere MQ Managed File Transfer background application definitions. Navigate to the DEFINE BACKGROUND APPLICATION screen and identify these definitions by searching for parameter lists that are prefixed with the following:

   `@f:\adxetc\mft75\`

2. Re-IPL the store controller.

3. Run the **uninstall** command to remove the WebSphere MQ Managed File Transfer product files. Use either the **-a** parameter or the **-c** parameter with the command.

   -a   Uninstalls all product files, configuration files, and log files

   -c   Uninstalls all product files, but retains configuration files and log files

For example, to uninstall all product files, configuration files, and log files, enter the following commands:

```
f:
cd \adxetc\mft75
uninstall -a
```

For more information, see "uninstall (uninstall WebSphere MQ Managed File Transfer from an IBM 4690 system)" on page 100.

4. Review the uninstall*n*.log file (where *n* is a number starting from zero) to ensure that the uninstallation completed with no errors . For example, uninstall0.log. This file is located at f:\adxetc\mft75.

5. Remove the uninstaller file and its log file. For example:

```
del uninstall.bat
del uninstall0.log
```

6. Optional: If you specified uninstall –a to remove all product, configuration, and log files, you can also remove the mft75 directory. For example:

```
f:
cd \adxetc
rmdir mft75
```

Creating a WebSphere MQ Managed File Transfer configuration on an IBM 4690 system

To create or replace a WebSphere MQ Managed File Transfer configuration on an IBM 4690 system, you first create the configuration on a non-4690 platform. You then copy the configuration to the IBM 4690 system in a compressed file and run the **ftecfg** command to deploy the configuration to the IBM 4690 system.

Procedure

To create or replace a WebSphere MQ Managed File Transfer configuration on an IBM 4690 system, complete the following steps:

1. Create the configuration on a non-4690 platform by running the **fteSetupCoordination**, **fteSetupCommands**, and **fteCreateAgent** commands. You can define only one coordination queue manager on an IBM 4690 system. You can configure more than one agent, but the agent name can only be a maximum of 23 characters. The 4690 system always makes client connections to the queue managers, so you must create the WebSphere MQ Managed File Transfer components by specifying client connections.

2. Complete the configuration setup by running the MQSC scripts that are generated by the **fteSetupCoordination** and **fteCreateAgent** commands. If you are using variable substitution, run the agent MQSC scripts that have been generated using the **fteDefine** command.

3. Create a config.zip file, containing the coordination.properties file and agents directories that are located under the MQ_DATA_PATH/mqft/config directory, by using the **fteBundleConfiguration** command. See "fteBundleConfiguration (create a WebSphere MQ Managed File Transfer IBM 4690 agent configuration .zip file)" on page 90 for details on using the command. See "Structure of the IBM 4690 WebSphere MQ Managed File Transfer configuration compressed file" on page 69 for details of the required file structure.

4. Copy the config.zip file to the IBM 4690 system.

5. On the IBM 4690 system, run the **ftecfg** command, passing the path to the config.zip file as a parameter. For example:

```
ftecfg f:\config.zip
```

If there is an existing configuration, it is backed up and then deleted by the **ftecfg** command. For more information, see ftecfg.

Structure of the IBM 4690 WebSphere MQ Managed File Transfer configuration compressed file

You create a WebSphere MQ Managed File Transfer configuration on an IBM 4690 system by passing, as a parameter to the **ftecfg** command, a compressed file that contains the details of the configuration.

Compressed file structure

The compressed file must be a `.zip` file with the following structure:

```
coordination.properties (properties file — the file must be populated with
                         the properties required to connect to the
                         coordination queue manager)

agents (directory)

        AgentName1 (directory — the name of the first agent, further agents can
                        be defined if required)
            .
            .
            .
        agent.properties (properties file — the file must be populated with
                        the properties required by the agent)
            .
            .
            .
```

You can create the file by compressing a WebSphere MQ Managed File Transfer configuration directory that is created on a non-4690 system. For more information about how to create a compressed file, see "fteBundleConfiguration (create a WebSphere MQ Managed File Transfer IBM 4690 agent configuration .zip file)" on page 90.

Example

```
coordination.properties
agents
    MyFirstAgent
        agent.properties
    MySecondAgent
        agent.properties
```

Related reference:

ftecfg
The **ftecfg** command configures one or more WebSphere MQ Managed File Transfer agents on an IBM 4690 system.

fteBundleConfiguration
Use the **fteBundleConfiguration** command to bundle a WebSphere MQ Managed File Transfer configuration tree, from a specified directory, into a .zip file.

Customizing agent names in a 4690 OS configuration bundle

You can deploy the same configuration bundle to multiple different 4690 OS store controllers. To allow the agent names within a bundle to be customized to match the 4690 OS store controller that they are being deployed to, you can use variable substitution that is based on the store number and the node ID (sometimes called the store controller ID).

Customizing the agent name is useful in the following examples:

- If you want to deploy the same configuration to many stores, you can embed the store number into the store's agent names, so creating a unique agent name.
- If you deploy a Multiple Controller Feature (MCF) Network at one or more stores and want to address a specific store controller, you can embed a store controller's node ID into the agent name.

To use variable substitution for agent names, you must use *@S* and optionally use *@N* as follows:

- Use *@S* to represent a four-digit store number unique to the store controller. For example, 1234.
- Use *@N* to represent a two-character node ID that identifies the store controller in an MCF network. For example, KD.

  For example, the agent name `AGENT@N@S` expands to `AGENTKD1234`.

The store number and node ID are substituted at the point when the configuration is deployed to an installation by using the **ftecfg** command. The agent directories that the **ftecfg** command creates have the name substitution completed before the directories are created. After the directory structure is created on disk, the agentName property in the `agent.properties` file is updated to match the final substituted agent name. A directory structure like the following is created:

```
COORDQM (directory)
     coordination.properties
     MQMFTCredentials.xml (optional)
         agents
             AGENTKD1234
                    agent.properties
```

How to use customized agent names

Complete the following steps:

1. Create a directory structure on your system.
2. Update the agent name to include the substitution variables and ensure that the agent name in this directory structure is in upper case.

   If the agent name is not in upper case, you will get the following error:

   ```
   BFGCL0626W: Ignoring invalid path 'C:\Program Files (x86)\IBM\WebSphere
        MQ\mqft\config\q1\agents\agent@N@S' within agents subdirectory of the source tree.
   ```
3. Ensure that you have the relevant files in place and add the substitution variables in the `agent.properties` file.
4. Run the "fteDefine (generate configuration scripts)" on page 483 command to generate the MQSC scripts for the agent.
1. Modify the `agent.properties` file to set the agentName property by using the substitution variables *@S* for the store number and *@N* for the node identifier where they must be included.
2. Modify the `agent.properties` file to set the properties that are used to connect to the agent queue manager.
3. In the configuration bundle directory structure, rename the directory `name` to match the agent name value you specified for the agentName property in the `agent.properties` file.
4. Modify the `coordination.properties` file to set the properties to connect to the coordination queue manager.

Sample

The `custom1.zip` file contains sample files that customize an agent name to contain a store number and node ID. For more information about how to tailor these sample files to your system, see "Configuration bundle samples for an IBM 4690 system" on page 78.

Related reference:

"ftecfg (creates a WebSphere MQ Managed File Transfer configuration on an IBM 4690 system)" on page 93

The **ftecfg** command configures one or more WebSphere MQ Managed File Transfer agents on an IBM 4690 system.

"Customizing agent properties in a 4690 OS configuration bundle"

If you want to deploy the same configuration to many stores, you can develop one standard, tested configuration bundle and deploy it to all your 4690 OS store controllers, therefore reducing errors. You can then customize that supplied bundle with a substitution.xml file to modify agent properties that are based on the attributes of a store controller.

Customizing agent properties in a 4690 OS configuration bundle

If you want to deploy the same configuration to many stores, you can develop one standard, tested configuration bundle and deploy it to all your 4690 OS store controllers, therefore reducing errors. You can then customize that supplied bundle with a substitution.xml file to modify agent properties that are based on the attributes of a store controller.

You can use a substitution.xml file to evaluate the following conditional expressions to give a true or false value:

- The store controllers store number is equal to a value or is within a defined range
- The store controllers node ID matches a regular expression, which is not case-sensitive
- The store controller has a network interface card (NIC) that is assigned an IP address equal to a value or within a defined range
- The store controller has a NIC that is assigned an MAC address that matches a value

These conditions are evaluated in the order they appear in the substitution.xml file; the first condition that evaluates to true determines the mapping between the symbolic variables and values. If no condition evaluates to true, the default values are used, if a default condition is specified in the XML. Otherwise, a deployment time error is output and no substitution takes place.

You can use the following Boolean operators to connect conditional expressions:

- AND
- OR
- NOT

You can nest operators, which are evaluated from the deepest level of nesting outward. The operators at the deepest level have the highest precedence.

The substitution.xml file is in the same directory as the agent.properties file in the configuration bundle layout before deployment. For example:

```
COORDQM (directory)
    coordination.properties
    MQMFTCredentials.xml (optional)
        agents
            AGENT1
                agent.properties
                substitution.xml (optional)
                UserSandboxes.xml (optional)
```

The agent properties are substituted with the real values at the point when the configuration bundle is deployed to store controller using the **ftecfg** command.

How to substitute agent properties

To use substitution for agent properties, complete the following steps:

1. Set entries in the `agent.properties` file to symbolic values of your choice for the properties that you want to substitute. For example, you can substitute values for the agent queue manager name, the agent queue manager host, and the agent queue manager port number with the following entries:
   - agentQMgr=${QM_NAME}
   - agentHost=${QM_HOST}
   - agentPort=${QM_PORT}
2. Create a `substitution.xml` file, like the following, that defines the conditions that must be satisfied to replace these symbolic values and defines the new values to replace them with.

   For example, based on this `substitution.xml` file, the `agent.properties` file for a store controller that satisfies all the following conditions:
   - Node ID: KM
   - Store number: 1234
   - IP address: 192.168.10.1
   - MAC address: 08-00-27-00-94-2D

   then has the following substitutions made at deployment time:
   - agentQMgr=qmgr1
   - agentHost=host1.example.org
   - agentPort=1414

```xml
<?xml version="1.0" encoding="UTF-8"?>
 <tns:substitution xmlns:tns="http://wmqfte.ibm.com/Substitution"
           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                    xsi:schemaLocation="http://wmqfte.ibm.com/Substitution Substitution.xsd">
     <tns:condition>
       <tns:and>
         <tns:storeNumber low="0" high="4999"/>
         <tns:nodeId matches="KM"/>
         <tns:ipAddress address="192.168.10.1"/>
         <tns:macAddress address="08-00-27-00-94-2D"/>
       </tns:and>
       <tns:variable name="QM_NAME" value="qmgr1"/>
       <tns:variable name="QM_HOST" value="host1.example.org"/>
       <tns:variable name="QM_PORT" value="1414"/>
     </tns:condition>
     <tns:condition>
       <tns:or>
         <tns:storeNumber low="5000" high="9998"/>
         <tns:not><tns:nodeId matches="KM"/></tns:not>
         <tns:ipAddress address="192.168.56.101"/>
       </tns:or>
       <tns:variable name="QM_NAME" value="qmgr2"/>
       <tns:variable name="QM_HOST" value="host2.example.org"/>
       <tns:variable name="QM_PORT" value="1416"/>
     </tns:condition>
     <tns:default>
       <tns:variable name="QM_NAME" value="qmgr3"/>
       <tns:variable name="QM_HOST" value="host3.example.org"/>
       <tns:variable name="QM_PORT" value="1417"/>
     </tns:default>
   </tns:substitution>
```

Sample

The `custom2.zip` file contains sample files that implement substitution based on store controller attributes. For more information about how to tailor these sample files to your system, see "Configuration bundle samples for an IBM 4690 system" on page 78.

Related reference:

"Substitution file format"

You can include a substitution.xml file in a configuration bundle to define how to customize an agent.properties file, based on the attributes of the store controller that the configuration bundle is deployed to.

"ftecfg (creates a WebSphere MQ Managed File Transfer configuration on an IBM 4690 system)" on page 93

The **ftecfg** command configures one or more WebSphere MQ Managed File Transfer agents on an IBM 4690 system.

"Customizing agent names in a 4690 OS configuration bundle" on page 69

You can deploy the same configuration bundle to multiple different 4690 OS store controllers. To allow the agent names within a bundle to be customized to match the 4690 OS store controller that they are being deployed to, you can use variable substitution that is based on the store number and the node ID (sometimes called the store controller ID).

Substitution file format

You can include a substitution.xml file in a configuration bundle to define how to customize an agent.properties file, based on the attributes of the store controller that the configuration bundle is deployed to.

The substitution.xml file must conform to the Substitution.xsd schema. The Substitution.xsd schema document is in the *MQ_INSTALLATION_PATH*/mqft/samples/schema directory of the MQMFT installation. Sample files are available in the *MQ_INSTALLATION_PATH*/mqft/samples/4690 directory of the MQMFT installation.

Schema

The following schema describes which elements are valid in the Substitution.xsd schema.

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
@start_non_restricted_prolog@
Version: %Z% %I% %W% %E% %U% [%H% %T%]

Licensed Materials - Property of IBM

5724-H72

Copyright IBM Corp. 2013 All Rights Reserved.

US Government Users Restricted Rights - Use, duplication or
disclosure restricted by GSA ADP Schedule Contract with
IBM Corp.
@end_non_restricted_prolog@
 -->

<!-- This schema defines the format of a substitution.xml file. Files of this type
     define a set of substitution variables and conditions on how they should be
     applied to and agents agent.properties file at configuration deployment time.
     -->

<!-- Example substitution.xml file:

 <?xml version="1.0" encoding="UTF-8"?>
 <tns:substitution xmlns:tns="http://wmqfte.ibm.com/Substitution"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                   xsi:schemaLocation="http://wmqfte.ibm.com/Substitution Substitution.xsd">
     <condition>
       <and>
         <storeNumber low="0" high="4999"/>
         <nodeId matches="KM"/>
         <ipAddress address="192.168.10.1"/>
         <macAddress address="08-00-27-00-94-2D"/>
       </and>
       <variable name="QM_NAME" value="qmgr1"/>
       <variable name="QM_HOST" value="host1.example.org"/>
```

```
          <variable name="QM_PORT" value="1414"/>
        </condition>
        <condition>
          <or>
            <storeNumber low="5000" high="9998"/>
            <not><nodeId matches="KM"/></not>
            <ipAddress address="192.168.10.1" mask="255.255.0.0"/>
          </or>
          <variable name="QM_NAME" value="qmgr2"/>
          <variable name="QM_HOST" value="host2.example.org"/>
          <variable name="QM_PORT" value="1416"/>
        </condition>
        <default>
          <variable name="QM_NAME" value="qmgr3"/>
          <variable name="QM_HOST" value="host3.example.org"/>
          <variable name="QM_PORT" value="1417"/>
        </default>
      </tns:substitution>

 -->

<schema targetNamespace="http://wmqfte.ibm.com/Substitution"
        elementFormDefault="qualified"
     xmlns="http://www.w3.org/2001/XMLSchema"
        xmlns:tns="http://wmqfte.ibm.com/Substitution">

  <element      name="substitution" type="tns:mqmftSubstitutionType"/>

  <complexType name="mqmftSubstitutionType">
    <sequence>
      <element name="condition" type="tns:conditionType" minOccurs="0" maxOccurs="unbounded"/>
      <element name="default" type="tns:defaultType" minOccurs="0" maxOccurs="1"/>
    </sequence>
  </complexType>


  <!--
    Defines the condition element.

    <condition>
      <and>
        ...
      </and>
      <variable name="QM_NAME" value="QM_test"/>
    </condition>

    or

    <condition>
      <or>
        ...
      </or>
      <variable name="QM_HOST" value="qmtest.hursley.ibm.com"/>
      <variable name="QM_PORT" value="1414"/>
    </condition>

    or

    <condition>
      <not>
        ...
      </not>
      <variable name="QM_HOST" value="qmtest.hursley.ibm.com"/>
    </condition>

  -->
  <complexType name="conditionType">
    <sequence>
      <choice minOccurs="1" maxOccurs="1">
        <element name="and"         type="tns:andType"         minOccurs="0" maxOccurs="1"/>
        <element name="or"          type="tns:orType"          minOccurs="0" maxOccurs="1"/>
        <element name="not"         type="tns:notType"         minOccurs="0" maxOccurs="1"/>
        <element name="storeNumber" type="tns:storeNumberType" minOccurs="0" maxOccurs="1"/>
        <element name="nodeId"      type="tns:nodeIdType"      minOccurs="0" maxOccurs="1"/>
        <element name="ipAddress"   type="tns:ipAddressType"   minOccurs="0" maxOccurs="1"/>
        <element name="macAddress"  type="tns:macAddressType"  minOccurs="0" maxOccurs="1"/>
      </choice>
      <element name="variable" type="tns:variableType" minOccurs="1" maxOccurs="unbounded"/>
    </sequence>
```

```
    </complexType>


<!--
  Defines the and element.

  <and>
    <storeNumber low="0" high="4999"/>
    <nodeId matches="KM"/>
  </and>
-->
<complexType name="andType">
  <sequence>
    <choice minOccurs="2" maxOccurs="unbounded">
      <element name="storeNumber" type="tns:storeNumberType" minOccurs="0" maxOccurs="1"/>
      <element name="nodeId"      type="tns:nodeIdType"      minOccurs="0" maxOccurs="1"/>
      <element name="ipAddress"   type="tns:ipAddressType"   minOccurs="0" maxOccurs="1"/>
      <element name="macAddress"  type="tns:macAddressType"  minOccurs="0" maxOccurs="1"/>
      <element name="or"          type="tns:orType"          minOccurs="0" maxOccurs="unbounded"/>
      <element name="not"         type="tns:notType"         minOccurs="0" maxOccurs="unbounded"/>
    </choice>
  </sequence>
</complexType>


<!--
  Defines the or element.

  <or>
    <ipAddress address="192.168.10.1"/>
    <macAddress address="08-00-27-00-94-2D"/>
  </or>
-->
<complexType name="orType">
  <sequence>
    <choice minOccurs="2" maxOccurs="unbounded">
      <element name="storeNumber" type="tns:storeNumberType" minOccurs="0" maxOccurs="1"/>
      <element name="nodeId"      type="tns:nodeIdType"      minOccurs="0" maxOccurs="1"/>
      <element name="ipAddress"   type="tns:ipAddressType"   minOccurs="0" maxOccurs="1"/>
      <element name="macAddress"  type="tns:macAddressType"  minOccurs="0" maxOccurs="1"/>
      <element name="and"         type="tns:andType"         minOccurs="0" maxOccurs="unbounded"/>
      <element name="not"         type="tns:notType"         minOccurs="0" maxOccurs="unbounded"/>
    </choice>
  </sequence>
</complexType>


<!--
  Defines the not element.

  <not><storeNumber low="0" high="4999"/></not>
-->
<complexType name="notType">
  <sequence>
    <choice minOccurs="1" maxOccurs="1">
      <element name="storeNumber" type="tns:storeNumberType" minOccurs="0" maxOccurs="1"/>
      <element name="nodeId"      type="tns:nodeIdType"      minOccurs="0" maxOccurs="1"/>
      <element name="ipAddress"   type="tns:ipAddressType"   minOccurs="0" maxOccurs="1"/>
      <element name="macAddress"  type="tns:macAddressType"  minOccurs="0" maxOccurs="1"/>
      <element name="and"         type="tns:andType"         minOccurs="0" maxOccurs="1"/>
      <element name="or"          type="tns:orType"          minOccurs="0" maxOccurs="1"/>
    </choice>
  </sequence>
</complexType>


<!--
  Defines the storeNumber element.

  <storeNumber low="0" high="3999"/>
-->
<complexType name="storeNumberType">
  <attribute name="low"  type="nonNegativeInteger" use="required"/>
  <attribute name="high" type="nonNegativeInteger" use="required"/>
</complexType>


<!--
  Defines the nodeId element.
```

```
    <nodeId matches="KM"/>
-->
<complexType name="nodeIdType">
  <attribute name="matches" type="string" use="required"/>
</complexType>


<!--
  Defines the ipAddress element.

  <ipAddress address="192.168.0.1" mask="255.255.0.0"/>

  or

  <ipAddress address="192.168.0.1"/>
-->
<complexType name="ipAddressType">
  <attribute name="address"  type="string" use="required"/>
  <attribute name="mask"     type="string" use="optional"/>
</complexType>


<!--
  Defines the macAddress element.

  <macAddress address="08-00-27-00-94-2D"/>
-->
<complexType name="macAddressType">
  <attribute name="address" type="string" use="required"/>
</complexType>


<!--
  Defines the default element.

  <default>
    <variable name="QM_NAME" value="QM_test"/>
    <variable name="QM_HOST" value="qmtest.hursley.ibm.com"/>
    <variable name="QM_PORT" value="1414"/>
  </default>
-->
<complexType name="defaultType">
  <sequence>
    <element name="variable" type="tns:variableType" minOccurs="1" maxOccurs="unbounded"/>
  </sequence>
</complexType>


<!--
  Defines the variable element.

  <variable name="QM_NAME" value="QM_test"/>
-->
<complexType name="variableType">
  <attribute name="name"  type="string" use="required"/>
  <attribute name="value" type="string" use="required"/>
</complexType>

</schema>
```

The Substitution.xsd file

The elements and attributes used in the Substitution.xsd file are described in the following list.

<substitution>
> Group element containing elements that describe the information to be substituted from the substitution.xml file to the agent.properties file, and under what conditions.

<condition>
> A conditional expression that is dependent on store controller attributes. The <condition> element can contain the elements <and>, <or>, and <not>.

<default>
> The default values to use for the substituted agent properties if no condition evaluates to true.

`<variable>`

The name and value of a variable for an agent property that you want to substitute.

Attribute	Description
name	Variable name. For example, QM_NAME, QM_HOST, or QM_PORT
value	Value that you want the variable to take

`<and>`

The AND Boolean operator that is used to connect conditional expressions.

`<or>`

The OR Boolean operator that is used to connect conditional expressions.

`<not>`

The NOT Boolean operator that is used to exclude conditional expressions.

`<storeNumber>`

The store controller's store number or a range of store controller numbers. For example, 1234 or 0 - 4999.

Attribute	Description
low	The lowest number in a range of store numbers
high	The highest number in a range of store numbers

`<nodeId>`

The node ID (also known as the store controller ID) that you want to match. For example, KM.

Attribute	Description
matches	A node ID to match (the node ID is not case-sensitive)

`<ipAddress>`

The IP address that is assigned to the store controller's network interface card (NIC).

Attribute	Description
address	The IP address to match, for example 192.168.10.1
mask	The subnet mask to use when matching the IP address, for example 255.255.0.0. This attribute is optional but when specified it causes the IP address of the store controller to be matched against a range of IP addresses. The range of IP addresses is defined by a combination of the address and mask attributes.

`<macAddress>`

The MAC address that is assigned to the store controller's network interface card (NIC). For example, 08-00-27-00-94-2D.

Attribute	Description
address	The MAC address to match

Example

```
<?xml version="1.0" encoding="UTF-8"?>
 <tns:substitution xmlns:tns="http://wmqfte.ibm.com/Substitution"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
                           xsi:schemaLocation="http://wmqfte.ibm.com/Substitution Substitution.xsd">
      <tns:condition>
        <tns:and>
          <tns:storeNumber low="0" high="4999"/>
          <tns:nodeId matches="KM"/>
          <tns:ipAddress address="192.168.10.1"/>
          <tns:macAddress address="08-00-27-00-94-2D"/>
        </tns:and>
        <tns:variable name="QM_NAME" value="qmgr1"/>
        <tns:variable name="QM_HOST" value="host1.example.org"/>
        <tns:variable name="QM_PORT" value="1414"/>
      </tns:condition>
      <tns:condition>
        <tns:or>
          <tns:storeNumber low="5000" high="9998"/>
          <tns:not><tns:nodeId matches="KM"/></tns:not>
          <tns:ipAddress address="192.168.56.101"/>
        </tns:or>
        <tns:variable name="QM_NAME" value="qmgr2"/>
        <tns:variable name="QM_HOST" value="host2.example.org"/>
        <tns:variable name="QM_PORT" value="1416"/>
      </tns:condition>
      <tns:default>
        <tns:variable name="QM_NAME" value="qmgr3"/>
        <tns:variable name="QM_HOST" value="host3.example.org"/>
        <tns:variable name="QM_PORT" value="1417"/>
      </tns:default>
    </tns:substitution>
```

Configuration bundle samples for an IBM 4690 system

A number of sample configuration bundles are provided in the `MQ_INSTALLATION_PATH/mqft/samples/4690`
directory. You can unpackage each sample with the **fteBundleConfiguration** command and then modify
the extracted files for your specific configuration. When modified, you can use the
fteBundleConfiguration command to package the files into a configuration bundle that can be deployed
to your IBM 4690 system.

basic.zip

You can use the `basic.zip` sample to configure an agent with minimal customization. The structure of the
configuration bundle is as follows:

```
agents/
  name/
    agent.properties
coordination.properties
```

To customize this sample for your requirements, complete the following steps:

1. Modify the `agent.properties` file to set the agentName property and to set the properties that are
   used to connect to the agent queue manager.
2. In the configuration bundle directory structure, rename the directory `name` to match the agent name
   value you specified for the agentName property in the `agent.properties` file.
3. Modify the `coordination.properties` file to set the properties to connect to the coordination queue
   manager.

custom1.zip

You can use the `custom1.zip` sample to configure the agent name to contain the store number, node
identifier, or both. This update then allows the configuration to be deployed to multiple 4690 systems.
The structure of the configuration bundle is as follows:

```
agents/
  name/
    agent.properties
coordination.properties
```

To customize this sample for your requirements, complete the following steps:

1. Modify the `agent.properties` file to set the agentName property by using the substitution variables *@S* for the store number and *@N* for the node identifier where they must be included.

2. Modify the `agent.properties` file to set the properties that are used to connect to the agent queue manager.

3. In the configuration bundle directory structure, rename the directory `name` to match the agent name value you specified for the agentName property in the `agent.properties` file.

4. Modify the `coordination.properties` file to set the properties to connect to the coordination queue manager.

custom2.zip

You can use the `custom2.zip` sample to configure the contents of the `agent.properties` file dependent on the store number, node identifier, IP address, and network MAC address of the store controller that the bundle is being deployed to. This configuration is done by defining one or more conditions in the `substitution.xml` file. The conditions set substitution variables that can then be substituted into the `agent.properties` file. The structure of the configuration bundle is as follows:

```
agents/
  name/
    agent.properties
    substitution.xml
coordination.properties
```

To customize this sample for your requirements, complete the following steps:

1. Modify the `substitution.xml` file to specify conditions and variables that are applicable to the configuration.

2. Modify the `agent.properties` file to set the agentName property. You can customize the agentName by using the substitution variables *@S* for the store number and *@N* for the node identifier in the same way as for the `custom1.zip` sample. You cannot however customize the agentName by using substitution variables that are specified in the `substitution.xml` file.

3. Modify the `agent.properties` file to set the properties that are used to connect to the agent queue manager, specifying substitution variables where required.

4. In the configuration bundle directory structure, rename the directory `name` to match the agent name value you specified for the agentName property in the `agent.properties` file.

5. Modify the `coordination.properties` to set the properties to connect to the coordination queue manager.

SSL.zip

You can use the `SSL.zip` sample to configure an agent to connect to the agent queue manager by using SSL. The sample specifies a configuration to be deployed to a single 4690 system. However, you can also use the techniques that are described in the custom1 and custom2 samples with SSL to deploy the configuration to multiple systems. The structure of the configuration bundle is as follows:

```
agents/
  name/
    agent.properties
coordination.properties
MQMFTCredentials.xml
```

To customize this sample for your requirements, complete the following steps:

1. Modify the `agent.properties` file to set the agentName property and the properties to connect to the agent queue manager.
2. Modify the `agent.properties` file to set the properties to configure SSL. For more information, see comments in the sample `agent.properties` file.
3. In the configuration bundle directory structure, rename the directory `name` to match the agent name value you specified for the agentName property in the `agent.properties` file.
4. Modify the `coordination.properties` to set the properties to connect to the coordination queue manager.
5. Modify the `MQMFTCredentials.xml` file to specify the passwords for any keystore and truststore files that are specified in SSL configuration.

Related tasks:

"Creating a WebSphere MQ Managed File Transfer configuration on an IBM 4690 system" on page 68
To create or replace a WebSphere MQ Managed File Transfer configuration on an IBM 4690 system, you first create the configuration on a non-4690 platform. You then copy the configuration to the IBM 4690 system in a compressed file and run the **ftecfg** command to deploy the configuration to the IBM 4690 system.

Related reference:

"Customizing agent names in a 4690 OS configuration bundle" on page 69
You can deploy the same configuration bundle to multiple different 4690 OS store controllers. To allow the agent names within a bundle to be customized to match the 4690 OS store controller that they are being deployed to, you can use variable substitution that is based on the store number and the node ID (sometimes called the store controller ID).

"Structure of the IBM 4690 WebSphere MQ Managed File Transfer configuration compressed file" on page 69
You create a WebSphere MQ Managed File Transfer configuration on an IBM 4690 system by passing, as a parameter to the **ftecfg** command, a compressed file that contains the details of the configuration.

"Customizing agent properties in a 4690 OS configuration bundle" on page 71
If you want to deploy the same configuration to many stores, you can develop one standard, tested configuration bundle and deploy it to all your 4690 OS store controllers, therefore reducing errors. You can then customize that supplied bundle with a `substitution.xml` file to modify agent properties that are based on the attributes of a store controller.

"fteBundleConfiguration (create a WebSphere MQ Managed File Transfer IBM 4690 agent configuration .zip file)" on page 90
Use the **fteBundleConfiguration** command to bundle a WebSphere MQ Managed File Transfer configuration tree, from a specified directory, into a .zip file.

"ftecfg (creates a WebSphere MQ Managed File Transfer configuration on an IBM 4690 system)" on page 93
The **ftecfg** command configures one or more WebSphere MQ Managed File Transfer agents on an IBM 4690 system.

Configuring WebSphere MQ Managed File Transfer in a master-backup 4690 OS controller setup

You can configure agents in a master-backup 4690 OS controller setup to provide fault tolerance. Agents with the same name can be configured to run on multiple controllers in a retail environment. However, only one of the agents can run at any one time. This configuration can be combined with applications that are configured to stop and start in different circumstances.

You can configure permanent background applications on 4690 in the following ways:

- The applications are started when the store controller becomes the acting master controller, and are stopped when the controller stops being the acting master controller.

- The applications are started when the store controller becomes the acting file server, and are stopped when the controller stops being the acting file server.

Therefore, by configuring agents, with the same name, to be background applications in this way, on a master and alternative master store controller configuration, or on a file server and alternative file server store controller configuration, you can provide a fault tolerant setup for WebSphere MQ Managed File Transfer on a 4690 OS system.

For details on how to configure agents, see "Starting an agent on a 4690 OS system" on page 82.

Related concepts:

"Configuring multiple WebSphere MQ Managed File Transfer agents in a 4690 OS controller setup"
You can configure multiple agents in a single store controller environment, or in an environment where there are multiple store controllers for a store.

Configuring multiple WebSphere MQ Managed File Transfer agents in a 4690 OS controller setup

You can configure multiple agents in a single store controller environment, or in an environment where there are multiple store controllers for a store.

You can configure multiple agents to run on a single store controller, if their names are unique for that controller, and that a background application slot is available for each agent.

In an environment where there are multiple store controllers for a store, it can be useful to configure the following agents for each store controller:

- Agents of the same name that are configured for a master and alternative master store controller for the store, such that an agent is running only when its store controller is the acting master controller. This agent can be used to transfer files that are common or shared across the two controllers for a store. You can also use the same configuration for a file server and alternative file server store controller.
- An agent with a name that is unique to the store controller, and that is configured to be running whenever the store controller is running. This agent can be used to transfer files that are specific to the store controller.

In this example, store 7777 has two controllers KD and KM.

Agents that are configured for store 7777, and controller KD:
- Agent1: AGENT7777 - Store specific agent (master instance)
- Agent2: AGENT7777KD - Controller-specific agent for controller KD

Agents that are configured for store 7777, and controller KM:
- Agent1: AGENT7777 - Store specific agent (alternative instance)
- Agent2: AGENT7777KM - Controller-specific agent for controller KM

For details on how to configure agents, see "Starting an agent on a 4690 OS system" on page 82.

Related concepts:

"Configuring WebSphere MQ Managed File Transfer in a master-backup 4690 OS controller setup" on page 80
You can configure agents in a master-backup 4690 OS controller setup to provide fault tolerance. Agents with the same name can be configured to run on multiple controllers in a retail environment. However, only one of the agents can run at any one time. This configuration can be combined with applications that are configured to stop and start in different circumstances.

Starting an agent on a 4690 OS system

To start an agent, you must first configure it as a background application. Background applications can be configured to start automatically when the system becomes the acting master controller or file server, and to stop when the system is no longer the acting master controller or file server.

About this task

Agents that are available to be started are already defined by running the `f:\adxetc\mft75\bin\ftecfg.bat` utility to load a new configuration. For each configured agent, there is an `agent_name.rsp` response file in the `f:\adxetc\mft75` directory. Each of these response files contains a command to start a particular agent when passed to the `ADXCHAIN.386` system command.

Procedure

Complete the following steps to start an agent that runs as a background service:
1. From the SYSTEM MAIN MENU panel, select **4** (**Installation and Update Aids**).
2. From the INSTALLATION AND UPDATE AIDS panel, select **1** (**Change Configuration Data**).
3. From the CONFIGURATION panel, select **2** (**Controller Configuration**).
4. If you are asked whether you are configuring a store system that uses the IBM Multiple Controller Feature, press **Enter** to select **Yes**.
5. If the LAN CONFIGURATION panel is displayed, select the options that are appropriate for your environment, and press **Enter**.
6. If the SNA CONFIGURATION panel is displayed, select the options that are appropriate for your environment, and press **Enter**.
7. If you are prompted to enter store controller IDs, specify the appropriate controller IDs, and press **Enter**.
8. Select the store controller that you want to configure, and press **Enter**.
9. From the list of controller configuration items, select **Background Application**, and press **Enter**.
10. From the BACKGROUND APPLICATION panel, select **1** (Define a Background Application).
11. On the DEFINE BACKGROUND APPLICATION panel, specify the following entries:
    - Initial message: `MQMFT`
    - Program name: `ADX_SPGM:ADXCHAIN.386`
    - Parameter list: `@f:\adxetc\mft75\`*agent_name*`.rsp`

      Note: The *agent_name* is restricted to a maximum of 23 characters, and the parameter list entry is restricted to a maximum of 45 characters. The parameter list path to the response file must be specified exactly in the format that is shown, meaning no change to uppercase characters or forward slashes. Defining a logical name to specify the path to the response file is not permitted.
12. Press **PgDn** to see more options, and specify whether the application should be started or stopped when the system becomes the acting master or stops being the acting master.
13. Press **PgDn** again to see further options, and specify whether the application should be started or stopped when the system becomes the acting file server or stops being the acting file server.

14. Press **Enter** to save the changes.

15. Press **Esc** to return to the CONFIGURATION panel.

16. From the CONFIGURATION panel, select **4** (**Activate Configuration**).

17. From the ACTIVATE CONFIGURATION panel, select **2** (**Controller Configuration**). The controller configuration is activated.

18. Re-IPL the store controller.

Agent status messages

On an IBM 4690 system you can see agent status messages for a WebSphere MQ Managed File Transfer supplied background application by navigating to the background applications control panel. There are several possible status messages.

```
hh:mm Initializing
hh:mm Starting
hh:mm Not connected to MQ (last MQRC=nnnn)
hh:mm Recovering
hh:mm Waiting for work
hh:mm Transferring source:nnnn destination:nnnn
hh:mm Failed (RC=nnnn)
hh:mm Controlled stopping
hh:mm Immediate stopping
hh:mm Controlled stopped
hh:mm Immediate stopped
```

Initializing
> The agent is starting and is initializing the resources that it requires.

Starting
> The agent is successfully initialized and is preparing to start processing transfers.

Not connected to MQ
> The agent is running but is waiting for the queue manager to become available. The agent is unable to process transfers until reconnection is successful. For more information about the `last` MQRC, see WebSphere MQ Reason codes.

Recovering
> The agent is running and re-established a connection to the queue manager. The agent is recovering any transfers that were running when the connection was lost.

Waiting for work
> The agent is running, but there are no transfers in progress.

Transferring source:nnnn destination:nnnn
> The agent is running, and is transferring files. The number of transfers where the agent is the source and where the agent is the destination is indicated by the values shown.

Failed The agent failed, and recovery was not possible. For more information about the `RC`, see the table of return codes later in the topic.

Controlled stopping
> The agent is stopping. A controlled shutdown was requested, allowing any running transfers to complete.

Immediate stopping
> The agent is stopping. An immediate shutdown was requested.

Controlled stopped
> The agent is stopped. It was shut down in a controlled manner.

Immediate stopped
> The agent is stopped. It was shut down in an uncontrolled manner.

The following table lists the return codes with their meanings:

Table 2. Return codes

Return code	Short name	Description
0	Success	The application ended successfully.
1	Failure	General application failure return code.
2	Exit	The application was forced to exit. For example, a diagnostic system requested the application to terminate.
70	Abend	The application had an unrecoverable problem and was forcibly terminated.
78	Config	The application cannot continue because there is a problem with the startup configuration data.

Related tasks:

"Starting an agent on a 4690 OS system" on page 82
To start an agent, you must first configure it as a background application. Background applications can be configured to start automatically when the system becomes the acting master controller or file server, and to stop when the system is no longer the acting master controller or file server.

Related reference:

"Process controller status messages"
On an IBM 4690 system you can see process controller status messages for a WebSphere MQ Managed File Transfer supplied background application by navigating to the background applications control panel. There are several possible status messages.

Process controller status messages

On an IBM 4690 system you can see process controller status messages for a WebSphere MQ Managed File Transfer supplied background application by navigating to the background applications control panel. There are several possible status messages.

```
hh:mm Starting
hh:mm Monitoring agent process
hh:mm nnnn agent restarts (last: hh:mm:ss)
hh:mm Failed (RC=nnnn)
hh:mm Stopped
```

Starting
 The process controller is successfully initialized and is preparing to start monitoring the agent.

Monitoring agent process
 The process controller is monitoring the agent process.

nnnn agent restarts (last: hh:mm:ss)
 The total number of restarts of the agent by the process controller since the process controller started, and the time of the last agent restart.

Failed The process controller failed. For more information about the RC, see the table of return codes later in the topic.

Stopped
 The process controller is stopped. It was shut down in a controlled manner.

The following table lists the return codes with their meanings:

Table 3. Return codes

Return code	Short name	Description
0	Success	The application ended successfully.
1	Failure	General application failure return code.
2	Exit	The application was forced to exit. For example, a diagnostic system requested the application to terminate.
70	Abend	The application had an unrecoverable problem and was forcibly terminated.
78	Config	The application cannot continue because there is a problem with the startup configuration data.

Related tasks:

"Starting an agent on a 4690 OS system" on page 82
To start an agent, you must first configure it as a background application. Background applications can be configured to start automatically when the system becomes the acting master controller or file server, and to stop when the system is no longer the acting master controller or file server.

Related reference:

"Agent status messages" on page 83
On an IBM 4690 system you can see agent status messages for a WebSphere MQ Managed File Transfer supplied background application by navigating to the background applications control panel. There are several possible status messages.

Restrictions when running on a 4690 OS system

There are a number of restrictions and unsupported functions when you run WebSphere MQ Managed File Transfer on a 4690 OS system in a retail environment.

The following restrictions apply on 4690 OS:

- A 4690 OS agent name can be a maximum of 23 characters only.
- The 4690 OS agent response files are in the `f:\adxetc\mft75` directory.
- For an agent defined as a background application, the `PARAM` setting must be in the format `@f:\adxetc\mft75\`*agent_name*`.rsp`.

  Note: The parameter list entry is restricted to a maximum of 45 characters. The parameter list path to the response file must be specified exactly in the format that is shown. Meaning no change to uppercase characters or forward slashes. Defining a logical name to specify the path to the response file is not allowed.
- You can define only one coordination queue manager on a 4690 OS system.
- The transferRoot property does not have a default directory path on 4690 OS. If you want to use relative paths for transfers to or from a 4690 agent, you must set transferRoot. For more information, see Advanced agent properties.

The following features are not supported on 4690 OS:

- You cannot run fteAnt scripts directly on the 4690 OS platform.
- You cannot run Apache Ant scripts from presrc or postsrc program invocations for a transfer that has an 4690 OS agent as the source agent.

- You cannot run Ant scripts from predest or postdest program invocations for a transfer that has an 4690 OS agent as the destination agent.
- You cannot configure a WebSphere MQ Managed File Transfer logger on the 4690 OS platform. This restriction includes all three versions: Stand-alone file logger, stand-alone database logger, and Java™ Platform Enterprise Edition (JEE) logger.
- You cannot configure an agent as a protocol bridge or Connect:Direct bridge on the IBM 4690 platform.
- You cannot configure the Web Gateway on the 4690 OS platform.
- International Components for Unicode (ICU) for Java is not supported on the 4690 OS platform. This means that for text file transfers the character set converters that are used are supplied by Java Runtime Environment Version 6.0 on the 4690 OS platform.

When a path value is required as part of the configuration on 4690 OS, two path formats are supported:
- `drive-letter:\remainder-of-path`. For example; `f:\adxetc\mft75`
- `logical-name:remainder-of-path`. For example; `f_drive:adxetc\mft75`

For WebSphere MQ Managed File Transfer if a backslash (\) is included at the start of the `remainder-of-path` it is ignored to give more predictable behavior.

In addition to this restriction, some properties are not supported on 4690 OS. The following table lists all the properties that expect a path as their value and indicates whether they are supported for use on 4690 OS.

Path properties that are supported on 4690 OS.	Path properties that are not supported on 4690 OS.
agentSslKeyStore	cdTmpDir
agentSslKeyStoreCredentialsFile	cdNodeKeystoreCredentialsFile
agentSslTrustStore	cdNodeTruststoreCredentialsFile
agentSslTrustStoreCredentialsFile	exitNativeLibraryPath
commandPath	wmqfte.database.credentials.file
exitClassPath	
javaCoreTriggerFile	
sandboxRoot	
transferRoot	
connectionSslKeyStore	
connectionSslKeyStoreCredentialsFile	
connectionSslTrustStore	
connectionSslTrustStoreCredentialsFile	
coordinationSslKeyStore	
coordinationSslKeyStoreCredentialsFile	
coordinationSslTrustStore	
coordinationSslTrustStoreCredentialsFile	

File distribution attributes

When WebSphere MQ Managed File Transfer transfers files to a destination agent running on an IBM 4690 store controller, it must set the file distribution attributes of the files. These distribution attributes determine whether the IBM 4690 store controller duplicates the file's contents with other store controllers and how it duplicates the file's contents in a Multiple Controller Feature (MCF) Network. WebSphere MQ Managed File Transfer file distribution attributes combine MCF's file type and file mode attributes.

You can set the distribution attributes for files at the destination in the following ways:

- Using the Add transfer item panel in WebSphere MQ Explorer.
- Using the **-dfa** parameter on the **fteCreateTransfer** command.

  If you do not specify the **-dfa** parameter for a transfer with a 4690 destination, the default distribution attribute is LOCAL. If the file already exists at the destination, the current distribution attribute of the existing file is used. For more information, see "Parameters for specifying the destination" on page 469.

Each combination of file type and file mode is represented by the following symbolic and numeric values. Specify either the symbolic or numeric value.

Table 4. File distribution attributes in WebSphere MQ Managed File Transfer

Symbolic value	Numeric value	Description
DIST(LOCAL)	DIST(1)	Local file. A local file exists on one store controller only. When a local file is updated, other store controllers in the network are not affected.
DIST(MIRRORED,UPDATE)	DIST(2)	Mirrored file, distribute at update A mirrored file exists on two store controllers and consists of a prime version and an image version. When a record in the prime version of the file is changed or deleted, the same record in the image version is also changed or deleted.
DIST(MIRRORED,CLOSE)	DIST(3)	Mirrored file, distribute at close A mirrored file exists on two store controllers and consists of a prime version and an image version. When the prime version of the file is closed, the image version of the file is then updated.
DIST(COMPOUND,UPDATE)	DIST(4)	Compound file, distribute at update A compound file is distributed to all store controllers except ineligible subordinate store controllers. When a record in the prime version of the file is changed or deleted, the same record in the image versions of the file is also changed or deleted.

Table 4. File distribution attributes in WebSphere MQ Managed File Transfer (continued)

Symbolic value	Numeric value	Description
DIST(COMPOUND,CLOSE)	DIST(5)	Compound file, distribute at close A compound file is distributed to all store controllers except ineligible subordinate store controllers. When the prime version of the file is closed, the image versions of the file are then updated.

For more information about MCF, see Chapter 22, "Using the Multiple Controller Feature" in 4690 OS Version 6 Release 3 User's Guide.

Directory requirements for using file distribution

For files to be distributed as you expect, ensure that the same directory structure exists on all the store controllers in the network. Typically, this directory structure is created on the store controllers by the operating system or the user at installation. If the directory structure is not replicated on all controllers, the distribution fails for the controllers that do not have the required directories. Distribution failures are ignored and not logged.

For example, if files with a distribution attribute MIRRORED,UPDATE are transferred to IBM 4690 and the destination agent creates directories, the files transferred to those new directories are not distributed because the directories on the other controllers are not created by WebSphere MQ Managed File Transfer.

Drives that support distribution

Distributed files can only be located in directories off the root directory and normally are only found in directories created by the operating system during installation. Generally, distributed files are located on drives that use the 8.3 naming convention, that is, drives C: and D:. Files with distribution attributes are not supported on the F: drive. The logical drives M: and N: are the exceptions because they are created using the 4690 Virtual File System, which was created to support long file and directory names.

Table 5. Drives that support distribution

IBM 4690 drive letter	Supports file distribution?
C:	YES
D:	YES
F:	NO
M:	YES Information on the logical M: drive is stored on the C: drive using VFS
N:	YES Information on the logical N: drive is stored on the D: drive using VFS

Related information:
TRANSFER_ITEM_ATTRIBUTES database logger table

fte:filespec dstAttributes attribute

Working in a sandbox on IBM 4690

WebSphere MQ Managed File Transfer uses sandboxing to restrict the area of the file system that an agent or a user can access as part of a transfer. To limit access, agent sandboxing uses the sandboxRoot property and user sandboxing uses the information in the `UserSandboxes.xml` file.

The sandboxRoot property and information in the `UserSandboxes.xml` file both specify path information that is compared against the paths specified in a transfer request. On 4690, the way that path information is interpreted in these files is subject to the following, additional, rules:

* Paths are assumed to be native 4690 paths, for example: `f:\adxetc`
* Paths on F: drive are case-sensitive. Paths on all drives except F: drive are case-insensitive.
* You can specify paths that contain logical names. Logical names are expanded as part of the comparison. Therefore, `LN1:\DIR\FILE.TXT` might be the same as `LN2:\FILE.TXT` depending on the path information associated with LN1 and LN2.

For more information about agent sandboxes, see "Working with agent sandboxes" on page 104 and for more information about user sandboxes, see "Working with user sandboxes" on page 105.

Summary of the WebSphere MQ Managed File Transfer commands for use in a retail environment

All WebSphere MQ Managed File Transfer commands for use in a retail environment are listed with links to their detailed descriptions.

Table 6. WebSphere MQ Managed File Transfer commands and their purpose

Command name	Purpose
fteBundleConfiguration	Bundle a WebSphere MQ Managed File Transfer configuration tree, from a specified directory, into a .zip file.
ftelap	Run the License Acceptance Process (LAP) tool. The LAP tool is used to read and accept the license that is associated with WebSphere MQ Managed File Transfer.
ftecfg	Configures one or more WebSphere MQ Managed File Transfer agents on an IBM 4690 system.
ftediag	Generate diagnostic information for a WebSphere MQ Managed File Transfer agent on an IBM 4690 system.
uninstall	Uninstall the WebSphere MQ Managed File Transfer agent from an IBM 4690 store controller.

The syntax for each command and its parameters is presented in the form of a syntax diagram that is called a railroad diagram. For information about how to interpret railroad diagrams, see How to read railroad diagrams.

fteBundleConfiguration (create a WebSphere MQ Managed File Transfer IBM 4690 agent configuration .zip file)

Use the **fteBundleConfiguration** command to bundle a WebSphere MQ Managed File Transfer configuration tree, from a specified directory, into a .zip file.

Purpose

The **fteBundleConfiguration** command packages, and unpackages the configuration files necessary for an IBM 4690 installation. The .zip file that is created can be supplied to the **ftecfg** command for use with an IBM 4690 installation. The **fteBundleConfiguration** command validates the files in the directory to ensure that the configuration is functional, warnings are displayed if the input is malformed. Any agent configurations which are correctly formed are bundled, producing a usable configuration .zip file. The **ftecfg** command fails to deploy configurations which are not correctly formed.

Note: Only WebSphere MQ Managed File Transfer Version 7.5 and later configurations are supported.

Paths to SSL property files

The SSL properties, agentSslKeyStore, agentSslKeyStoreCredentialsFile, agentSslTrustStore, agentSslTrustStoreCredentialsFile, and the mqmftcredentials.xml file contain path values that can be included in the configuration bundle. The **fteBundleConfiguration** command processes these values according to whether the path is relative or absolute.

Local (non-4690 platform) absolute path

> If the absolute path is in the bundle configuration directory and the file exists, the path is converted to relative, and included in the configuration bundle. The **ftecfg** command unpackages to an absolute path when the bundle is deployed.

> If the absolute path is not in the bundle configuration directory or the file does not exist, an error message is produced.

Local (non-4690 platform) relative path

> If the relative path refers to a file that exists in the bundle configuration directory, the path is included in the configuration bundle. The **ftecfg** command unpackages to an absolute path when the bundle is deployed.

> If the relative path refers to a file that does not exist in the bundle configuration directory, an error message is produced.

IBM 4690 absolute path in the format *<drive_or_logical_name>:<rest of path>*

> On UNIX, the absolute path is assumed to refer to an IBM 4690 file and is included in the bundle configuration.

> On Windows, if the absolute path is to a file that does not exist locally, it is assumed the path refers to an IBM 4690 file and is included in the bundle configuration. If the file does exist locally and is located under the bundle configuration directory, the path is converted to relative, and included in the configuration bundle.

> On Windows, if the absolute path is to a file that does exist locally but is not in the bundle configuration directory, it is assumed the path refers to a file that is not an IBM 4690 file. An error message is produced.

Syntax

fteBundleConfiguration

```
►►─fteBundleConfiguration─┬────┬─bundle.zip─directory──────────────────────────►◄
                          └─-x─┘
```
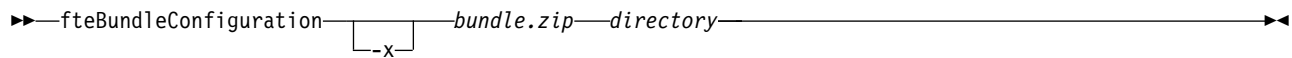
Parameters

bundle.zip
> Required. The path to the configuration bundle to process. Without the **-x** parameter *bundle.zip* is the file that is created.

directory
> Required. The path to the directory which the command operates on. Without the **-x** parameter, the *directory* is the source of the configuration to be included in the compressed file. When the **-x** parameter is specified, *directory* is the destination for the configuration that is extracted from the compressed file.

-x Optional. This parameter is used to unpackage configuration files from the *bundle.zip* file to the specified directory. When the parameter is not used, the *bundle.zip* file is created from the specified directory.

Example

In this example, the config.zip bundle is successfully created with the configuration in the QM_test directory:

```
fteBundleConfiguration C:\config.zip C:\WebSphereMQ\7500\mqft\config\QM_test
```

```
BFGCL0620I: The bundle 'C:\config.zip' has been successfully created from the configuration in
 directory  C:\WebSphereMQ\7500\mqft\config\QM_test'.
```

In this example, the config.zip bundle is successfully extracted to the config_extract directory:

```
fteBundleConfiguration -x C:\config.zip C:\Temp\config_extract
```

```
BFGCL0621I: The contents of the bundle 'C:\config.zip' have been successfully extracted to directory
 'C:\Temp\config_extract'.
```

In this example, the config.zip bundle creation fails because the contents of the agent properties file is invalid:

```
fteBundleConfiguration C:\config.zip C:\WebSphereMQ\7500\mqft\config\QM_test
```

```
BFGUB0039E: A required property of 'agentQMgr' is missing from the properties.
BFGCL0629E: The contents of the agent.properties file for agent 'FTEAGENT7777' are not valid.
 Configuration for this agent will not be included in the bundle.
BFGCL0628E: No valid agent configurations found in directory
 'C:\WebSphereMQ\7500\mqft\config\QM_test'. The new bundle cannot be created.
```

In this example, the config.zip bundle creation fails because the contents of the coordination properties file is invalid:

```
fteBundleConfiguration C:\config.zip C:\WebSphereMQ\7500\mqft\config\QM_test
```

```
BFGUB0022E: The property name "coordinationQMgrPort" has an invalid numeric value of "14B14"
BFGCL0631E: The contents of the coordination.properties file in directory
 'C:\WebSphereMQ\7500\mqft\config\QM_test' are not valid. The new bundle cannot be created.
```

In this example, the config.zip bundle is successfully created changing from a local absolute path to a relative path:

```
fteBundleConfiguration C:\config_ssl_paths.zip C:\WebSphereMQ\7500\mqft\config\QM_test
```

```
BFGCL0660I: The local absolute path 'C:\\WebSphereMQ\\7500\\mqft\\config\\ssl_path_config\\QM_gbthink\\mqmftcredentials.xml
 supplied for property 'agentSslKeyStoreCredentialsFile' was converted to relative path 'mqmftcredentials.xml'
 for inclusion in the bundle.
BFGCL0620I: The bundle 'C:\config_ssl_paths.zip' has been successfully created from the configuration in directory
 'C:\WebSphereMQ\7500\mqft\config\QM_test'.
```

In this example, the `config.zip` bundle is successfully created including a path that could not be determined:

`fteBundleConfiguration C:\config_ssl_paths.zip C:\WebSphereMQ\7500\mqft\config\QM_test`

```
BFGCL0662W: It could not be determined if the absolute path 'f:\\keystore.jks' referred to a local file or a 4690 file
 so it will be left unchanged.
BFGCL0620I: The bundle 'C:\config_ssl_paths.zip' has been successfully created from the configuration in directory
 'C:\WebSphereMQ\7500\mqft\config\QM_test'.
```

In this example, the `config.zip` bundle creation fails because the referenced local absolute agent property path is outside of the configuration directory:

`fteBundleConfiguration C:\config_ssl_paths.zip C:\WebSphereMQ\7500\mqft\config\QM_test`

```
BFGCL0659E: The local absolute path 'D:\\AGENTS\\mqmftcredentials.xml' supplied for property 'agentSslKeyStoreCredentialsFi
 is outside of the configuration directory being bundled.
```

In this example, the `config.zip` bundle creation fails because the referenced local absolute credential path is outside of the configuration directory:

`fteBundleConfiguration C:\config_ssl_paths.zip C:\WebSphereMQ\7500\mqft\config\QM_test`

```
BFGCL0663E: The local absolute path 'C:\keystore.jks' supplied in credentials file
 'C:\WebSphereMQ\7500\mqft\config\QM_test\agents\FTEAGENT7777\mqmftcredentials.xml' is outside of the configuration
 directory being bundled.
```

Related tasks:

"Creating a WebSphere MQ Managed File Transfer configuration on an IBM 4690 system" on page 68
To create or replace a WebSphere MQ Managed File Transfer configuration on an IBM 4690 system, you first create the configuration on a non-4690 platform. You then copy the configuration to the IBM 4690 system in a compressed file and run the **ftecfg** command to deploy the configuration to the IBM 4690 system.

Related reference:

"Structure of the IBM 4690 WebSphere MQ Managed File Transfer configuration compressed file" on page 69
You create a WebSphere MQ Managed File Transfer configuration on an IBM 4690 system by passing, as a parameter to the **ftecfg** command, a compressed file that contains the details of the configuration.

ftelap (accept the license agreement during WebSphere MQ Managed File Transfer installation)

The **ftelap** command runs the License Acceptance Process (LAP) tool. The LAP tool is used to read and accept the license that is associated with WebSphere MQ Managed File Transfer.

Purpose

Use the **ftelap** command to view and accept the license before using any WebSphere MQ Managed File Transfer function.

After you unpack the installation files, you must accept the product license before you can continue with the installation by going to the \adxetc\mft75\bin directory and running the **ftelap** command.

Syntax

ftelap

```
►►──ftelap──┬────────┬──────────────────────────────────────────►◄
            └─-accept─┘
```

Parameters

accept
> Optional. If you specify the **accept** parameter, the license is automatically accepted. If you do not specify the **accept** parameter, the license is displayed, which you must then accept or reject.

Related tasks:

"Installing IBM WebSphere MQ Managed File Transfer on 4690 OS" on page 62
Use the `MQMFT75.ZIP` file to install WebSphere MQ Managed File Transfer on 4690 OS.

ftecfg (creates a WebSphere MQ Managed File Transfer configuration on an IBM 4690 system)

The **ftecfg** command configures one or more WebSphere MQ Managed File Transfer agents on an IBM 4690 system.

Purpose

Use the **ftecfg** command to configure one or more WebSphere MQ Managed File Transfer agents on an IBM 4690 system. The command takes, as a parameter, the path to a compressed file that is created by using the **fteBundleConfiguration** command. The compressed file contains the configuration details of the agents that are to be installed, and generates a configuration directory tree that is populated directly from the compressed file. The **ftecfg** command validates the structure of the compressed file. For details of the compressed file structure, see "Structure of the IBM 4690 WebSphere MQ Managed File Transfer configuration compressed file" on page 69.

If a configuration directory exists, it is backed up by the **ftecfg** command before the new configuration is created. For more information, see "Backups" on page 95.

The **ftecfg** command generates the `config`, `installations`, and `logs` directories under the WebSphere MQ Managed File Transfer installation directory `f:\adxetc\mft75\mqft`.

The generated `installations` and `logs` directories are based on the coordination queue manager name and agent names that are specified in the compressed file. There is only one coordination queue manager defined. There can be one or more agents defined. The name of created installation is always `installation1`.

If you are using variable substitution for agent names (where the agent name AGENT@*N*@*S* expands to AGENTKD1234 for example), the agent directories that the **ftecfg** command creates use the fully expanded names that result from the substitution. After the directory structure is created on disk, the agentName property in the `agent.properties` file is updated to match the final substituted agent name. For more information, see "Customizing agent names in a 4690 OS configuration bundle" on page 69.

The generated output has the following directory structure:
```
mqmft (directory)

    AgentName1.rsp (file - generated by ftecfg)
    AgentName1.pc (file - generated by ftecfg)
    .
    .
    .
    (further files for agents if defined)
```

```
.
.
.
mqft (directory)

    config (directory)

      coordination-qmgr (directory)

            agents (directory)

                AgentName1 (directory - from the compressed file)
                    agent.properties (properties file - from the compressed file
                 .
                 .
                 .
                (further agents if defined)
                 .
                 .
                 .
            command.properties (properties file - from the compressed file)

            coordination.properties (properties file - from the compressed file)

    installations (directory)

        installation1 (directory)

            installation.properties (properties file)

    logs

            agents (directory)

                AgentName1 (directory - name taken from the compressed file)
                 .
                 .
                 .
                (further agents if defined)
                 .
                 .
                 .
```

The following example shows a sample of a generated directory structure:

```
mqmft
    MyFirstAgent.rsp
    MyFirstAgent.pc
    MySecondAgent.rsp
    MySecondAgent.pc
    mqft
        config
          QM_gbthink
                agents
                    MyFirstAgent
                        agent.properties
                    MySecondAgent
                        agent.properties
                command.properties
                coordination.properties
        installations
            installation1
                installation.properties
```

```
            logs
                agents
                    MyFirstAgent
                    MySecondAgent
```

Backups

Each time that you run the **ftecfg** command, a backup is taken of any existing configuration and log files
before a new configuration is extracted into the installation directory. This backup allows you to revert to
a previous configuration if you find a problem with the new configuration.

The configuration (in the `f:\adxetc\mft75\mqft\config` directory) is backed up to a file with the
following naming format:

`f:\adxetc\mft75\backup-`*timestamp*`-config.zip`

The log files (in the `f:\adxetc\mft75\mqft\logs` directory) are backed up to a file with the following
naming format:

`f:\adxetc\mft75\backup-`*timestamp*`-logs.zip`

For a pair of associated backup configuration and log files, the timestamp value matches, so you can
identify the previous configuration and log files.

A maximum of 10 backups are taken before the oldest backup is deleted when a new backup is created.
This limitation ensures that the amount of disk space that is used by backups on the system is restricted.
However, you are advised to monitor your backups to ensure that the disk does not fill up.

Syntax

ftecfg

```
►►──ftecfg──compressed_file_path──────────────────────────────────────────────►◄
```

Parameters

compressed_file_path
> Required. The path to the compressed `.zip` file that contains the agent configuration details.

Example

In this example, the command **ftecfg** runs successfully. The maximum number of backups is reached, so
the oldest backup files are deleted before the new backup files are created.

```
ftecfg config.zip

5724-H72 Copyright IBM Corp.  2008, 2012.  ALL RIGHTS RESERVED
BFGCL0643I: The maximum number of configuration backups has been reached. The oldest will be deleted.
BFGCL0644I: The backup file 'f:/adxetc/mft75/backup-20121122102214379-config.zip' has been deleted.
BFGCL0644I: The backup file 'f:/adxetc/mft75/backup-20121122102214379-logs.zip'has been deleted.
BFGCL0645I: A backup of the product configuration has been created in file 'f:/adxetc/mft75/backup-20121127104428148-con
BFGCL0646I: A backup of the product logs has been created in file 'f:/adxetc/mft75/backup-20121127104428148-logs.zip'.
BFGCL0610I: Agent FTEAGENT7777 successfully configured. ADXCHAIN.386 response file: f:\adxetc\mft75\FTEAGENT7777.rsp
```

In this example, the **ftecfg** command fails because the absolute path in `agent.properties` is not found.

```
ftecfg config.zip

BFGCL0645I: A backup of the product configuration has been created in file 'f:/adxetc/mft75/backup-20130129080041321-con
BFGCL0646I: A backup of the product logs has been created in file 'f:/adxetc/mft75/backup-20130129080041321-logs.zip'.
BFGCL0668E: The absolute path 'F:\\mqmftcredentials.xml' supplied for property 'agentSslKeyStoreCredentialsFile' does no
 to an existing local file.
BFGCL0635E: Failed to create the Managed File Transfer configuration.
```

Using WebSphere MQ Managed File Transfer in a retail environment **95**

In this example, the **ftecfg** command fails because the absolute path in the credentials file is not found.

```
ftecfg config.zip

BFGCL0645I: A backup of the product configuration has been created in file 'f:/adxetc/mft75/backup-20130129080733868-config
BFGCL0646I: A backup of the product logs has been created in file 'f:/adxetc/mft75/backup-20130129080733868-logs.zip'.
BFGCL0669E: The absolute path 'D:\truststore.jks' supplied in credentials file 'mqmftcredentials.xml' does not refer
 to an existing local file.
BFGCL0635E: Failed to create the Managed File Transfer configuration.
```

Related reference:

"Structure of the IBM 4690 WebSphere MQ Managed File Transfer configuration compressed file" on page 69
You create a WebSphere MQ Managed File Transfer configuration on an IBM 4690 system by passing, as a parameter to the **ftecfg** command, a compressed file that contains the details of the configuration.

"fteBundleConfiguration (create a WebSphere MQ Managed File Transfer IBM 4690 agent configuration .zip file)" on page 90
Use the **fteBundleConfiguration** command to bundle a WebSphere MQ Managed File Transfer configuration tree, from a specified directory, into a .zip file.

ftediag (generates WebSphere MQ Managed File Transfer agent diagnostic information about an IBM 4690 system)

The **ftediag** command generates diagnostic information for a WebSphere MQ Managed File Transfer agent on an IBM 4690 system.

Purpose

Use the **ftediag** command to enable, disable, and collect diagnostic trace information from a WebSphere MQ Managed File Transfer agent on an IBM 4690 system. The diagnostic information can include the status of the agent and can also generate a Javacore file. The command is valid even when the agent is not connected to its queue manager. The diagnostic information is output to the screen. If you specify the **-jc** parameter, the location of the Javacore file in your file system is also displayed on the screen.

Specify the optional **-p** parameter for this command only if you want to use a set of configuration options different from your default set. For more information, see "The agent.properties file" on page 548.

Syntax

ftediag
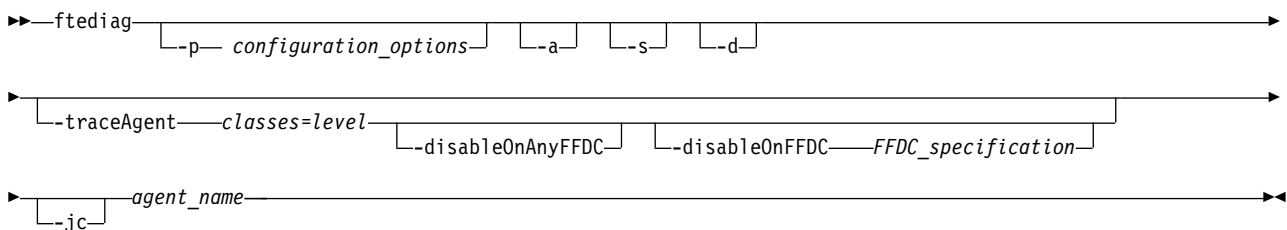
```
►►──ftediag──┬────────────────────────────┬──┬────┬──┬────┬──┬────┬──────────►
             └─-p── configuration_options─┘  └─-a─┘  └─-s─┘  └─-d─┘

►──┬───────────────────────────────────────────────────────────────────────────────┬──►
   └─-traceAgent──── classes=level ──┬──────────────────┬──┬───────────────────────────────────┬─┘
                                     └─-disableOnAnyFFDC─┘  └─-disableOnFFDC──── FFDC_specification ─┘

►──┬────┬──── agent_name ──────────────────────────────────────────────────────────►◄
   └─-jc─┘
```

Parameters

-p *configuration options*
> Optional. Specifies the set of configuration options that are used by the command. By convention this option is the name of a coordination queue manager. If you do not specify this parameter, the default configuration options are used.

-a Optional. Specifies that all diagnostic options, **-s**, **-d**, and **-jc** are selected.

-s Optional. Shows the status of the agent. This option is the default if no other option is specified.

-d Optional. Specifies that diagnostic information is displayed for the *agent_name*. Use this parameter when the agent is running, and is on the local system.

-traceAgent *classes=level*

Optional. Level to set the agent trace and which classes to apply the trace to. Specify the following format:

```
classes=level
```

For example:

```
com.ibm.wmqfte=all
```

Specify a comma-separated list of class specifications that you want the level of trace to apply to. If you do not specify this parameter, the trace level is applied to all agent classes.

If (*classes*) starts with a plus sign (+), the list of trace classes after the plus sign are added to any existing trace classes currently being traced.

The valid trace level options are as follows and are listed in ascending order of trace file size and detail:

off Switches off the agent trace but continues to write information to the log files.

flow Captures data for trace points associated with processing flow in the agent.

moderate
Captures a moderate amount of diagnostic information in the trace.

verbose
Captures a verbose amount of diagnostic information in the trace.

all Captures all diagnostic information in the trace.

-disableOnAnyFFDC

Optional. If this parameter is specified, trace is disabled on the agent when it generates a First Failure Data Capture (FFDC) file. This parameter is only valid if the **-traceAgent** parameter is also specified.

You can specify only one of the **-disableOnAnyFFDC** and **-disableOnFFDC** parameters.

-disableOnFFDC *FFDC_specification*

Optional. If this parameter is specified, trace is disabled on the agent when it generates a First Failure Data Capture (FFDC) file that matches the *FFDC_specification*. This parameter is only valid if the **-traceAgent** parameter is also specified. *FFDC_specification* is a comma-separated list of one or more of the following options:

class_name
The name of the class where the FFDC originated. For example, com.ibm.wmqfte.classA.

class_name:probe_ID
The name of the class and the probe ID of the location in the class that the FFDC originated from. For example, com.ibm.wmqfte.classB:1. For example:

```
-disableonFFDC com.ibm.wmqfte.transfer,com.ibm.wmqfte:1
```

You can specify only one of the **-disableOnAnyFFDC** and **-disableOnFFDC** parameters.

-jc

Optional. Requests that the agent generates a Javacore file. The IBM service team might request that you run the command with this parameter to assist with problem diagnosis. When you run the command with the **-jc** parameter, the location of the generated Javacore file is displayed on the screen.

agent_name
Required. The name of the WebSphere MQ Managed File Transfer agent that you want to extract diagnostic information from.

Example

In this example, only the Javacore parameter is used with the **ftediag** command for agent FTEAGENT1997:

```
ftediag -jc FTEAGENT1997

5724-H72 Copyright IBM Corp.  2008, 2012.  ALL RIGHTS RESERVED
BFGCL0549I: The javacore request was successfully sent to agent 'FTEAGENT1997'.
The created javacore file name is: /cdrive/f_drive/adxetc/java/core/javacore.201
20927.073416.31718.0001.txt
```

In this example, the agent trace level for class com.ibm.wmqfte is set to capture all diagnostic information in the trace, and a diagnostic and a Javacore file are created for the agent FTEAGENT1997.

```
ftediag -d -jc -traceAgent com.ibm.wmqfte=all FTEAGENT1997

5724-H72 Copyright IBM Corp.  2008, 2012.  ALL RIGHTS RESERVED
BFGCL0549I: The javacore request was successfully sent to agent 'FTEAGENT1997'.
The created javacore file name is: /cdrive/f_drive/adxetc/mqmft/mqft/logs/MUNGEE
/agents/FTEAGENT1997/javacore.20121101.123902.5728.0001.txt
Agent Information:
    Name:                          FTEAGENT1997
    Type:                          Standard
    Description:
    Operating System:              4690 OS
    Host Name:                     KD
    Time Zone:                     Greenwich Mean Time
    Product Version:               7.5.0.2
    Build Level:                   f000-personal-20121031-0905
    Trace Level:                   No trace specified
    Trace FFDC:                    No FFDC specified

Agent Controller Information:
    Status:                        STARTED_BY_AGENT
    Status Details:                The agent has started the process
                                   controller.
    Agent Restarts within Interval:  0
    Total Agent Restart Count:     0

Agent Availability Information:
    Status:                        READY
    Status Details:                The agent is running and is publishing
                                   its status at regular intervals. The
                                   last update was received within the
                                   expected time period. The agent is ready
                                   to process transfers, but none are
                                   currently in progress.

Queue Manager Information:
    Name:                          MUNGEE
    Transport:                     Client
    Host:                          192.168.255.1
    Port:                          1414
    Channel:                       SYSTEM.DEF.SVRCONN
    Last Status Reported:          UNKNOWN
    Status Details:                Information about the queue manager is
                                   not available because the agent has a
                                   client connection to the queue  manager.

Maximum Number of Running Source Transfers: 25
Maximum Number of Queued Source Transfers: 1000
Source Transfer States:
    No current transfers

Maximum Number of Running Destination Transfers: 25
Destination Transfer States:
    No current transfers
```

```
Agent Diagnostic Information:
    Diagnostic Properties File name:   f:\adxetc\mft75\mqft\logs\MUNGEE\agents\F
                                       TEAGENT1997\logs\diagnostics.20121101.123
                                       904.0909.1.properties

Command Handler Diagnostics:
    Last Command Queue Read Time:      2012-11-01T12:38:41.286Z
    Pending Command Queue Size:        0

Command Handler Worker Thread 0 Diagnostics:
    Status:                            Waiting

Command Handler Worker Thread 1 Diagnostics:
    Status:                            Waiting

Command Handler Worker Thread 2 Diagnostics:
    Status:                            Waiting

Command Handler Worker Thread 3 Diagnostics:
    Status:                            Waiting

Command Handler Worker Thread 4 Diagnostics:
    Status:                            Waiting

File Transfer Diagnostics:
    Source Transfers:                  0
    Destination Transfers:             0
```

In this example, the -a parameter is used with the **ftediag** command to select status information, a diagnostic file, and a Javacore file for agent FTEAGENT1997:

```
ftediag -a FTEAGENT1997

5724-H72 Copyright IBM Corp.  2008, 2012.  ALL RIGHTS RESERVED
BFGCL0549I: The javacore request was successfully sent to agent 'FTEAGENT1997'.
The created javacore file name is: /cdrive/f_drive/adxetc/java/core/javacore.201
20927.073454.31718.0002.txt
BFGCL0598I: The diagnostics request was successfully sent to agent 'FTEAGENT1997
'. The created diagnostics file name is: /cdrive/f_drive/adxetc/mqmft/mqft/logs/
MUNGEE/agents/FTEAGENT1997/logs/diagnostics.20120927.073454.0859.0.properties
Agent Information:
    Name:                              FTEAGENT1997
    Type:                              Standard
    Description:
    Operating System:                  4690 OS
    Host Name:                         KD
    Time Zone:                         Greenwich Mean Time
    Product Version:                   7.5.0.2
    Build Level:                       f000-personal-20120925-1131
    Trace Level:                       No trace specified
    Trace FFDC:                        No FFDC specified

Agent Controller Information:
    Status:                            STARTED
    Status Details:                    The agent process controller has started
                                       the agent process.
    Agent Restarts within Interval:    0
    Total Agent Restart Count:         0

Agent Availability Information:
    Status:                            READY
    Status Details:                    The agent is running and is publishing
                                       its status at regular intervals. The last
                                       update was received within the expected
                                       time period. The agent is ready to
                                       process transfers, but none are currently
                                       in progress.
```

```
Queue Manager Information:
    Name:                        MUNGEE
    Transport:                   Client
    Host:                        192.168.255.1
    Port:                        1414
    Channel:                     SYSTEM.DEF.SVRCONN
    Last Status Reported:        UNKNOWN
    Status Details:              Information about the queue manager is
                                 not available because the agent has a
                                 client connection to the queue  manager.

Maximum Number of Running Source Transfers: 25
Maximum Number of Queued Source Transfers: 1000
Source Transfer States:
    No current transfers

Maximum Number of Running Destination Transfers: 25
Destination Transfer States:
    No current transfers
```

uninstall (uninstall WebSphere MQ Managed File Transfer from an IBM 4690 system)

The **uninstall** command uninstalls WebSphere MQ Managed File Transfer from an IBM 4690 store controller.

Purpose

Use the **uninstall** command to uninstall WebSphere MQ Managed File Transfer product files from an IBM 4690 system. Optionally, you can also you this command to uninstall configuration and log files.

Syntax

uninstall

```
►►─uninstall─┬─ -a ─┬────────────────────────────────────────────────►◄
             └─ -c ─┘
```

Parameters

-a   Required: you must specify either the **-a** parameter or the **-c** parameter. Uninstalls all the WebSphere MQ Managed File Transfer product files, configuration files, and log files from the system.

-c   Required: you must specify either the **-c** parameter or the **-a** parameter. Uninstalls all the WebSphere MQ Managed File Transfer product files, but does not uninstall the configuration files or log files.

Example

In this example, all WebSphere MQ Managed File Transfer product files are uninstalled, but the configuration files and log files created are retained.

```
f:
cd \adxetc\mft75
uninstall -c
```

Related tasks:

"Uninstalling WebSphere MQ Managed File Transfer from a 4690 system" on page 67
To uninstall the WebSphere MQ Managed File Transfer agent from a 4690 store controller, complete the following steps:

Troubleshooting the IBM 4690 system

Use the following reference information to help you diagnose errors returned from the IBM 4690 system.

- "Troubleshooting IBM 4690 program calls"

Troubleshooting IBM 4690 program calls

Transfer program calls on an IBM 4690 system can be run in foreground (call type: EXECUTABLE), or in background (call type: OS4690BACKGROUND). These call types support the running of native IBM 4690 applications and batch command scripts. If you run a batch script the COMMAND.286 application is used with the -C option as the application, passing the batch script and any specified arguments as the parameters.

IBM 4690 application failure

If the start of an application fails for any reason, the command call result indicates an error and gives the reason why. If the application is a batch script, and the script is malformed, it may be reported as successfully ran even though it failed. In the following example, a batch script incorrectly uses a semicolon (;) as a command separator:

```
echo "First Echo" > stdout1.txt ; echo "Second Echo" > stdout2.txt
```

If this batch script is run as a WebSphere MQ Managed File Transfer program call, the result is reported as successful. However, the following error text for the call is in the transfer log:

```
STDOUT:  The STDOUT parameter was previously redirected.
STDOUT can only be redirected to one place at a time.
```

This is because the COMMAND.286 application that is running the batch script returns a success (0) exit code for this scenario. For this reason, it is important to validate any batch scripts to be run as program calls before you use them. Also, if a batch script runs in the foreground it should invoke the EXIT command with an appropriate code, so the transfer status reflects the outcome of the batch script call.

Security overview for WebSphere MQ Managed File Transfer

Directly after installation and with no modification, WebSphere MQ Managed File Transfer has a level of
security that might be suitable for test or evaluation purposes in a protected environment. However, in a
production environment, you must consider appropriately controlling who can start file transfer
operations, who can read and write the files being transferred, and how to protect the integrity of files.

Related concepts:

"Sandboxes"
You can restrict the area of the file system that the agent can access as part of a transfer. The area that the
agent is restricted to is called the sandbox. You can apply restrictions to either the agent or to the user
that requests a transfer.

"Securing the Web Gateway" on page 111
There are a number of ways that you can secure the Web Gateway. You must perform some of these
security steps before you can use the Web Gateway. The other steps are optional and can increase the
security of your Web Gateway and WebSphere MQ Managed File Transfer network, but they are not
required for you to use the Web Gateway.

Related tasks:

"Configuring SSL encryption for WebSphere MQ Managed File Transfer" on page 109
Use SSL with WebSphere MQ and WebSphere MQ Managed File Transfer to prevent unauthorized
connections between agents and queue managers, and to encrypt message traffic between agents and
queue managers.

Related reference:

"Group authorities for resources specific to WebSphere MQ Managed File Transfer" on page 405
Instead of granting authority to individual users for all of the various objects that might be involved,
configure two security groups for the purposes of administering WebSphere MQ Managed File Transfer
access control: FTEUSER and FTEAGENT. It is the responsibility of the WebSphere MQ administrator to
create and populate these groups. The administrator can choose to extend or modify the proposed
configuration described here.

"Authorities for resources specific to WebSphere MQ Managed File Transfer" on page 404
For any file transfer request, the agent processes require some level of access to their local file systems. In
addition, both the user identifier associated with the agent process, and the user identifiers associated
with users performing file transfer operations must have the authority to use certain WebSphere MQ
objects.

"Authorities to access file systems" on page 414
For any file transfer request, the agent processes require some level of access to their local file systems.

"The commandPath property" on page 414
Use the commandPath property to restrict the locations that WebSphere MQ Managed File Transfer can
run commands from.

"Authority to publish log and status messages" on page 413
Agents issue various log, progress, and status messages that are published on the coordination queue
manager. The publication of these messages is subject to the WebSphere MQ security model, and in some
cases you might have to perform further configuration to enable publication.

Sandboxes

You can restrict the area of the file system that the agent can access as part of a transfer. The area that the
agent is restricted to is called the sandbox. You can apply restrictions to either the agent or to the user
that requests a transfer.

Sandboxes are not supported when the agent is a protocol bridge agent or a Connect:Direct bridge agent. You can not use agent sandboxing for agents that need to transfer to or from WebSphere MQ queues.

Related reference:

"Working with agent sandboxes"
To add an additional level of security to WebSphere MQ Managed File Transfer, you can restrict the area of a file system that an agent can access.

"Working with user sandboxes" on page 105
You can restrict the area of the file system that files can be transferred into and out of based on the MQMD user name that requests the transfer.

Working with agent sandboxes

To add an additional level of security to WebSphere MQ Managed File Transfer, you can restrict the area of a file system that an agent can access.

You cannot use agent sandboxing for agents that transfer to or from WebSphere MQ queues. Restricting access to WebSphere MQ queues with sandboxing can be implemented instead by using user sandboxing which is the recommended solution for any sandboxing requirements. For more information about user sandboxing, see "Working with user sandboxes" on page 105

To enable agent sandboxing, add the following property to the `agent.properties` file for the agent you want to restrict:

`sandboxRoot=[!]`*restricted_directory_name*`<separator>`...`<separator>[!]`*restricted_directory_name*

where:

- *restricted_directory_name* is a directory path to be allowed or denied.
- ! is optional and specifies that the following value for *restricted_directory_name* is denied (excluded). If ! is not specified *restricted_directory_name* is an allowed (included) path.
- *<separator>* is the platform-specific separator.

For example, if you want to restrict the access that AGENT1 has to the /tmp directory only, but not allow the subdirectory `private` to be accessed, set the property as follows in the `agent.properties` file belonging to AGENT1: `sandboxRoot=/tmp:!/tmp/private`.

The sandboxRoot property is described in Advanced agent properties.

Both agent and user sandboxing are not supported on protocol bridge agents or on Connect:Direct bridge agents.

Working in a sandbox on UNIX, Linux, and Windows platforms

On UNIX, Linux, and Windows platforms, sandboxing restricts which directories a WebSphere MQ Managed File Transfer agent can read from and write to. When sandboxing is activated, the WebSphere MQ Managed File Transfer agent can read and write to the directories specified as allowed, and any subdirectories that the specified directories contain unless the subdirectories are specified as denied in the sandboxRoot. WebSphere MQ Managed File Transfer sandboxing does not take precedence over operating system security. The user that started the WebSphere MQ Managed File Transfer agent must have the appropriate operating system level access to any directory to be able to read from or write to the directory. A symbolic link to a directory is not followed if the directory linked to is outside the specified sandboxRoot directories (and subdirectories).

Working in a sandbox on IBM 4690 systems

For information about how paths specified in the sandboxRoot property are interpreted on IBM 4690, see "Working in a sandbox on IBM 4690" on page 89.

Working with user sandboxes

You can restrict the area of the file system that files can be transferred into and out of based on the MQMD user name that requests the transfer.

User sandboxes are not supported when the agent is a protocol bridge agent or a Connect:Direct bridge agent.

To enable user sandboxing, add the following property to the `agent.properties` file for the agent that you want to restrict:

`userSandboxes=true`

When this property is present and set to true the agent uses the information in the *MQ_DATA_PATH*/mqft/config/*coordination_qmgr_name*/agents/*agent_name*/UserSandboxes.xml file to determine which parts of the file system the user who requests the transfer can access.

The `UserSandboxes.xml` XML is composed of an `<agent>` element that contains zero or more `<sandbox>` elements. These elements describe which rules are applied to which users. The `user` attribute of the `<sandbox>` element is a pattern that is used to match against the MQMD user of the request.

The file `UserSandboxes.xml` is periodically reloaded by the agent and any valid changes to the file will affect the behavior of the agent. The default reload interval is 30 seconds. This interval can be changed by specifying the agent property xmlConfigReloadInterval in the `agent.properties` file.

If you specify the `userPattern="regex"` attribute or value, the `user` attribute is interpreted as a Java regular expression. For more information, see "Regular expressions used by WebSphere MQ Managed File Transfer" on page 685.

If you do not specify the `userPattern="regex"` attribute or value the `user` attribute is interpreted as a pattern with the following wildcard characters:
- asterisk (*), which represents zero or more characters
- question mark (?), which represents exactly one character

Matches are performed in the order that the `<sandbox>` elements are listed in the file. Only the first match is used, all following potential matches in the file are ignored. If none of the `<sandbox>` elements specified in the file match the MQMD user associated with the transfer request message, the transfer cannot access the file system. When a match has been found between the MQMD user name and a `user` attribute, the match identifies a set of rules inside a `<sandbox>` element that are applied to the transfer. This set of rules is used to determine which files can be read from or written to as part of the transfer.

Each set of rules can specify a `<read>` element, which identifies which files can be read, and a `<write>` element which identifies which files can be written. If you omit the `<read>` or `<write>` elements from a set of rules, it is assumed that the user associated with that set of rules is not allowed to perform any reads or any writes, as appropriate.

Note: The `<read>` element must be before the `<write>` element, and the `<include>` element must be before the `<exclude>` element, in the `UserSandboxes.xml` file.

Each `<read>` or `<write>` element contains one or more patterns that are used to determine whether a file is in the sandbox and can be transferred. Specify these patterns by using the `<include>` and `<exclude>` elements. The `name` attribute of the `<include>` or `<exclude>` element specifies the pattern to be matched. An optional `type` attribute specifies whether the name value is a file or queue pattern. If the `type` attribute is not specified, the agent treats the pattern as a file or directory path pattern. For example:

```
<tns:read>
  <tns:include name="/home/user/**"/>
  <tns:include name="USER.**" type="queue"/>
  <tns:exclude name="/home/user/private/**"/>
</tns:read>
```

The `<include>` and `<exclude>` name patterns are used by the agent to determine whether files or queues can be read from or written to. An operation is allowed if the canonical file path or queue name matches at least one of the included patterns and exactly zero of the excluded patterns. The patterns specified by using the `name` attribute of the `<include>` and `<exclude>` elements use the path separators and conventions appropriate to the platform that the agent is running on. If you specify relative file paths, the paths are resolved relative to the `transferRoot` property of the agent.

When specifying a queue restriction, a syntax of QUEUE@QUEUEMANAGER is supported, with the following rules:

- If the at character (@) is missing from the entry, the pattern is treated as a queue name that can be accessed on any queue manager. For example, if the pattern is `name` it is treated the same way as `name@**`.
- If the at character (@) is the first character in the entry, the pattern is treated as a queue manager name and all queues on the queue manager can be accessed. For example, if the pattern is `@name` it is treated the same way as `**@name.`.

The following wildcard characters have special meaning when you specify them as part of the `name` attribute of the `<include>` and `<exclude>` elements:

*	A single asterisk matches zero or more characters in a directory name, or in a qualifier of a queue name.
?	A question mark matches exactly one character in a directory name, or in a qualifier of a queue name.
**	Two asterisk characters match zero or more directory names, or zero or more qualifiers in a queue name. Also, paths that end with a path separator have an implicit "*" added to the end of the path. So /home/user/ is the same as /home/user/**.

For example:

- `/**/test/**` matches any file that has a `test` directory in its path
- `/test/file?` matches any file inside the `/test` directory that starts with the string `file` followed by any single character
- `c:\test\*.txt` matches any file inside the `c:\test` directory with a `.txt` extension
- `c:\test\**\*.txt` matches any file inside the 'c:\test directory, or one of its subdirectories that has a `.txt` extension
- `TEST.*.QUEUE@QM1` matches any queue on the queue manager `QM1` that has the first qualifier of `TEST`, has any second qualifier, and a third qualifier of `QUEUE`.

Symbolic links

You must fully resolve any symbolic links that you use in file paths in the `UserSandboxes.xml` file by specifying hard links in the `<include>` and `<exclude>` elements. For example, if you have a symbolic link where /var maps to /SYSTEM/var, you must specify this path as `<tns:include name="/SYSTEM/var"/>`, otherwise the intended transfer fails with a user sandbox security error.

Paths on IBM 4690 systems

For information about how paths specified in the `UserSandboxes.xml` file are interpreted on IBM 4690, see "Working in a sandbox on IBM 4690" on page 89.

Example

To allow the user with the MQMD user name `guest` to transfer any file from the `/home/user/public` directory or any of its subdirectories on the system where the agent AGENT_JUPITER is running, add the following `<sandbox>` element to the file `UserSandboxes.xml` in AGENT_JUPITER's configuration directory

```
<?xml version="1.0" encoding="UTF-8"?>
<tns:userSandboxes
   xmlns:tns="http://wmqfte.ibm.com/UserSandboxes"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://wmqfte.ibm.com/UserSandboxes UserSandboxes.xsd">
 <tns:agent>
  <tns:sandbox user="guest">
   <tns:read>
    <tns:include name="/home/user/public/**"/>
    </tns:read>
  </tns:sandbox>
 </tns:agent>
</tns:userSandboxes>
```

Example

To allow any user with the MQMD user name `account` followed by a single digit, for example `account4`, to complete the following actions:

- Transfer any file from the `/home/account` directory or any of its subdirectories, excluding the `/home/account/private` directory on the system where the agent AGENT_SATURN is running

- Transfer any file to the `/home/account/output` directory or any of its subdirectories on the system where the agent AGENT_SATURN is running

- Read messages from queues on the local queue manager starting with the prefix `ACCOUNT.` unless it starts with `ACCOUNT.PRIVATE.` (that is has `PRIVATE` at the second level).

- Transfer data onto queues starting with the prefix `ACCOUNT.OUTPUT.` on any queue manager.

add the following `<sandbox>` element to the file `UserSandboxes.xml`, in AGENT_SATURN's configuration directory,

```
<?xml version="1.0" encoding="UTF-8"?>
<tns:userSandboxes
    xmlns:tns="http://wmqfte.ibm.com/UserSandboxes"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://wmqfte.ibm.com/UserSandboxes UserSandboxes.xsd">
 <tns:agent>
  <tns:sandbox user="account[0-9]" userPattern="regex">
   <tns:read>
    <tns:include name="/home/account/**"/>
    <tns:include name="ACCOUNT.**" type="queue"/>
    <tns:exclude name="ACCOUNT.PRIVATE.**" type="queue"/>
    <tns:exclude name="/home/account/private/**"/>
                              </tns:read>
   <tns:write>
    <tns:include name="/home/account/output/**"/>
    <tns:include name="ACCOUNT.OUTPUT.**" type="queue"/>
   </tns:write>
  </tns:sandbox>
 </tns:agent>
</tns:userSandboxes>
```

Additional checks for wildcard transfers

From IBM WebSphere MQ Version 7.5.0, Fix Pack 8, if an agent has been configured with a user or agent sandbox in order to restrict the locations that the agent can transfer files to and from, you can specify that additional checks are to be made on wildcard transfers for that agent.

additionalWildcardSandboxChecking property

To enable additional checking for wildcard transfers, add the following property to the `agent.properties` file for the agent that you want to check.

`additionalWildcardSandboxChecking=true`

When this property is set to true, and the agent makes a transfer request that attempts to read a location that is outside the defined sandbox for file matching of the wildcard, the transfer fails. If there are multiple transfers within one transfer request, and one of these requests fails due to it attempting to read a location outside of the sandbox, the entire transfer fails. If checking fails, the reason for failure is given in an error message.

If the additionalWildcardSandboxChecking property is omitted from an agent's `agent.properties` file or is set to false, no additional checks are made on wildcard transfers for that agent.

Error messages for wildcard checking

From Version 7.5.0, Fix Pack 8, the messages that are reported when a wildcard transfer request is made to a location outside a configured sandbox location have changed.

The following message occurs when a wildcard file path in a transfer request is located outside of the restricted sandbox:

```
BFGSS0077E: Attempt to read file path: <path> has been denied.
The file path is located outside of the restricted transfer sandbox.
```

The following message occurs when a transfer within a multiple transfer request contains a wildcard transfer request where the path is located outside of the restricted sandbox:

```
BFGSS0078E: Attempt to read file path: <path> has been ignored as another transfer
item in the managed transfer attempted to read outside of the restricted transfer sandbox.
```

The following message occurs when a file is located outside of the restricted sandbox:

```
BFGSS0079E: Attempt to read file <file path> has been denied.
The file is located outside of the restricted transfer sandbox.
```

The following message occurs in a multiple transfer request where another wildcard transfer request has caused this one to be ignored:

```
BFGSS0080E: Attempt to read file: <file path> has been ignored as another transfer
item in the managed transfer attempted to read outside of the restricted transfer sandbox.
```

In the case of single file transfers that do not include wildcards, the message that is reported when the transfer involves a file that is located outside of the sandbox is unchanged from earlier releases:

```
Fails with BFGIO0056E: Attempt to read file "<FILE>" has been denied.
The file is located outside of the restricted transfer sandbox.
```

Related reference:

"Working with user sandboxes" on page 105
You can restrict the area of the file system that files can be transferred into and out of based on the MQMD user name that requests the transfer.

"Working with agent sandboxes" on page 104
To add an additional level of security to WebSphere MQ Managed File Transfer, you can restrict the area of a file system that an agent can access.

"The agent.properties file" on page 548
Each agent has its own properties file, `agent.properties`, that must contain the information that an agent uses to connect to its queue manager. The `agent.properties` file can also contain properties that alter the behavior of the agent.

Configuring SSL encryption for WebSphere MQ Managed File Transfer

Use SSL with WebSphere MQ and WebSphere MQ Managed File Transfer to prevent unauthorized connections between agents and queue managers, and to encrypt message traffic between agents and queue managers.

Before you begin

SSL encryption encrypts messages only on the channels between queue managers, and between queue managers and agents.

About this task

For general information about using SSL with WebSphere MQ, see the WebSphere MQ product documentation. In WebSphere MQ terms, WebSphere MQ Managed File Transfer is a standard Java client application.

Follow these steps to use SSL with WebSphere MQ Managed File Transfer:

Procedure

1. Create a truststore file and optionally a keystore file (these files can be the same file). If you do not need client-authentication (that is, SSLCAUTH=OPTIONAL on channels) you do not need to provide a keystore. You require a truststore only to authenticate the queue manager's certificate against.

   The key algorithm of the truststore file and keystore file must be RSA to work with WebSphere MQ.

   If you need instructions about how to create truststore and keystore files, see the developerWorks® article, Configuring Secure Sockets Layer connectivity in WebSphere MQ File Transfer Edition, or see the information about the keytool at the Oracle keytool documentation.

2. Set up your WebSphere MQ queue manager to use SSL. For information about setting up a queue manager to use SSL using WebSphere MQ Explorer for example, see Configuring SSL on queue managers.

3. Save the truststore file and keystore file (if you have one) in a suitable location. A suggested location is the *config_directory/coordination_qmgr*/agents/*agent_name* directory.

4. Set the SSL properties as required for each SSL-enabled queue manager in the appropriate WebSphere MQ Managed File Transfer properties file. Each set of properties refers to a separate queue manager (agent, coordination, and command), although one queue manager might perform two or more of these roles.

   One of the **CipherSpec** or **CipherSuite** properties is required, otherwise the client tries to connect without SSL. Both the **CipherSpec** or **CipherSuite** properties are provided because of the terminology differences between WebSphere MQ and Java. WebSphere MQ Managed File Transfer accepts either property and does the necessary conversion, so you do not need to set both properties. If you do specify both the **CipherSpec** or **CipherSuite** properties, **CipherSpec** takes precedence.

The **PeerName** property is optional. You can set the property to the Distinguished Name of the queue manager that you want to connect to. WebSphere MQ Managed File Transfer rejects connections to an incorrect SSL server with a Distinguished Name that does not match.

Set the **SslTrustStore** and **SslKeyStore** properties to file names that point to the truststore and keystore files. If you are setting up these properties for an agent that is already running, stop and restart the agent to reconnect in SSL mode.

Properties files contain plain-text passwords so consider setting appropriate file system permissions.

For more information about SSL properties, see "SSL properties" on page 598.

5. If an agent queue manager uses SSL, you cannot provide the necessary details when you create the agent. Use the following steps to create the agent:

   a. Create the agent by using the **fteCreateAgent** command. You receive a warning about being unable to publish the existence of the agent to the coordination queue manager.

   b. Edit the `agent.properties` file that was created by the previous step to add the SSL information. When the agent is successfully started, the publish is attempted again.

6. If agents or instances of the WebSphere MQ Explorer are running while the SSL properties in the `agent.properties` file or `coordination.properties` file are changed, you must restart the agent or WebSphere MQ Explorer.

Related reference:

"The agent.properties file" on page 548
Each agent has its own properties file, `agent.properties`, that must contain the information that an agent uses to connect to its queue manager. The `agent.properties` file can also contain properties that alter the behavior of the agent.

Connecting to a IBM WebSphere MQ V7.1 or later queue manager in client mode with channel authentication

IBM WebSphere MQ V7.1 introduced channel authentication records to control more precisely access at a channel level. This change in behavior means that by default newly created IBM WebSphere MQ V7.1 or later queue managers reject client connections from the Managed File Transfer component.

For more information about channel authentication, see Channel authentication records.

If the channel authentication configuration for the SVRCONN used by WebSphere MQ Managed File Transfer specifies a non-privileged MCAUSER ID, you must grant specific authority records for the queue manager, queues, and topics, to allow the Managed File Transfer agent and commands to work correctly. Use the MQSC command SET CHLAUTH or the PCF command Set Channel Authentication Record to create, modify, or remove channel authentication records. For all Managed File Transfer agents that you want to connect to the V7.1 or later queue manager, you can either set up an MCAUSER ID to use for all your agents, or set up a separate MCAUSER ID for each agent.

Grant each MCAUSER ID the following permissions:
- Authority records required for the queue manager:
  - connect
  - setid
  - inq
- Authority records required for queues.

  For all agent-specific queues, that is queue names that end in *agent_name* in the following list, you must create these queue authority records for each agent that you want to connect to the IBM WebSphere MQ V7.1 or later queue manager by using a client connection.
  - put, get, dsp (SYSTEM.DEFAULT.MODEL.QUEUE)
  - put, get, setid, browse (SYSTEM.FTE.COMMAND.*agent_name*)

- put, get (SYSTEM.FTE.DATA.*agent_name*)
- put, get (SYSTEM.FTE.REPLY.*agent_name*)
- put, get, inq, browse (SYSTEM.FTE.STATE.*agent_name*)
- put, get, browse (SYSTEM.FTE.EVENT.*agent_name*)
- put, get (SYSTEM.FTE)
- Authority records required for topics:
  - sub, pub (SYSTEM.FTE)
- Authority records required for file transfers.

  If you have separate MCAUSER IDs for source and destination agent, create the authority records on agents' queues at both source and destination.For example, if the source agent's MCAUSER ID is **user1** and the destination agent MCAUSER ID is **user2**, set the following authorities for the agent users:

AGENT user	Queue	Authority required
user1	SYSTEM.FTE.DATA.*destination_agent_name*	put
user1	SYSTEM.FTE.COMMAND.*destination_agent_name*	put
user2	SYSTEM.FTE.REPLY.*source_agent_name*	put
user2	SYSTEM.FTE.COMMAND.*source_agent_name*	put

Securing the Web Gateway

There are a number of ways that you can secure the Web Gateway. You must perform some of these security steps before you can use the Web Gateway. The other steps are optional and can increase the security of your Web Gateway and WebSphere MQ Managed File Transfer network, but they are not required for you to use the Web Gateway.

Related concepts:

"Required security for the Web Gateway"
There are security configuration steps that you must complete before you can use the Web Gateway. These steps are configuring user roles for the Web Gateway, setting file space permissions, and, if you are using WebSphere Application Server Version 7.0, setting the correct level of security in the application server.

"Optional security for the Web Gateway" on page 114
There are security configuration steps that are not required before you can use the Web Gateway. These optional steps can add extra security to your Web Gateway and your WebSphere MQ Managed File Transfer network. The optional steps are filtering Web Gateway requests and enabling sandboxing on destination agents.

Required security for the Web Gateway

There are security configuration steps that you must complete before you can use the Web Gateway. These steps are configuring user roles for the Web Gateway, setting file space permissions, and, if you are using WebSphere Application Server Version 7.0, setting the correct level of security in the application server.

WebSphere MQ Managed File Transfer has two stages of authorization: user roles and file space permissions. To upload a file or to query transfer information, the user must have the appropriate user role assigned to them. To access a file space the user must have both the appropriate user role assigned to them and have the appropriate level of permission for the file space that they are trying to access.

Application server security

If you are deploying the Web Gateway in WebSphere Application Server Version 7.0, use the **Global security** panel to enable the correct level of security. Select **Enable administrative security** and **Enable**

application security. Ensure that **Use Java 2 security to restrict application access to local resources** is not selected.

User roles for Web Gateway

Web Gateway users must have one or more roles assigned before they can use the Web Gateway. When deploying the Web Gateway to an application server these roles can be mapped to users and groups that exist in that application server.

WebSphere MQ Managed File Transfer defines the following roles:
- wmqfte-agent-upload
- wmqfte-filespace-user
- wmqfte-filespace-create
- wmqfte-filespace-modify
- wmqfte-filespace-permissions
- wmqfte-filespace-delete
- wmqfte-audit
- wmqfte-admin

For more information about these roles, see "User roles for the Web Gateway" on page 113.

For example, if your application server defines the groups 'Employees', 'Managers' and 'Administrators', the roles could be assigned to the groups as shown:

Employees
> wmqfte-agent-upload

> wmqfte-filespace-user

Managers
> wmqfte-filespace-create

> wmqfte-filespace-modify

> wmqfte-filespace-permissions

Administrators
> wmqfte-admin

In this example, only users in the Administrators group can delete file spaces.

File space permissions

A Web Gateway user can access a file space if they are the owner of the file space, or if they have been given explicit permission to access the file space. When you create a file space you can specify lists of authorized or unauthorized user names, or Java regular expressions to match user names. Users that are in the authorized list can download from and upload to the file space. Users that are in the unauthorized list cannot access the file space, even if they are also in the authorized list, or match a regular expression in the authorized list. For more information, see "Example: Creating a file space" on page 304.

Related concepts:

"Securing the Web Gateway" on page 111

There are a number of ways that you can secure the Web Gateway. You must perform some of these security steps before you can use the Web Gateway. The other steps are optional and can increase the security of your Web Gateway and WebSphere MQ Managed File Transfer network, but they are not required for you to use the Web Gateway.

"Optional security for the Web Gateway" on page 114

There are security configuration steps that are not required before you can use the Web Gateway. These optional steps can add extra security to your Web Gateway and your WebSphere MQ Managed File Transfer network. The optional steps are filtering Web Gateway requests and enabling sandboxing on destination agents.

Related reference:

"User roles for the Web Gateway"

WebSphere MQ Managed File Transfer has defined several different roles that control the actions a user can take.

User roles for the Web Gateway

WebSphere MQ Managed File Transfer has defined several different roles that control the actions a user can take.

You configure these roles on your application server, either before deploying the Web Gateway, or during deployment. For information about how to configure WebSphere Application Server Community Edition, including how to set up security roles, see "Preparing to deploy the Web Gateway with WebSphere Application Server Community Edition" on page 167. For information about how to deploy the Web Gateway on WebSphere Application Server Version 7.0, including how to set up security roles, see "Deploying the Web Gateway with WebSphere Application Server Version 7.0" on page 184.

The following table lists the different roles and the level of access associated with each role:

Table 7. Roles and associated permissions

Role	Description
wmqfte-agent-upload	User can upload files to an agent
wmqfte-filespace-user	User can list contents of their own file space User can download from their own file space User can delete files from their own file space
wmqfte-filespace-create	User can create a file space, if a file space of that name does not already exist
wmqfte-filespace-modify	User can modify the properties of a file space
wmqfte-filespace-permissions	User can modify the permissions [(1)] of a file space
wmqfte-filespace-delete	User can delete a file space
wmqfte-audit	User can view information in the audit database. **Note:** Users who are not associated with this role can view audit information for the following transfers only: • Uploads that are initiated by the user • Transfers to a file space that is owned by the user
wmqfte-admin	User can perform the actions associated with all roles, except one: • User cannot receive the contents of a file that the user deletes from a file space
(1) Permissions can be set on individual file spaces. For more information see the topics "Web Gateway administration API reference" on page 885 and "Example: Modifying file space configuration" on page 306.	

Optional security for the Web Gateway

There are security configuration steps that are not required before you can use the Web Gateway. These optional steps can add extra security to your Web Gateway and your WebSphere MQ Managed File Transfer network. The optional steps are filtering Web Gateway requests and enabling sandboxing on destination agents.

Filtering Web Gateway requests

As a Web Gateway administrator (with a wmqfte-admin role), you can filter HTTP requests to the Web Gateway by using the servlet filtering functions that are provided by your application server. Servlet filtering allows HTTP requests to be parsed and optionally rejected or modified before the request is delivered to the Web Gateway. WebSphere MQ Managed File Transfer includes a sample implementation of a servlet filter, which demonstrates this capability.

For example, for security reasons you might want to reject any requests that use the `x-fte-postdest` header to specify a command to execute after a file transfer has completed. Alternatively you might want to modify one of the values in the request, such as the queue manager name.

For more information about the sample servlet filter, see "Filtering requests with the sample servlet filter" on page 115.

Sandboxing on destination agents

When uploading files to a destination agent using the Web Gateway, you can upload the file to an absolute path on the destination agent's system. If you do not want to allow transfers from the Web Gateway to have access to the entire file system of the destination agent, you must configure agent sandboxes or user sandboxes on any agent that is the destination of a Web Gateway file upload.

For more information about the user sandboxing, see "Working with user sandboxes" on page 105. For more information about agent sandboxing, see "Working with agent sandboxes" on page 104.

▶ V 7.5.0.1

Protecting against cross-site request forgery (CSRF) attacks

CSRF attacks use code embedded in malicious websites and HTML pages to submit requests to a web server without the knowledge of the user. Exploiting this technique can allow a malicious user to create, modify, or delete resources on the web server. For example, a malicious user could create and delete file spaces or modify permissions of a file space. The Web Gateway provides the option to perform checks on all HTTP POST and HTTP DELETE requests to ensure that they contain a CSRF validation token. The token must be included as an HTTP header or an HTML form property and must contain the value of the current JSESSIONID. This double-submission technique ensures that code originating from a malicious source cannot create a complete request message and will be rejected by the server.

By default, CSRF protection is disabled for the Web Gateway. To enable it, set the value of the Web Gateway **CSRFProtection** initialization parameter to `true`.

For more information about enabling CSRF protection, see "Deploying the Web Gateway with WebSphere Application Server Version 7.0" on page 184 or "Preparing to deploy the Web Gateway with WebSphere Application Server Community Edition" on page 167.

For more information about setting the CSRF token in HTTP requests, see "HTTP headers and HTML form fields for using the Web Gateway" on page 867.

►V 7.5.0.1
Enabling session security

If you are using WebSphere Application Server, enabling this feature in the application server ensures that a particular JSESSIONID can be used only by the same user who it was granted to. This prevents a malicious user who might have intercepted the JSESSIONID from using it to gain access to a user's account.

For more information about enabling session security, see Session security support.

Related concepts:

"Securing the Web Gateway" on page 111
There are a number of ways that you can secure the Web Gateway. You must perform some of these security steps before you can use the Web Gateway. The other steps are optional and can increase the security of your Web Gateway and WebSphere MQ Managed File Transfer network, but they are not required for you to use the Web Gateway.

"Required security for the Web Gateway" on page 111
There are security configuration steps that you must complete before you can use the Web Gateway. These steps are configuring user roles for the Web Gateway, setting file space permissions, and, if you are using WebSphere Application Server Version 7.0, setting the correct level of security in the application server.

Related tasks:

"Filtering requests with the sample servlet filter"
You can filter HTTP requests to reject or modify them before they are delivered to the WebSphere MQ Managed File Transfer Web Gateway.

Filtering requests with the sample servlet filter

You can filter HTTP requests to reject or modify them before they are delivered to the WebSphere MQ Managed File Transfer Web Gateway.

Before you begin

You need the Java Platform, Enterprise Edition (Java EE) libraries on your class path to compile the sample servlet filter file.

About this task

The sample servlet filter provided with WebSphere MQ Managed File Transfer shows you an example of how to filter HTTP requests. The sample filter file, `SampleServletFilter.java`, is located in the `samples/web/filter` directory of your WebSphere MQ Managed File Transfer installation. It is also reproduced at the end of this topic.

Procedure

1. Compile the `SampleServletFilter.java` file to create the `SampleServletFilter.class` and `RequestWrapper.class` files.
2. Put the compiled class files onto your application server classpath. The process for doing this is specific to the application server you are using. For example, if you are using WebSphere Application Server Version 7.0, put the class files into a JAR file, and copy the JAR file into the `WAS_install_root`/lib directory.
3. Extract the module `com.ibm.wmqfte.web.war` from the Web Gateway EAR file, `com.ibm.wmqfte.web.ear`. The EAR file is located in the `MQ_INSTALLATION_PATH`/mqft/web directory of your WebSphere MQ Managed File Transfer Service installation. To extract the `com.ibm.wmqfte.web.war` file, run the following command:

   ```
   jar -xf com.ibm.wmqfte.web.ear com.ibm.wmqfte.web.war
   ```

4. Extract the `web.xml` file from the `com.ibm.wmqfte.web.war` file by running the following command:

```
jar -xf com.ibm.wmqfte.web.war WEB-INF/web.xml
```

5. Use a text editor to uncomment the following lines in the `web.xml` file:

```
<filter>
  <filter-name>SampleServletFilter</filter-name>
  <filter-class>SampleServletFilter</filter-class>
</filter>

<filter-mapping>
  <filter-name>SampleServletFilter</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
```

Note: If you are writing your own servlet filter, change the <filter-name> and <filter-class> values in the `web.xml` file to match your servlet filter. Leave the `url-pattern` value as `/*`.

6. Update the Web Gateway application with the modified `WEB-INF/web.xml` file, by running the following command:

```
jar -uf com.ibm.wmqfte.web.war WEB-INF/web.xml
```

7. Update the EAR file with the updated WAR file, by running the following command:

```
jar -uf com.ibm.wmqfte.web.ear com.ibm.wmqfte.web.war
```

8. Deploy the Web Gateway application to your application server. For instructions on deploying the application, see "Deploying the WebSphere MQ Managed File Transfer Web Gateway" on page 183.

Example

```
/*
 *
 * Version: %Z% %I% %W% %E% %U% [%H% %T%]
 *
 * Licensed Materials - Property of IBM
 *
 * 5724-H72
 *
 * Copyright IBM Corp. 2010 All Rights Reserved.
 *
 * US Government Users Restricted Rights - Use, duplication or
 * disclosure restricted by GSA ADP Schedule Contract with
 * IBM Corp.
 *
 */

import java.io.IOException;
import java.util.Enumeration;
import java.util.logging.Level;
import java.util.logging.LogRecord;
import java.util.logging.Logger;

import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletRequestWrapper;
import javax.servlet.http.HttpServletResponse;

/**
 * A sample servlet filter implementation that demonstrates how an application
 * server administrator can filter (reject or modify) HTTP requests before they
```

```
* are passed to the Web Gateway. The filter is called when a request
* is received by the application server for any servlet which has this
* class configured as a filter.
*
* In this example implementation two parts of an HTTP request are checked before
* the request is passed to the servlet:
*
* 1 - If the x-fte-postdest header has been set, the request is rejected by
*     returning an HTTP 400 Bad Request in a response to the HTTP client.
*
*     This demonstrates how an administrator can use servlet filters to reject
*     WMQFTE HTTP requests that they don't want to reach the WMQFTE environment.
*     In this example, the filter rejects any HTTP request that specifies a
*     command to execute after the transfer has completed.
*
* 2 - If the destination agent that is specified in a file upload URI matches one
*     of the three aliases defined in this filter (ACCOUNTS, MARKETING and WAREHOUSE),
*     the destination alias is replaced with the actual destination agent and queue
*     manager values for that alias.
*
*     This demonstrates how an administrator can use servlet filters to modify
*     any part of a request before it is passed through to the WMQFTE
*     environment. In this example, the destination agent is changed in the
*     request URI if it matches one of a number of known aliases.
*
public class SampleServletFilter implements Filter {

/*
 * (non-Javadoc)
 * @see javax.servlet.Filter#doFilter(javax.servlet.ServletRequest,
 *     javax.servlet.ServletResponse, javax.servlet.FilterChain)
 */
public void doFilter(ServletRequest request, ServletResponse response, FilterChain
      chain) throws IOException, ServletException {

 Logger sampLogger = Logger.getLogger("SampleServletFilter");
 sampLogger.log(new LogRecord(Level.INFO, "WebSphere MQ File Transfer Edition Web
  Gateway - SampleServletFilter invoked"));

 RequestWrapper modifiedRequest = null;

 if (request instanceof HttpServletRequest && response instanceof HttpServletResponse) {

  HttpServletRequest httpRequest = (HttpServletRequest) request;
  HttpServletResponse httpResponse = (HttpServletResponse) response;

  /*****************************************************************
   * The first part of the filter - reject any requests that attempt
   * to run commands on the destination agent system
   *****************************************************************/

  /*
   *  Get any 'x-fte-postdest' headers which might have been set
   */
  Enumeration<?> postDestCalls = httpRequest.getHeaders("x-fte-postdest");

  if (postDestCalls != null && postDestCalls.hasMoreElements()) {

   /*
    * Because we want to filter out all requests that attempt to run commands
    * on the destination agent system, if we find any values at all for the
    * x-fte-postdest header then we reject the request instead of proceeding.
    */
```

```
        httpResponse.setContentType("text/html");
        httpResponse.sendError(HttpServletResponse.SC_BAD_REQUEST, "Request
            rejected - an attempt to run commands was detected.");
      }

      /*******************************************************************
       * The second part of the filter - map our own aliases for WMQFTE
       * agents to the correct agent and queue manager pair
       *******************************************************************/
      String requestURI = httpRequest.getRequestURI();

      if (requestURI.indexOf("/agent/ACCOUNTS") >= 0) {
       modifiedRequest = new RequestWrapper(httpRequest);
       modifiedRequest.changeDestinationAgent("/agent/ACCOUNTS", "/agent/ACTS.AGENT@ACTS.QM");
      } else if (requestURI.indexOf("/agent/MARKETING") >= 0) {
       modifiedRequest = new RequestWrapper(httpRequest);
       modifiedRequest.changeDestinationAgent("/agent/MARKETING", "/agent/MKTG.AGENT@MKTG.QM");
      } else if (requestURI.indexOf("/agent/WAREHOUSE") >= 0) {
       modifiedRequest = new RequestWrapper(httpRequest);
       modifiedRequest.changeDestinationAgent("/agent/WAREHOUSE", "/agent/WRHS.AGENT@WRHS.QM");
      } else {
       // Leave the original request URI in place
      }

      /*******************************************************************
       * Finally call the next filter in the chain with the original
       * request (or a new wrapped request if one has been created) and
       * the original response.
       *******************************************************************/
      if (modifiedRequest != null) {
       chain.doFilter(modifiedRequest, response);
      } else {
       chain.doFilter(request, response);
      }
    } else {
     chain.doFilter(request, response);
    }
  }

  /*
   * (non-Javadoc)
   * @see javax.servlet.Filter#destroy()
   */
  public void destroy() {
   // Do nothing
  }

  /*
   * (non-Javadoc)
   * @see javax.servlet.Filter#init(javax.servlet.FilterConfig)
   */
  public void init(FilterConfig config) throws ServletException {
   // Do nothing
  }

}

/**
 * A class to wrap an <code>HttpServletRequest</code> so we can modify parts of the request
 */
class RequestWrapper extends HttpServletRequestWrapper {
```

```
private String originalDestination, newDestinationAgent;

/*
 * Constructor
 */
public RequestWrapper(HttpServletRequest request) {
  super(request);
}

/*
 *
 * (non-Javadoc)
 * @see javax.servlet.http.HttpServletRequestWrapper#getRequestURI()
 */
@Override
public String getRequestURI() {
  String originalURI = super.getRequestURI();

  StringBuffer newURI = new StringBuffer();

  newURI.append(originalURI.substring(0, originalURI.indexOf(originalDestination)));
  newURI.append(newDestinationAgent);
  newURI.append(originalURI.substring(originalURI.indexOf(originalDestination)
    + originalDestination.length(),
          originalURI.length()));

  return newURI.toString();
}

/**
 * Change the original destination agent/queue manager set in the request by the
 * HTTP client (or a previous filter) to a new destination agent/queue manager.
 *
 * @param originalDestination
 * @param newDestination
 */
protected void changeDestinationAgent(String originalDestination, String newDestination) {
  this.originalDestination = originalDestination;
  this.newDestinationAgent = newDestination;
}

}
```

Configuring an SSL or TLS connection between the Connect:Direct bridge agent and the Connect:Direct node

Configure the Connect:Direct bridge agent and the Connect:Direct node to connect to each other through the SSL protocol by creating a keystore and a truststore, and by setting properties in the Connect:Direct bridge agent properties file.

About this task

These steps include instructions for getting your keys signed by a certificate authority. If you do not use a certificate authority, you can generate a self-signed certificate. For more information about generating a self-signed certificate, see Working with SSL or TLS on UNIX and Windows systems.

These steps include instructions for creating a new keystore and truststore for the Connect:Direct bridge agent. If the Connect:Direct bridge agent already has a keystore and truststore that it uses to connect securely to WebSphere MQ queue managers, you can use the existing keystore and truststore when connecting securely to the Connect:Direct node. For more information, see "Configuring SSL encryption

for WebSphere MQ Managed File Transfer" on page 109.

Procedure

For the Connect:Direct node, complete the following steps:

1. Generate a key and signed certificate for the Connect:Direct node. You can do this by using the IBM Key Management tool that is provided with WebSphere MQ. For more information, see Working with SSL or TLS.

2. Send a request to a certificate authority to have the key signed. You receive a certificate in return.

3. Create a text file; for example, /test/ssl/certs/CAcert, that contains the public key of your certification authority.

4. Install the Secure+ Option on the Connect:Direct node. If the node already exists, you can install the Secure+ Option by running the installer again, specifying the location of the existing installation, and choosing to install only the Secure+ Option.

5. Create a new text file; for example, /test/ssl/cd/keyCertFile/*node_name*.txt.

6. Copy the certificate that you received from your certification authority and the private key, located in /test/ssl/cd/privateKeys/*node_name*.key, into the text file. The contents of /test/ssl/cd/keyCertFile/*node_name*.txt must be in the following format:

```
-----BEGIN CERTIFICATE-----
MIICnzCCAgigAwIBAgIBGjANBgkqhkiG9w0BAQUFADBeMQswCQYDVQQGEwJHQjES
MBAGA1UECBMJSGFtcHNoaXJlMRAwDgYDVQQHEwdIdXJzbGV5MQwwCgYDVQQKEwNJ
Qk0xDjAMBgNVBAsTBU1RSVBMQswCQYDVQQDEwJDQTAeFw0xMTAzMDExNjIwNDZa
Fw0yMTAyMjYxNjIwNDZaMFAxCzAJBgNVBAYTAkdCMRIwEAYDVQQIEwlIYW1wc2hp
cmUxDDAKBgNVBAoTA0lCTTEOMAwGA1UECxMFTVFVTDzANBgNVBAMTBmMpbmJh
ZzCBnzANBgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEAvgP1QIklU9ypSKD1XoODo1yk
EyMFXBOUpZRrDVxjoSEC0vtWNcJ199e+Vc4UpNybDyBu+NkDlMNofX4QxeQcLAFj
WnhakqCiQ+JIAD5AurhnrwChe0MV3kjA84GKH/rOSVqtl984mu/lDyS819XcfSSn
cOOMsK1KbneVSCIV2XECAwEAAaN7MHkwCQYDVR0TBAIwADAsBglghkgBhvhCAQ0E
HxYdT3BlblNTTCBHZW5lcmF0ZWQgQ2VydGlmaWNhdGUwHQYDVR0OBBYEFNXMIpSc
csBXUniW4A3UrZnCRsv3MB8GA1UdIwQYMBaAFDXY8rmj4lVz5+FVAoQb++cns+B4
MA0GCSqGSIb3DQEBBQUAA4GBAFc7klXa4pGKYgwchxKpE3ZF6FNwy4vBXS216/ja
8h/vl8+iv01OCL8t0ZOKSU95fyZLzOPKnCH7v+ItFSE3CIiEk9Dlz2U6WO9lICwn
17PL72TdfaL3kabwHYVf17IVcuL+VZsZ3HjLggP2qHO9ZuJPspeT9+AxFVMLiaAb
8eHw
-----END CERTIFICATE-----
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: DES-EDE3-CBC,64A02DA15B6B6EF9

57kqxLOJ/gRUOIQ6hVK2YN13B4E1jAi1gSme0I5ZpEIG8CHXISKB7/0cke2FTqsV
lvI99QyCxsDWoMNt5fj51v7aPmVeS60bOm+UlGre8B/Ze18JVj2O4K2Uh72rDCXE
5e6eFxSdUM207sQDy20euBVELJtM2kOkL1ROdoQQSlU3XQNgJw/t3ZIx5hPXWEQT
rjRQO64BEhb+PzzxPF8uwzZ9IrUK9BJ/UUnqC6OdBR87IeA4pnJD1Jvb2ML7EN9Z
5Y+50hTKI8OGvBvWXO4fHyvIX5aslwhBoArXIS1AtNTrptPvoaP1zyIAeZ6OCVo/
SFo+A2UhmtEJeOJaZG2XZ3H495fAw/EHmjehzIACwukQ9nSIETgu4A1+CV64RJED
aYBCM8UjaAkbZDH5gn7+eBov0ssXAXWDyJBVhUOjXjvAj/e1h+kcSF1hax5D//AI
66nRMZzboSxNqkjcVd8wfDwP+bEjDzUaaarJTS7lIFeLLw7eJ8MNAkMGicDkycL0
EPBU9X5QnHKLKOfYHN/1WgUk8qt3UytFXXfzTXGF3EbsWbBupkT5e5+lYcX8OVZ6
sHFPNlHluCNy/riUcBy9iviVeodX8IomOchSyO5DKl8bwZNjYtUP+CtYHNFU5BaD
I+1uUOAeJ+wjQYKT1WaeIGZ3VxuNITJul8y5qDTXXfX7vxM5OoWXa6U5+AYuGUMg
/itPZmUmNrHjTk7ghT6i1IQOaBowXXKJBlMmq/6BQXN2IhkD9ys2qrvM1hdi5nAf
egmdiG50loLnBRqWbfR+DykpAhK4SaDi2F52Uxovw3Lhiw8dQP7lzQ==
-----END RSA PRIVATE KEY-----
```

7. Start the Secure+ Admin Tool.

   • On Linux or UNIX systems, run the command `spadmin.sh`.

   • On Windows systems, click **Start** > **Programs** > **Sterling Commerce Connect:Direct** > **CD Secure+ Admin Tool**

   The CD Secure+ Admin Tool starts.

8. In the CD Secure+ Admin Tool, double-click the **.Local** line to edit the main SSL or TLS settings.

   a. Select **Enable SSL Protocol** or **Enable TLS Protocol**, depending on which protocol you are using.

   b. Select **Disable Override**.

   c. Select at least one Cipher Suite.

   d. If you want two-way authentication, change the value of **Enable Client Authentication** to `Yes`.

   e. In the **Trusted Root Certificate** field, enter the path to the public certificate file of your certification authority, `/test/ssl/certs/CAcert`.

   f. In the **Key Certificate File** field, enter the path to the file that you created, `/test/ssl/cd/keyCertFile/`*node_name*`.txt`.

9. Double-click the **.Client** line to edit the main SSL or TLS settings.

   a. Select **Enable SSL Protocol** or **Enable TLS Protocol**, depending on which protocol you are using.

   b. Select **Disable Override**.

For the Connect:Direct bridge agent, perform the following steps:

10. Create a truststore. You can do this by creating a dummy key and then deleting the dummy key. You can use the following commands:

    ```
    keytool -genkey -alias dummy -keystore /test/ssl/fte/stores/truststore.jks
    ```

    ```
    keytool -delete -alias dummy -keystore /test/ssl/fte/stores/truststore.jks
    ```

11. Import the public certificate of the certification authority into the truststore. You can use the following command:

    ```
    keytool -import -trustcacerts -alias myCA
            -file /test/ssl/certs/CAcert
            -keystore /test/ssl/fte/stores/truststore.jks
    ```

12. Edit the Connect:Direct bridge agent properties file. Include the following lines anywhere in the file:

    ```
    cdNodeProtocol=protocol
    cdNodeTruststore=/test/ssl/fte/stores/truststore.jks
    cdNodeTruststorePassword=password
    ```

    In the example in this step, *protocol* is the protocol you are using, either SSL or TLS, and *password* is the password that you specified when you created the truststore.

13. If you want two-way authentication, create a key and certificate for the Connect:Direct bridge agent.

    a. Create a keystore and key. You can use the following command:

       ```
       keytool -genkey -keyalg RSA -alias agent_name
               -keystore /test/ssl/fte/stores/keystore.jks
               -storepass password -validity 365
       ```

    b. Generate a signing request. You can use the following command:

       ```
       keytool -certreq -v -alias agent_name
               -keystore /test/ssl/fte/stores/keystore.jks -storepass password
               -file /test/ssl/fte/requests/agent_name.request
       ```

    c. Import the certificate you receive from the preceding step into the keystore. The certificate must be in x.509 format. You can use the following command:

       ```
       keytool -import -keystore /test/ssl/fte/stores/keystore.jks
               -storepass password -file certificate_file_path
       ```

    d. Edit the Connect:Direct bridge agent properties file. Include the following lines anywhere in the file:

       ```
       cdNodeKeystore=/test/ssl/fte/stores/keystore.jks
       cdNodeKeystorePassword=password
       ```

       In the example in this step, *password* is the password that you specified when you created the keystore.

Related tasks:

"Configuring the Connect:Direct bridge" on page 190
Configure the Connect:Direct bridge to transfer files between a WebSphere MQ Managed File Transfer network and a Connect:Direct network. The components of the Connect:Direct bridge are a Connect:Direct node and a WebSphere MQ Managed File Transfer agent that is dedicated to communicating with that node. This agent is referred to as the Connect:Direct bridge agent.

Configuring WebSphere MQ Managed File Transfer

You can configure the features of WebSphere MQ Managed File Transfer after installation.

Related concepts:

"Configuring WebSphere MQ Managed File Transfer for first use" on page 125
You must perform some configuration tasks for WebSphere MQ Managed File Transfer agents and queue managers once, the first time you want to use them.

"Configuration options" on page 124
WebSphere MQ Managed File Transfer provides a set of properties files that contain key information about your setup and are required for operation. These properties files are in the configuration directory that you defined when you installed the product.

"Resource monitoring" on page 215
You can monitor WebSphere MQ Managed File Transfer resources; for example, a queue or a directory. When a condition on this resource is satisfied, the resource monitor starts a task, such as a file transfer. You can create a resource monitor by using the `fteCreateMonitor` command or the Monitors view in the WebSphere MQ Managed File Transfer plug-in for WebSphere MQ Explorer.

"Customizing WebSphere MQ Managed File Transfer with user exit routines" on page 330
You can customize the features of WebSphere MQ Managed File Transfer by using your own programs known as user exit routines.

"Configuring a WebSphere MQ Managed File Transfer logger" on page 133

"Recovery and restart for WebSphere MQ Managed File Transfer" on page 276
If your agent or queue manager are unavailable for any reason, for example because of a power or network failure, WebSphere MQ Managed File Transfer recovers as follows in these scenarios:

"Hints and tips for using WebSphere MQ Managed File Transfer" on page 362
Here are some suggestions to help you to make best use of WebSphere MQ Managed File Transfer:

"Administering WebSphere MQ Managed File Transfer" on page 201
Use WebSphere MQ Managed File Transfer commands to administer WebSphere MQ Managed File Transfer. You can also use the WebSphere MQ Explorer for some of the administrative tasks.

Related tasks:

"Configuring the Web Gateway" on page 165
You must configure the WebSphere MQ Managed File Transfer Web Gateway SupportPac to work with your existing WebSphere MQ Managed File Transfer environment. The process of configuration is specific to the application server you are using. Before configuring a Web Gateway, create a web agent on the same system as the application server.

"Configuring the Connect:Direct bridge" on page 190
Configure the Connect:Direct bridge to transfer files between a WebSphere MQ Managed File Transfer network and a Connect:Direct network. The components of the Connect:Direct bridge are a Connect:Direct node and a WebSphere MQ Managed File Transfer agent that is dedicated to communicating with that node. This agent is referred to as the Connect:Direct bridge agent.

Related reference:

"Summary of the WebSphere MQ Managed File Transfer commands" on page 415
All WebSphere MQ Managed File Transfer commands are listed with links to their detailed descriptions.

"Security overview for WebSphere MQ Managed File Transfer" on page 103
Directly after installation and with no modification, WebSphere MQ Managed File Transfer has a level of security that might be suitable for test or evaluation purposes in a protected environment. However, in a production environment, you must consider appropriately controlling who can start file transfer operations, who can read and write the files being transferred, and how to protect the integrity of files.

Each agent has its own properties file, `agent.properties`, that must contain the information that an agent uses to connect to its queue manager. The `agent.properties` file can also contain properties that alter the behavior of the agent.

WebSphere MQ Managed File Transfer provides tasks that you can use to integrate file transfer function into the Apache Ant tool.

Use the following reference information to help you to diagnose errors in WebSphere MQ Managed File Transfer:

Configuration options

WebSphere MQ Managed File Transfer provides a set of properties files that contain key information about your setup and are required for operation. These properties files are in the configuration directory that you defined when you installed the product.

You can have multiple sets of configuration options, each set of configuration options contains a set of directories and properties files. The values defined in these properties files are used as the default parameters for all WebSphere MQ Managed File Transfer commands, unless you explicitly specify a different value on the command line.

To change the default set of configuration options that you are using you can use the **fteChangeDefaultConfigurationOptions** command. To change the set of configuration options that you are using for an individual command you can use the **-p** parameter with any WebSphere MQ Managed File Transfer command.

The name of a set of configuration options is the name of the coordination queue manager, and it is recommended that this is not changed. However, it is possible to change the name of a set of configuration options but you must change the name of the `config` and `logs` directories. In the following examples, the name of the set of configuration options is represented as *coordination_qmgr_name*.

Configuration options directory structure

When you configure the product, directories and properties files are created in the following structure in the configuration directory. You can also change these directories and properties files with the following commands: **fteSetupCoordination**, **fteSetupCommands**, **fteChangeDefaultConfiguration**, and **fteCreateAgent**.

```
MQ_DATA_PATH/mqft/
    config/
        coordination_qmgr_name/
            coordination.properties
            command.properties
            agents/
                agent_name/
                    agent.properties
                    exits
            loggers/
                logger_name
                    logger.properties
    installations/
        installation_name/
            installation.properties
```

The *coordination_qmgr_name* directory is a configuration options directory. There can be more than one configuration options directory in the configuration directory. The *agent_name* directory is an agent directory. In addition to containing the `agent.properties` file, this directory contains the `exits` directory, which is the default location for user exit routines and various XML files generated by the

fteCreateBridgeAgent and **fteCreateCDAgent** commands. There can be more than one agent directory in the `agents` directory of a set of configuration options.

Properties files

installation.properties

> The `installation.properties` file specifies the name of your default set of configuration options. This entry points WebSphere MQ Managed File Transfer to a structured set of directories and property files that contain the configuration to use. Typically the name of a set of configuration options is the name of the associated coordination queue manager. For more information about the `installation.properties` file, see "The installation.properties file" on page 538.

coordination.properties

> The `coordination.properties` file specifies the connection details to the coordination queue manager. Because several WebSphere MQ Managed File Transfer installations might share the same coordination queue manager, you can use a symbolic link to a common `coordination.properties` file on a shared drive. For more information about the `coordination.properties` file, see "The coordination.properties file" on page 540.

command.properties

> The `command.properties` file specifies the command queue manager to connect to when you issue commands and the information that WebSphere MQ Managed File Transfer requires to contact that queue manager. For more information about the `command.properties` file, see "The command.properties file" on page 545.

agent.properties

> Each agent has its own properties file, `agent.properties`, that must contain the information that an agent uses to connect to its queue manager. The `agent.properties` file can also contain properties that alter the behavior of the agent. For more information about the `agent.properties` file, see "The agent.properties file" on page 548.

logger.properties

> The `logger.properties` file specifies the configuration properties for the loggers. For more information about the `logger.properties` file, see "Logger configuration properties for WebSphere MQ Managed File Transfer" on page 144.

Properties files and code pages

The content of all the WebSphere MQ Managed File Transfer properties files must remain in US English because of a limitation of Java. If you edit properties files on a non-US English system, you must use Unicode escape sequences.

Configuring WebSphere MQ Managed File Transfer for first use

You must perform some configuration tasks for WebSphere MQ Managed File Transfer agents and queue managers once, the first time you want to use them.

Connecting to WebSphere MQ

All network communication with WebSphere MQ queue managers, including communication related to WebSphere MQ Managed File Transfer, involves WebSphere MQ channels. A WebSphere MQ channel represents one end of a network link. Channels are classified as either message channels or MQI channels.

WebSphere MQ Managed File Transfer and channels

WebSphere MQ Managed File Transfer uses MQI channels to connect agents in client mode to their agent queue managers, and to connect command applications (for example, **fteCreateTransfer**) to their command and coordination queue managers. In the default configuration, these connections are made

using a SVRCONN channel called SYSTEM.DEF.SVRCONN, which exists by default on all queue managers. Because of these defaults, you do not need to alter any MQI channels for a basic WebSphere MQ Managed File Transfer installation.

There are six types of message channel end points, but this topic covers only sender-receiver pairs. See Distributed queuing components for information about other channel combinations.

Required message paths

WebSphere MQ messages can travel over message channels only, so you must ensure that channels are available for all message paths required by WebSphere MQ Managed File Transfer. These paths do not need to be direct; messages can travel through intermediate queue managers if required. This topic covers only direct point-to-point communication. See How to get to the remote queue manager for more information about these options.

The communication paths used by WebSphere MQ Managed File Transfer are as follows:

Agent to agent
> Any two agents that files are transferred between require bidirectional communication between their associated queue managers. Because this path carries the bulk data, consider making the path as short, fast, or cheap as possible according to your needs.

Agent to coordination
> Log messages from the agents that participate in a transfer must be able to reach the coordination queue manager.

Command to agent
> Any queue manager that command applications or the WebSphere MQ Explorer (using the command queue manager) connect to must be able to send messages to the queue managers of the agents that those command applications are used to control. To enable feedback messages to be shown by the commands, use a bidirectional connection.

For more information, see Verifying a server-to-server installation using the command line.

Related concepts:

"WebSphere MQ multi-instance queue managers" on page 130
WebSphere MQ Version 7.0.1 supports the creation of multi-instance queue managers. A multi-instance queue manager restarts automatically on a standby server. WebSphere MQ Managed File Transfer supports connection to multi-instance agent queue managers, a multi-instance coordination queue manager, and a multi-instance command queue manager.

Related tasks:

"Configuring WebSphere MQ queue managers"
If your WebSphere MQ Managed File Transfer network includes more than one WebSphere MQ queue manager, these WebSphere MQ queue managers must be able to remotely communicate with each other.

"Configuring the coordination queue manager" on page 128
After running the **fteSetupCoordination** command, run the *coordination_qmgr_name*.mqsc script in the *MQ_DATA_PATH*/mqft/config/*coordination_qmgr_name* directory to perform the necessary configuration for the coordination queue manager. However, if you want to do this configuration manually, complete the following steps on the coordination queue manager.

Configuring WebSphere MQ queue managers

If your WebSphere MQ Managed File Transfer network includes more than one WebSphere MQ queue manager, these WebSphere MQ queue managers must be able to remotely communicate with each other.

About this task

There are two ways to configure your queue managers to be able to communicate with each other:

- By setting up a WebSphere MQ queue manager cluster.

  For information about WebSphere MQ queue manager clusters and how to configure them, see Configuring a queue manager cluster.

- By setting up channels between the queue managers, which is described as follows:

Setting up channels between queue managers

> Set up the following message channels between your queue managers:
>
> - From the agent queue manager to the coordination queue manager
> - From the command queue manager to the agent queue manager.
> - From the agent queue manager to the command queue manager (to enable feedback messages to be shown by the commands).
> - From the command queue manager to the coordination queue manager
> - From the agent queue manager to any other agent queue manager in the WebSphere MQ Managed File Transfer network
>
> If you need further information about how to set up this communication, start with this information: Administering remote WebSphere MQ objects using MQSC.
>
> Some suggested example steps are:

Procedure

1. Create a transmission queue on the WebSphere MQ queue manager with the same name as the coordination queue manager. You can use the following MQSC command:

   `DEFINE QLOCAL(`*`coordination-qmgr-name`*`) USAGE(XMITQ)`

2. On the WebSphere MQ queue manager, create a sender channel to the WebSphere MQ Managed File Transfer coordination queue manager. The name of the transmission queue created in the previous step is a required parameter for this channel. On the sender channel, ensure the CONVERT parameter of the channel is set to no. (WebSphere MQ Managed File Transfer always publishes messages in UTF-8 format, which means that any data conversion corrupts the message.) You can use the following MQSC command:

```
DEFINE CHANNEL(channel-name) CHLTYPE(SDR) CONNAME('coordination-qmgr-host(coordination-qmgr-port)')
 XMITQ(coordination-qmgr-name) CONVERT(NO)
```

3. On the WebSphere MQ Managed File Transfer coordination queue manager, create a receiver channel to the WebSphere MQ queue manager. Give this receiver channel the same name as the sender channel on the WebSphere MQ queue manager. You can use the following MQSC command:

   `DEFINE CHANNEL(`*`channel-name`*`) CHLTYPE(RCVR)`

What to do next

Next, follow the configuration steps for your coordination queue manager: Configuring the coordination queue manager.

Related tasks:

"Configuring the coordination queue manager"
After running the **fteSetupCoordination** command, run the *coordination_qmgr_name*.mqsc script in the *MQ_DATA_PATH*/mqft/config/*coordination_qmgr_name* directory to perform the necessary configuration for the coordination queue manager. However, if you want to do this configuration manually, complete the following steps on the coordination queue manager.

Configuring the coordination queue manager

After running the **fteSetupCoordination** command, run the *coordination_qmgr_name*.mqsc script in the *MQ_DATA_PATH*/mqft/config/*coordination_qmgr_name* directory to perform the necessary configuration for the coordination queue manager. However, if you want to do this configuration manually, complete the following steps on the coordination queue manager.

About this task

Procedure

1. Create a local queue named SYSTEM.FTE.
2. Add the SYSTEM.FTE queue to the SYSTEM.QPUBSUB.QUEUE.NAMELIST namelist.
3. Create a topic named SYSTEM.FTE with a topic string of SYSTEM.FTE.
4. Ensure the Non-persistent message delivery (NPMSGDLV) and Persistent messages delivery (PMSGDLV) attributes of the SYSTEM.FTE topic are set to ALLAVAIL.
5. Ensure the Publish/Subscribe mode (PSMODE) attribute of the coordination queue manager is set to ENABLED.

What to do next

If you run the strmqm -c command on a queue manager that has been configured as a coordination queue manager, the command deletes the change made in step 2 (adding the SYSTEM.FTE queue to the SYSTEM.QPUBSUB.QUEUE.NAMELIST namelist). This is because strmqm -c re-creates the default IBM WebSphere MQ objects and reverses the WebSphere MQ Managed File Transfer changes. Therefore, if you have started the queue manager with strmqm -c, complete either one of the following steps:

- Run the *coordination_qmgr_name*.mqsc script on the queue manager again.
- Repeat step 2.

Related concepts:

"Connecting to WebSphere MQ" on page 125
All network communication with WebSphere MQ queue managers, including communication related to WebSphere MQ Managed File Transfer, involves WebSphere MQ channels. A WebSphere MQ channel represents one end of a network link. Channels are classified as either message channels or MQI channels.

"WebSphere MQ multi-instance queue managers" on page 130
WebSphere MQ Version 7.0.1 supports the creation of multi-instance queue managers. A multi-instance queue manager restarts automatically on a standby server. WebSphere MQ Managed File Transfer supports connection to multi-instance agent queue managers, a multi-instance coordination queue manager, and a multi-instance command queue manager.

Related tasks:

"Configuring WebSphere MQ queue managers" on page 126
If your WebSphere MQ Managed File Transfer network includes more than one WebSphere MQ queue manager, these WebSphere MQ queue managers must be able to remotely communicate with each other.
"Configuring WebSphere MQ queue managers" on page 126
If your WebSphere MQ Managed File Transfer network includes more than one WebSphere MQ queue manager, these WebSphere MQ queue managers must be able to remotely communicate with each other.

Related reference:

The **fteSetupCoordination** command creates properties files and the coordination queue manager directory for WebSphere MQ Managed File Transfer.

Configuring agent queue managers

After installation, run the *agent_name*_create.mqsc script in the *MQ_DATA_PATH*/mqft/config/ *coordination_qmgr_name*/agents/*agent_name* directory to perform the necessary configuration for the agent queue manager. However, if you want to do this configuration manually, complete these steps on the agent queue manager:

About this task

Procedure

1. Create the agent operation queues. These queues are named:
   - SYSTEM.FTE.COMMAND.*agent_name*
   - SYSTEM.FTE.DATA.*agent_name*
   - SYSTEM.FTE.EVENT.*agent_name*
   - SYSTEM.FTE.REPLY.*agent_name*
   - SYSTEM.FTE.STATE.*agent_name*

   For information about the queue parameters, see "Agent queues for WebSphere MQ Managed File Transfer" on page 662.

2. Create the agent authority queues. These queues are named:
   - SYSTEM.FTE.AUTHADM1.*agent_name*
   - SYSTEM.FTE.AUTHAGT1.*agent_name*
   - SYSTEM.FTE.AUTHMON1.*agent_name*
   - SYSTEM.FTE.AUTHOPS1.*agent_name*
   - SYSTEM.FTE.AUTHSCH1.*agent_name*
   - SYSTEM.FTE.AUTHTRN1.*agent_name*

   For information about the queue parameters, see "Agent queues for WebSphere MQ Managed File Transfer" on page 662.

3. If the agent is a web agent, create the web agent operation queues. These queues are named:
   - SYSTEM.FTE.WEB.*gateway_name*
   - SYSTEM.FTE.WEB.RESP.*agent_name*
   - For information about the queue parameters, see "Agent queues for WebSphere MQ Managed File Transfer" on page 662.

What to do next

For information about creating and configuring a protocol bridge agent, see "fteCreateBridgeAgent (create and configure WebSphere MQ Managed File Transfer protocol bridge agent)" on page 432 and "Configuring a protocol bridge for an FTPS server" on page 261.

WebSphere MQ multi-instance queue managers

WebSphere MQ Version 7.0.1 supports the creation of multi-instance queue managers. A multi-instance queue manager restarts automatically on a standby server. WebSphere MQ Managed File Transfer supports connection to multi-instance agent queue managers, a multi-instance coordination queue manager, and a multi-instance command queue manager.

For information about the concept of a WebSphere MQ multi-instance queue manager, see the WebSphere MQ product documentation.

Configuring a multi-instance queue manager

For information about how to set up a WebSphere MQ multi-instance queue manager, see the WebSphere MQ product documentation.

Important: Ensure that you have read the relevant topics in the WebSphere MQ product documentation before attempting to configure a multi-instance queue manager to work with WebSphere MQ Managed File Transfer.

Using a multi-instance queue manager as an agent queue manager

To enable an agent to connect to both the active and standby instance of your multi-instance queue manager, add the `agentQMgrStandby` property to the agent's `agent.properties` file. The `agentQMgrStandby` property defines the host name and the port number used for client connections for the standby queue manager instance. The value of the property must be given in MQ CONNAME format, that is, `host_name(port_number)`.

The `agentQMgr` property specifies the name of the multi-instance queue manager. The `agentQMgrHost` property specifies host name for the active queue manager instance and the `agentQMgrPort` property specifies the port number for the active queue manager instance. The agent must connect in client mode to both the active and the standby instance of the multi-instance queue manager.

See "The agent.properties file" on page 548 for more information.

This example shows the contents of the `agent.properties` file for AGENT1 that connects to a multi-instance queue manager called QM_JUPITER. The active instance of QM_JUPITER is on the system host1 and uses the port number 1414 for client connections. The standby instance of QM_JUPITER is on the system host2 and uses port number 1414 for client connections.

```
agentName=AGENT1
agentDesc=
agentQMgr=QM_JUPITER
agentQMgrPort=1414
agentQMgrHost=host1
agentQMgrChannel=SYSTEM.DEF.SVRCONN
agentQMgrStandby=host2(1414)
```

Using a multi-instance queue manager as the coordination queue manager

To enable connections to both the active and standby instance of your multi-instance coordination queue manager, add the `coordinationQMgrStandby` property to all the `coordination.properties` files in your WebSphere MQ File Transfer Edition topology.

See "The coordination.properties file" on page 540 for more information.

This example shows the contents of a `coordination.properties` file that specifies the connection details to a multi-instance coordination queue manager called QM_SATURN. The active instance of QM_SATURN

is on the system coordination_host1 and uses the port number 1420 for client connections. The standby instance of QM_SATURN is on the system coordination_host2 and uses the port number 1420 for client connections.

```
coordinationQMgr=QM_SATURN
coordinationQMgrHost=coordination_host1
coordinationQMgrPort=1420
coordinationQMgrChannel=SYSTEM.DEF.SVRCONN
coordinationQMgrStandby=coordination_host2(1420)
```

The WebSphere MQ Managed File Transfer stand-alone logger must always connect to its queue manager in bindings mode. When using the stand-alone logger with a multi-instance coordination queue manager connect stand-alone logger, in bindings mode, to a different queue manager. The steps to do this are described in "Alternative configurations for the stand-alone logger" on page 151. You must define the channels between the stand-alone logger's queue manager and the coordination queue manager with the host name and port number of both instances of the multi-instance coordination queue manager. For information on how to do this, see the WebSphere MQ product documentation.

The WebSphere MQ Managed File Transfer plug-in for WebSphere MQ Explorer connects to the coordination queue manager in client mode. If the active instance of the multi-instance coordination queue manager fails the standby instance of the coordination queue manager becomes active and the plug-in reconnects.

The WebSphere MQ Managed File Transfer commands **fteList*** and **fteShowAgentDetails** connect directly to the coordination queue manager. If the active instance of the multi-instance coordination is unavailable these commands will attempt to connect to the standby instance of the coordination queue manager.

Using a multi-instance queue manager as the command queue manager

To enable connections to both the active and standby instance of your multi-instance command queue manager, add the `connectionQMgrStandby` property to all the `command.properties` files in your WebSphereMQ File Transfer Edition topology.

See "The command.properties file" on page 545 for more information.

This example shows the contents of a `command.properties` file that specifies the connection details to a multi-instance command queue manager called QM_MARS. The active instance of QM_MARS is on the system command_host1 and uses the port number 1424 for client connections. The standby instance of QM_MARS is on the system command_host2 and uses the port number 1424 for client connections.

```
connectionQMgr=QM_SATURN
connectionQMgrHost=command_host1
connectionQMgrPort=1424
connectionQMgrChannel=SYSTEM.DEF.SVRCONN
connectionQMgrStandby=command_host2(1424)
```

Related concepts:

"Connecting to WebSphere MQ" on page 125
All network communication with WebSphere MQ queue managers, including communication related to WebSphere MQ Managed File Transfer, involves WebSphere MQ channels. A WebSphere MQ channel represents one end of a network link. Channels are classified as either message channels or MQI channels.

Related tasks:

"Configuring WebSphere MQ queue managers" on page 126
If your WebSphere MQ Managed File Transfer network includes more than one WebSphere MQ queue manager, these WebSphere MQ queue managers must be able to remotely communicate with each other.

After running the **fteSetupCoordination** command, run the *coordination_qmgr_name*.mqsc script in the *MQ_DATA_PATH*/mqft/config/*coordination_qmgr_name* directory to perform the necessary configuration for the coordination queue manager. However, if you want to do this configuration manually, complete the following steps on the coordination queue manager.

Ensuring that WebSphere MQ Managed File Transfer log messages are retained

WebSphere MQ Managed File Transfer sends file transfer progress and log information to the coordination queue manager. The coordination queue manager publishes this information to any matching subscriptions to the SYSTEM.FTE topic. If there are no subscriptions, this information is not retained.

If transfer progress or log information is significant to your business, you must take one of the following steps to ensure that the information is retained:

- Use the WebSphere MQ Managed File Transfer database logger to copy messages published to the SYSTEM.FTE/Log topic to an Oracle or Db2 database.
- Define a subscription to the SYSTEM.FTE topic, which stores publications on a WebSphere MQ queue. Define this subscription before transferring any file transfers to ensure that all progress and log messages are retained on the queue.
- Write an application that uses the message queue interface (MQI) or WebSphere MQ JMS to create a durable subscription and process the publications that are delivered to the subscription. This application must be in operation before any files are transferred to ensure that the application receives all progress and log messages.

Each of these approaches is described in more detail in the sections that follow.

Do not rely on the WebSphere MQ Explorer plug-in to retain log information.

Using the WebSphere MQ Managed File Transfer database logger to retain log messages

The database logger is an optional component of WebSphere MQ Managed File Transfer that you can use to copy log information in to a database for analysis and auditing purposes. The database logger is a stand-alone Java application that you install on a system that hosts the coordination queue manager and the database. For more information about the database logger, see "Configuring a WebSphere MQ Managed File Transfer logger" on page 133.

Retaining progress and log messages by using the WebSphere MQ Explorer plug-in

When an instance of the WebSphere MQ Explorer plug-in is first started, the instance creates a durable subscription on the coordination queue manager. This durable subscription is used to collect the information displayed in the Transfer Log and Current Transfer Progress views. The name of the durable subscription is prefixed with the host name of the system running the associated instance of WebSphere MQ Explorer. This prefix is added in case an administrator wants to delete a durable subscription that is no longer in active use by an instance of the WebSphere MQ Explorer plug-in.

Using a durable subscription on the coordination queue manager can cause messages to build up on the SYSTEM.MANAGED.DURABLE queues. If you have a high-volume WebSphere MQ Managed File Transfer network, use the WebSphere MQ Explorer plug-in infrequently, or this message data can fill the local file system.

To avoid this happening, you can specify that the WebSphere MQ Explorer plug-in use a non-durable subscription to the coordination queue manager. Perform the following steps in your WebSphere MQ Explorer:

1. Select **Window** > **Preferences** > **WebSphere MQ Explorer** > **Managed File Transfer**
2. From the **Transfer Log subscription type** list, choose NON_DURABLE.

Storing publications on a WebSphere MQ queue

To store log or progress messages on a WebSphere MQ queue, configure a subscription on the coordination queue manager that forwards messages to this queue. For example, to forward all log messages to a queue named LOG.QUEUE, submit the following MQSC command:

```
define sub(MY.SUB) TOPICSTR('Log/#') TOPICOBJ(SYSTEM.FTE) DEST(LOG.QUEUE)
```

After the log messages have been forwarded to a WebSphere MQ queue, they are persisted on the queue until they are processed by a WebSphere MQ application that uses the queue.

Writing applications that manage a durable subscription to the SYSTEM.FTE topic

You can write applications that manage their own durable subscriptions to the SYSTEM.FTE topic by using one of the application programming interfaces supported by WebSphere MQ. These applications can receive WebSphere MQ queue or log messages and act on them appropriately for your business needs.

For more information about the available application programming interfaces, see Developing applications.

Configuring a WebSphere MQ Managed File Transfer logger

When WebSphere MQ Managed File Transfer transfers files, it publishes information about its actions to a topic on the coordination queue manager. The database logger is an optional component of WebSphere MQ Managed File Transfer that you can use to copy this information into a database for analysis and auditing purposes.

There are three versions of the logger:
- stand-alone file logger
- stand-alone database logger
- Java Platform Enterprise Edition (JEE) logger

The stand-alone file logger is only available in Version 7.5 and later.

Stand-alone file logger

The stand-alone file logger is a Java process that either runs on the system that hosts the coordination queue manager, or on a system which hosts a queue manager with connectivity to the coordination queue manager. The stand-alone file logger uses WebSphere MQ bindings to connect to its associated queue manager. The stand-alone logger is created using the **fteCreateLogger** command.

For Version 7.5 and later, you can run the stand-alone file logger as a Windows service to ensure that the file logger continues running when you log off from your Windows session, and it can be configured to start automatically when a system restarts. For instructions, see "Installing and configuring the WebSphere MQ Managed File Transfer stand-alone file logger" on page 134.

Stand-alone database logger

The stand-alone database logger is a Java application that you install on a system that hosts a queue manager and a database. The stand-alone database logger is often installed on the same system as the coordination queue manager, however it can also be installed on the same system as any queue manager which has connectivity to the coordination queue manager. The stand-alone database logger uses WebSphere MQ bindings to connect to its associated queue manager, and a type 2 or type 4 JDBC driver to connect to a Db2 or Oracle database. These types of connection are required because the stand-alone database logger uses the queue manager's XA support to coordinate a global transaction over both the queue manager and database, protecting the data.

If you are using a Windows system, you can run the stand-alone loggers as Windows services to ensure that the loggers continue running when you log off from your Windows session. For instructions, see "Installing the WebSphere MQ Managed File Transfer stand-alone database logger" on page 141 for a stand-alone database logger.

JEE database logger

The JEE database logger is provided as an EAR file, which you install into an application server. This can be more convenient than using the stand-alone database logger if you have an existing JEE application server environment available because the JEE database logger can be managed alongside your other enterprise applications. You can also install the JEE database logger on a separate system to the systems hosting your WebSphere MQ server and database. The JEE database logger is supported for use with Db2 and Oracle databases. The JEE database logger also supports Oracle Real Application Clusters when installed on WebSphere Application Server Version 7.0.

For instructions on how to configure a logger, see the following topics:
* "Installing and configuring the WebSphere MQ Managed File Transfer stand-alone file logger"
* "Installing the WebSphere MQ Managed File Transfer stand-alone database logger" on page 141
* "Installing the WebSphere MQ Managed File Transfer JEE database logger" on page 152

Installing and configuring the WebSphere MQ Managed File Transfer stand-alone file logger

The stand-alone file logger is a Java process that must connect to a coordination queue manager using WebSphere MQ bindings. To define a stand-alone file logger use the **fteCreateLogger** command and follow the steps in this topic.

About this task

For more information about the stand-alone file logger, see "Configuring a WebSphere MQ Managed File Transfer logger" on page 133. The steps in this topic configure a logger to connect to a coordination queue manager. For alternative logger configurations see "Alternative configurations for the stand-alone logger" on page 151

Procedure
1. Ensure that you have the WebSphere MQ Managed File Transfer Logger component installed. For more information, see "WebSphere MQ Managed File Transfer product options" on page 4
2. Run the **fteCreateLogger** command specifying the coordination queue manager, and setting the parameter -loggerType to FILE to create your stand-alone file logger. For more information, see "fteCreateLogger (create a WebSphere MQ Managed File Transfer logger)" on page 441.
3. Optional: If you want to use a custom format then you can modify the XML file created by the **fteCreateLogger** command. The log format definition is located in the FileLoggerFormat.xml file. For more information, see "Stand-alone file logger format" on page 135.

4. Run the MQSC commands, provided by the **fteCreateLogger** command, against your coordination queue manager to create the logger queues.

5. Identify a user to run the logger process and configure permissions for that user. For more information, see "Configuring user access for a stand-alone file logger" on page 140.

6. Optional: You can configure the stand-alone file logger further by editing the `logger.properties` file created when you ran the **fteCreateLogger** command. This file is a Java properties file that consists of key-value pairs. The `logger.properties` file is in the *MQ_DATA_PATH*/mqft/config/*coordination_qmgr_name*/loggers/*logger_name* directory. For more information about available properties and their affects, see "Logger configuration properties for WebSphere MQ Managed File Transfer" on page 144.

7. Optional: If you are using a Windows system, you can run the stand-alone file logger as a Windows service. Run the **fteModifyLogger** command with the **-s** parameter. For more information, see "fteModifyLogger (run a WebSphere MQ Managed File Transfer logging application as a Windows service)" on page 509.

8. Start the stand-alone file logger with the **fteStartLogger** command. For more information, see "fteStartLogger (start a logger)" on page 531.

   If you carried out the previous step and used the **fteModifyLogger** command with the **-s** parameter on Windows, the stand-alone file logger starts as a Windows service.

9. Check the logger output. The stand-alone file logger generates two types of output, file transfer audit data and logger diagnostic data. The file transfer audit data can be found in *MQ_DATA_PATH*/mqft/logs/*coordination_qmgr_name*/loggers/*logger_name*/logs. The logger diagnostic data can be found in *MQ_DATA_PATH*/mqft/logs/*coordination_qmgr_name*/loggers/*logger_name*

10. You can stop the logger by using the **fteStopLogger** command. For more information, see "fteStopLogger (stop a logger)" on page 536.

Results

Stand-alone file logger format

The format of message information written by the file logger can be defined in the `FileLoggerFormat.xml` file.

The configuration directory for the logger is located in *MQ_DATA_PATH*/mqft/config/*coordination_qmgr_name*/loggers/*logger_name*. When creating a new file logger, a version of this file is created that contains a default set of definitions used by the file logger. This default file can be used as a starting point when designing your own log format definition. For more information about the default log format definition, see "Stand-alone file logger default log format definition" on page 603.

A custom log format definition

A log format definition consists of a set of message types with each message type having a format definition. A format definition for a message type consists of a set of inserts provided in XPATH format and a separator that is used to separate each insert. The ordering of the inserts determines the order in which the content is placed in the lines generated for output to the log files. For example, this is the definition for the `callStarted` message type:

```
<callStarted>
  <format>
    <inserts>
      <insert type="user"   width="19" ignoreNull="false">/transaction/action/
       @time</insert>
      <insert type="user"   width="48" ignoreNull="false">/transaction/@ID</insert>
      <insert type="system" width="6"  ignoreNull="false">type</insert>
      <insert type="user"   width="0"  ignoreNull="false">/transaction/agent/
       @agent</insert>
      <insert type="user"   width="0"  ignoreNull="false">/transaction/agent/@QMgr</insert>
      <insert type="user"   width="0"  ignoreNull="false">/transaction/job/name</insert>
```

```
        <insert type="user"   width="0"  ignoreNull="true">/transaction/transferSet/
         call/command/@type</insert>
        <insert type="user"   width="0"  ignoreNull="true">/transaction/transferSet/
        call/command/@name</insert>
        <insert type="system" width="0"  ignoreNull="true">callArguments</insert>
      </inserts>
      <separator></separator>
    </format>
</callStarted>
```

This format produces a line in the log file like this:

```
2011-11-25T10:53:04;414d5120514d5f67627468696e6b20206466cf4e20004f02;[CSTR];
  AGENT1;AGENT_QM;Managed Call;executable;echo;call test;
```

The inserts provided in the format definition are in the order in which the information appears on the line in the log file. For more information on the XML schema defining the format for the `FileLoggerFormat.xml` file, see "Stand-alone file logger format XSD" on page 608.

Message types

The FTE agents write a range of different message types to the `SYSTEM.FTE/Log` sub-topic. For more information, see "The SYSTEM.FTE topic" on page 610. The log file definition can contain format definitions for these types of messages:

```
callCompleted
callStarted
monitorAction
monitorCreate
monitorFired
notAuthorized
scheduleDelete
scheduleExpire
scheduleSkipped
scheduleSubmitInfo
scheduleSubmitTransfer
scheduleSubmitTransferSet
transferStarted
transferCancelled
transferComplete
transferDelete
transferProgress
```

The format of the messages can vary. The majority of message types write a single line in the log file for each log message consumed from the `SYSTEM.FTE/Log` sub-topic. This leads to the simple case where the XPATH addresses provided in the log format definition relate to the root of the message. These are the message types that use this method to write output:

```
callCompleted
callStarted
monitorAction
monitorCreate
monitorFired
notAuthorized
scheduleDelete
scheduleExpire
scheduleSkipped
scheduleSubmitInfo
scheduleSubmitTransfer
transferStarted
transferCancelled
transferComplete
transferDelete
```

The other method used to write a log message uses multiple lines to represent the items in a transfer set within a log message. In this case the format provided is applied to each item in the transfer set within the log message. If you want to include information that is specific to each item within the transfer set, then the XPATH provided is required to use the item as its XPATH root. These are the message types that use this method to write output:

```
scheduleSubmitTransferSet
transferProgress
```

A line of output is written for each item in the transfer set. Information that you want to be fixed for all items in a transfer set can still use XPATH addresses relative to the root of the log message. In the following simplified `transferProgress` format definition example it's the timestamp and transfer ID that are fixed. Any information that is relative to an item as its root will vary for each line written. In this example the source and destination file information for each item are written.

```
<transferProgress>
    <format>
      <inserts>
        <insert type="user"   width="19" ignoreNull="false">/transaction/action/
         @time</insert>
        <insert type="user"   width="48" ignoreNull="false">/transaction/@ID</insert>
        <insert type="system" width="6"  ignoreNull="false">type</insert>
        <insert type="user"   width="3"  ignoreNull="true">status/@resultCode</insert>
        <insert type="user"   width="0"  ignoreNull="false">source/file |
         source/queue</insert>
        <insert type="user"   width="0"  ignoreNull="false">source/file/@size |
         source/queue/@size</insert>
        <insert type="user"   width="5"  ignoreNull="true">source/@type</insert>
        <insert type="user"   width="6"  ignoreNull="true">source/@disposition</insert>
        <insert type="user"   width="0"  ignoreNull="false">destination/file |
          destination/queue</insert>
        <insert type="user"   width="0"  ignoreNull="false">destination/file/@size |
          destination/queue/@size</insert>
        <insert type="user"   width="5"  ignoreNull="true">destination/@type</insert>
        <insert type="user"   width="9"  ignoreNull="true">destination/@exist</insert>
        <insert type="user"   width="0"  ignoreNull="true">status/supplement</insert>
      </inserts>
      <separator></separator>
    </format>
  </transferProgress>
```

This produces a log file entry of one or more lines in this format:

```
2011-11-25T13:45:16;414d5120514d5f67627468696e6b20206466cf4e20033702;[TPRO];0
  ;/src/test1.file;3575;file;leave ;/dest/test1.file;3575;file;overwrite;;
2011-11-25T13:45:16;414d5120514d5f67627468696e6b20206466cf4e20033702;[TPRO];0
  ;/src/test2.file;3575;file;leave ;/dest/test2.file;3575;file;overwrite;;
```

Insert Format

There are two types of insert available when defining a format for a message type: `user` and `system`. The type of an insert is defined in the `type` attribute of the insert element. Both types of inserts can also have their layout customized using the **width** and **ignoreNull** attributes of the insert element. For example:

```
<insert type="user" width="48" ignoreNull="false">/transaction/@ID</insert>
```

In this example, the insert takes the information found in the log message at `/transaction/@ID` and trims or pads it to 48 characters before writing it to the log. If the content of `/transaction/@ID` is null it writes the string null after padding it to 48 characters because the `ignoreNull` attribute is set to `false`. If `ignoreNull` is set to `true` the empty string, padded to 48 characters, is written instead. Setting width="0" means the column width is not trimmed, it does not mean that the width is trimmed to 0. The `ignoreNull` attribute can be used in this way to detect in the log when a null is found when it was not expected. This can be useful when debugging a new log file definition.

User defined inserts

A user insert contains an XPATH address for the information to be written in that insert. This address refers to a piece of information found in the FTE log message. For more information about log message formats, see:

- "File transfer log message formats" on page 626
- "Scheduled transfer log message formats" on page 650
- "Monitor log message format" on page 656

System defined inserts

System defined inserts contain a keyword that refers to a piece of information that either cannot be found in the log message or is not easy to define using the XPATH language.

Supported systems inserts are:

- `type` - Writes the type of the log message in a short format.
- `callArguments` - Writes the set of arguments supplied to a managed call in a space separated format.
- `transferMetaData` - Writes the set of metadata entries defined for a transfer in a comma separated <key>=<value> format.

The following table lists the value of "type" for system defined inserts for each message type.

Table 8. Summary of supported message types and their "type" system inserts.

Message type	Value of "type" system insert
callCompleted	[CCOM]
callStarted	[CSTR]
monitorAction	[MACT]
monitorCreate	[MCRT]
monitorFired	[MFIR]
notAuthorized	[AUTH]
scheduleDelete	[SDEL]
scheduleExpire	[SEXP]
scheduleSkipped	[SSKP]
scheduleSubmitInfo	[SSIN]
scheduleSubmitTransfer	[SSTR]
scheduleSubmitTransferSet	[SSTS]
transferStarted	[TSTR]
transferCancelled	[TCAN]
transferComplete	[TCOM]
transferDelete	[TDEL]
transferProgress	[TPRO]

Excluding message types from the stand-alone file logger:

If you want to exclude a certain message type from the file logger output, you can use empty message type elements.

About this task

Example

For example, the following format definition stops `transferProgress` messages being output by the file logger.
```
<?xml version="1.0" encoding="UTF-8"?>
<logFormatDefinition  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1.00"
 xsi:noNamespaceSchemaLocation="FileLoggerFormat.xsd">
  <messageTypes>
    <transferProgress></transferProgress>
  </messageTypes>
</logFormatDefinition>
```

Defining a limited set of custom formats for the stand-alone file logger:

It is possible to define a subset of custom message types within a log format definition to reduce the amount of configuration required to customize your log file format.

About this task

If a `messageTypes` element is not included in the `FileLoggerFormat.xml` file, the format for that message type uses the default format. You need only to specify the formats you want to be different from the default.

Example

In this example, the format definition replaces the default format for the `transferStarted` message type, with this reduced version that outputs only the user that started the transfer. All other message types use the default format because they are not included in this log format definition:
```
<?xml version="1.0" encoding="UTF-8"?>
<logFormatDefinition  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1.00"
 xsi:noNamespaceSchemaLocation="FileLoggerFormat.xsd">
  <messageTypes>
    <transferStarted>
      <format>
        <inserts>
          <insert type="user"   width="19" ignoreNull="false">/transaction/action/
           @time</insert>
          <insert type="user"   width="48" ignoreNull="false">/transaction/@ID</insert>
          <insert type="system" width="6"  ignoreNull="false">type</insert>
          <insert type="user"   width="0"  ignoreNull="true">/transaction/originator/
           userID</insert>
        </inserts>
        <separator>;</separator>
      </format>
    </transferStarted>
  </messageTypes>
</logFormatDefinition>
```

Related reference:
"Stand-alone file logger default log format definition" on page 603
Default log file format definition for the stand-alone file logger.

"Stand-alone file logger format XSD" on page 608
The schema for a stand-alone file format.

Reducing duplicate messages in the stand-alone file logger:

Duplicate log messages can occur in the log of the stand-alone file logger. By using the
`logger.properties` file you can tune the stand-alone file logger and reduce the number of duplicates.

Duplicate messages in the file logger log

In the case of a failure, a log message might be written to the log of the stand-alone file logger without
the consumption of the log message from the SYSTEM.FTE/Log# topic being committed to WebSphere®
MQ. If this happens, when the stand-alone file logger restarts it will retrieve the same message a second
time and write it to the log file again. Plan to handle the possibility of these duplicates when looking at
the log files either manually or when processing them automatically. To assist in the detection of
duplicates, the stand-alone file logger outputs the following message to the log file when it starts:
`BFGDB0054I: The file logger has successfully started`

Duplicates always happen around the start time of the stand-alone file logger, because this is when the
last message read before the previous instance failed is processed. By knowing when the new instance
has started you can detect if duplicates should be expected, and whether they need to be handled or not.

Reducing the number of duplicates

The stand-alone file logger groups together log messages it processes into transactions to improve
performance. This batch size is the maximum number of duplicate messages you may see in the case of a
failure. To reduce the number of duplicates you can tune the following property in the
`logger.properties` file:
`wmqfte.max.transaction.messages`

For example, by setting this to 1 the maximum number of duplicated messages is reduced to 1. Be aware
that modifying this value has an effect on the performance of your stand-alone file logger so thorough
testing is required to ensure this does not adversely affect your system.

The `logger.properties` file is in the *MQ_DATA_PATH*/mqft/config/*coordination_qmgr_name*/loggers/
logger_name directory. For more information on available properties and their effects, see "Logger
configuration properties for WebSphere MQ Managed File Transfer" on page 144

Configuring user access for a stand-alone file logger

In a test environment, you can add any new privileges needed to your normal user account. In a
production environment, you are recommended to create a new user with the minimum permissions
required to do the job.

About this task

You must install the stand-alone file logger and WebSphere MQ on a single system. Configure the user's
permissions as follows:

Procedure

1. Ensure that the user has permission to read and, where necessary, execute, the files installed as part of
   the WebSphere MQ Managed File Transfer installation.

2. Ensure that the user has permission to create and write to any file in the `logs` directory which is in the configuration directory. This directory is used for an event log, and if necessary for diagnostic trace and First Failure Data Capture (FFDC) files.

3. Ensure that the user has its own group, and is also not in any groups with wide-ranging permissions on the coordination queue manager. The user should not be in the mqm group. On certain platforms, the staff group is automatically given queue manager access as well; the stand-alone file logger user should not be in the staff group. You can view authority records for the queue manager itself and for objects in it by using the WebSphere MQ Explorer. Right-click the object and select **Object Authorities > Manage Authority Records**. At the command line, you can use the commands dspmqaut (display authority) or dmpmqaut (dump authority).

4. Use the Manage Authority Records window in the WebSphere MQ Explorer or the setmqaut (grant or revoke authority) command to add authorities for the user's own group (on UNIX, WebSphere MQ authorities are associated with groups only, not individual users). The authorities required are as follows:

   - Connect and Inquire on the queue manager (the WebSphere MQ Java libraries require Inquire permission to operate).
   - Subscribe permission on the SYSTEM.FTE topic.
   - Put permission on the SYSTEM.FTE.LOG.RJCT.*logger_name* queue.
   - Get permission on the SYSTEM.FTE.LOG.CMD.*logger_name* queue.

   The reject and command queue names given are the default names. If you chose different queue names when you configured the stand-alone file logger queues, add the permissions to those queue names instead.

Installing the WebSphere MQ Managed File Transfer stand-alone database logger

Complete these steps to install and configure the stand-alone database logger.

About this task

For more information about the stand-alone database logger, see "Configuring a WebSphere MQ Managed File Transfer logger" on page 133.

Note: You cannot run more than one database logger (stand-alone or JEE) against the same schema in a database at any one time. Attempting to do so would result in clashes when attempting to write transfer log data to the database.

Procedure

1. Install your database software using the documentation for your database. If JDBC support is an optional component for your database, you must install this component.

2. Run the **fteCreateLogger** command setting the parameter **-loggerType** to DATABASE to create your stand-alone database logger. For more information, see "fteCreateLogger (create a WebSphere MQ Managed File Transfer logger)" on page 441. The default schema name is FTELOG. If you use a schema name other than FTELOG, you must edit the provided SQL file appropriate to your database, `ftelog_tables_db2.sql` or `ftelog_tables_oracle.sql`, to reflect this schema name before proceeding to the next step. For more information, see wmqfte.database.schema in Choose a user and configure permissions.

3. Create the required database tables using your database's tools. The files `ftelog_tables_db2.sql` and `ftelog_tables_oracle.sql` contain SQL commands that you can run to create the tables. The SQL files are in the `MQ_INSTALLATION_PATH`/mqft/sql directory on distributed platforms. Choose the file appropriate to your database. Details of how to use and customize the SQL commands are described in comments at the start of the files. The stand-alone database logger is able to work with a schema name other than the default FTELOG if required by your local naming conventions.

4. Run the MQSC commands, provided by the **fteCreateLogger** command, against your logger command queue manager to create the logger queues. The stand-alone database logger uses two queues on the coordination queue manager. The first queue is a command queue where messages to control the operation of the stand-alone database logger are placed. The default name of this command queue is SYSTEM.FTE.LOG.CMD.*logger_name*. The second queue is a reject queue. Because the stand-alone database logger never discards log messages, if the logger encounters a message that it cannot handle, it places the message on the reject queue for examination, and possible reprocessing. You are not recommended to use the queue manager's dead letter queue for this purpose, because rejected messages do not have a DLH header and because rejected messages should not be combined with messages put to the dead letter queue for other reasons. The default name for the reject queue is SYSTEM.FTE.LOG.RJCT.*logger_name*. These two queues are defined in the MQSC script files generated by the **fteCreateLogger** command.
5. Choose a user and configure permissions
6. Optional: You can configure the stand-alone database logger further by editing the `logger.properties` file created by the **fteCreateLogger** command in step 2 on page 141. This file is a Java properties file that consists of key-value pairs. The `logger.properties` file is in the *MQ_DATA_PATH*/mqft/config/*coordination_qmgr_name*/loggers/*logger_name* directory. For more information about available properties and their effects, see "Logger configuration properties for WebSphere MQ Managed File Transfer" on page 144.
7. Optional: If you are using a Windows system, you can run the stand-alone database logger as a Windows service. Run the **fteModifyLogger** command with the **-s** parameter. For more information, see "fteModifyLogger (run a WebSphere MQ Managed File Transfer logging application as a Windows service)" on page 509.
8. Optional: If the database being used is Oracle or you are connecting to a Db2 database remotely, you will need to specify a user name and password that the logger will use to authenticate with your database server. This user name and password is specified in a credentials file that conforms to the format defined by the `MQMFTCredentials.xsd` schema. For more information, see "MQMFT credentials file format" on page 835. After creating the credential file, you must specify the location of the credentials file in the `logger.properties` file using the `wmqfte.database.credentials.file` property.
9. Start the stand-alone database logger using the **fteStartLogger** command. By default, the stand-alone database logger runs in the background and the stand-alone database logger places output into a file in the `logs` directory. If you want to run the stand-alone database logger in the foreground and produce output to the console as well as to the log file, add the **-F** parameter to the **fteStartLogger** command.

   If you carried out the previous step and used the **fteModifyLogger** command with the **-s** parameter on Windows, the stand-alone database logger starts as a Windows service.

Results

Working with a remote database
You can use the WebSphere MQ Managed File Transfer logger to communicate with a database on a remote system.

About this task

If you have a database installed on a different machine from the machine WebSphere MQ Managed File Transfer is installed on, complete the following steps. The steps apply to both Db2 and Oracle unless stated otherwise.

Procedure
1. Install a database client on the system that you have installed WebSphere MQ Managed File Transfer on.

2. Add your remote database server to your local database client configuration. This configuration update is needed for WebSphere MQ Managed File Transfer and WebSphere MQ to correctly access the database.

3. Specify the new properties in the `logger.properties` file to connect to the database using the credentials file: **`wmfte.database.credentials.file`**.

   Note: Earlier versions of WebSphere MQ Managed File Transfer used the properties **`wmqfte.oracle.user`** or **`wmqfte.database.user`**, and **`wmqfte.oracle.password`** or **`wmqfte.database.password`**. These properties are now deprecated. Use **`wmfte.database.credentials.file`** instead.

4. **Oracle only:** To allow a remote connection to the database, change the XAResourceManager stanza in the coordination queue manager's `qm.ini` file to the following (ensuring you change the database name, user name and user password to match your own information): `Oracle_XA+Acc=P/ftelog/ qgw783jhT+SesTm=35+DB=FTEAUDIT1`**`+SqlNet=FTEAUDIT1`**`+threads=false`, the change is highlighted in bold.

5. **Oracle only:** Specify a host and port in the `logger.properties` file, using the **`wmqfte.oracle.host`** and **`wmqfte.oracle.port`** properties. The default values for the host and port allow you to work with a local database client so if you have previously worked with a local database, you might not have set these values.

Related reference:

"Logger configuration properties for WebSphere MQ Managed File Transfer" on page 144
The logger has a set of configuration properties. Specify these properties in the `logger.properties` file, which is in the *MQ_DATA_PATH*`/mqft/config/`*coordination_qmgr_name*`/loggers/`*logger_name* directory.

Configuring user access for a stand-alone database logger

In a test environment, you can add any new privileges needed to your normal user account. In a production environment, you are recommended to create a new user with the minimum permissions required to do the job.

About this task

The number and type of user accounts you need to run the stand-alone database logger depends on the number of systems you use. You can install the stand-alone database logger, WebSphere MQ and your database on a single system, or across two systems. The stand-alone database logger must be on the same system as WebSphere MQ. The components can be installed in the following topologies:

Stand-alone database Logger, WebSphere MQ and the database all on the same system
You can define a single operating system user for use with all three components. This is a suitable configuration for the stand-alone database logger. The stand-alone database logger uses Bindings mode to connect to WebSphere MQ and a native connection to connect to the database.

Stand-alone database Logger and WebSphere MQ on one system, the database on a separate system
You create two users for this configuration: an operating system user on the system running the stand-alone database logger, and a operating system user with remote access to the database on the database server. This is a suitable configuration for the stand-alone database logger using a remote database. The stand-alone database logger uses Bindings mode to connect to WebSphere MQ and a client connection to access the database.

As an example, the rest of these instructions assume that the user is called `ftelog`, but you can use any user name. Configure the user's permissions as follows:

Procedure

1. Ensure that the user has permission to read and, where necessary, execute, the files installed as part of the WebSphere MQ Managed File Transfer Remote Tools and Documentation installation.

2. Ensure that the user has permission to create and write to any file in the `logs` directory (in the configuration directory). This directory is used for an event log, and if necessary for diagnostic trace and FFDC files.

3. Ensure that the user has its own group, and is not also in any groups with wide-ranging permissions on the coordination queue manager. The user should not be in the mqm group. On certain platforms, the staff group is automatically given queue manager access as well; the stand-alone database logger user should not be in the staff group. You can view authority records for the queue manager itself and for objects in it using the WebSphere MQ Explorer. Right-click the object and select **Object Authorities > Manage Authority Records**. At the command line, you can use the commands dspmqaut (display authority) or dmpmqaut (dump authority).

4. Use the Manage Authority Records window in the WebSphere MQ Explorer or the setmqaut (grant or revoke authority) command to add authorities for the user's own group (on UNIX, WebSphere MQ authorities are associated with groups only, not individual users). The authorities required are as follows:
   - Connect and Inquire on the queue manager (the WebSphere MQ Java libraries require Inquire permission to operate).
   - Subscribe permission on the SYSTEM.FTE topic.
   - Put permission on the SYSTEM.FTE.LOG.RJCT.*logger_name* queue.
   - Get permission on the SYSTEM.FTE.LOG.RJCT.*logger_name* queue.

   The reject and command queue names given are the default names. If you chose different queue names when you configured the stand-alone database logger queues, add the permissions to those queue names instead.

5. Perform the user configuration that is specific to the database you are using.
   - If your database is Db2, carry out the following steps:
   There are several mechanisms for managing database users with Db2. These instructions apply to the default scheme based on operating system users.
     – Ensure that the `ftelog` user is not in any Db2 administration groups (for example, db2iadm1, db2fadm1, or dasadm1)
     – Give the user permission to connect to the database and permission to select, insert and update on the tables that you created as part of Step 2: create the required database tables
   - If your database is Oracle, carry out the following steps:
     – Ensure that the `ftelog` user is not in any Oracle administration groups (for example, ora_dba on Windows or dba on Unix)
     – Give the user permission to connect to the database and permission to select, insert and update on the tables that you created as part of Step 2: create the required database tables

Logger configuration properties for WebSphere MQ Managed File Transfer

The logger has a set of configuration properties. Specify these properties in the `logger.properties` file, which is in the *MQ_DATA_PATH*/mqft/config/*coordination_qmgr_name*/loggers/*logger_name* directory.

For WebSphere MQ V7.5, there is the ability for environment variables to be used in some Managed File Transfer properties that represent file or directory locations. This allows the locations of files or directories that are used when running parts of the product, to vary depending on environment changes, such as which user is running the process. For more information, see "Environment variables in WebSphere MQ Managed File Transfer properties" on page 537.

Note: When you specify file paths on Windows, the backslash (\) separator character must appear as double backslashes (\\) (that is, escaped \). Alternatively, you can use a single forward slash character (/) as a separator. For more information about character escaping in Java properties files in Oracle, see Javadoc for the Properties class.

Property name	Description	Default value
wmqfte.logger.type	The logger type in use: file, or database. Set this value to FILE, or DATABASE.	No default value
wmqfte.max.transaction.messages	The maximum number of messages that is processed in a transaction before the transaction is committed. In circular logging mode, a queue manager has a fixed amount of space available for inflight data. Ensure that you set this property with a sufficiently low value so that the available space does not run out.	50
wmqfte.max.transaction.time	The maximum length of time in milliseconds that passes between transaction commits.	5000
wmqfte.max.consecutive.reject	The maximum number of messages that can be rejected consecutively (that is, without encountering a valid message). If this number is exceeded the logger concludes that the problem is not with the messages themselves but with the configuration. For example, if you make an agent-name column in the database narrower than all of your agent names, all messages referring to agents are rejected.	50
wmqfte.reject.queue.name	The name of a queue to which the logger puts messages that the logger cannot handle. If you have a database logger see Database logger error handling and rejection for details of which messages might be put onto this queue.	SYSTEM.FTE.LOG.RJCT.*logger_name*
wmqfte.command.queue.name	The name of a queue that the logger reads command messages controlling its behavior from.	SYSTEM.FTE.LOG.CMD.*logger_name*
wmqfte.queue.manager	The queue manager that the logger connects to (the queue manager must be on the same machine as the logger).	No default value

wmqfte.message.source.type	One of the following values:
	
	automatic subscription
	    The default value. The logger creates and uses its own durable, managed subscription on the queue manager that is defined in SYSTEM.FTE/Log/#. This is an appropriate value for most scenarios.
	
	administrative subscription
	    If the automatic subscription is not appropriate, you can define a different subscription (for example, by using the WebSphere MQ Explorer, MQSC, or PCF) and instruct the logger to use that subscription. For example, use this value to partition the log space so that one logger handles agents from A-H, another logger handles I-P, and a third logger from Q-Z.
	
	queue If the WebSphere MQ topology means that creating a subscription for the logger is not convenient, you can use a queue instead. Configure WebSphere MQ so that the queue receives the messages that are typically received by a subscription to SYSTEM.FTE/Log/# on the coordination queue manager.
		automatic subscription

Property name	Description	Default value
wmqfte.message.source.name	If the message source type is `administrative subscription` or `queue`, the name of the subscription or queue to use. This property is ignored if the source type is `automatic subscription`.	No default value
wmqfte.database.credentials.file	The file that contains the user name and password for connecting to the database. For WebSphere MQ V7.5, the value of this property can contain environment variables. For more information, see "MQMFT credentials file format" on page 835.	The default value for this property is `%HOMEDRIVE%%HOMEPATH%\mqmftcredentials.xml`" on Windows and `$HOME/MQMFTCredentials.xml` on other platforms.
wmqfte.database.driver	The location of the JDBC driver classes for the database. This is typically the path and file name of a JAR file. For example, the Type 2 driver for Db2 on AIX systems requires the file `/opt/IBM/db2/V9.5/java/db2jcc.jar`. On Windows systems, specify the path separator as a forward slash character (/) for example, `C:/Program Files/IBM/SQLLIB/java/db2jcc.jar`. If your database driver consists of multiple JAR files (for example, Db2 V9.1 requires a driver JAR file and a license JAR file), include all of these JAR files in this property. Separate multiple file names by using the classpath separator for your platform, that is, the semicolon character (;) on Windows systems and the colon character (:) on other platforms.	No default value
wmqfte.database.exclude.duplicate.metadata	Controls whether entries are stored in the metadata table that contains information that can be found in other tables within the database logger schema. Set this value to `true`, or `false`. These metadata entries are no longer stored by default as it is duplication of existing data and a waste of database storage capacity. The property entries and the tables, where the same data appears, are as follows: • com.ibm.wmqfte.SourceAgent TRANSFER_EVENT or CALL_REQUEST • com.ibm.wmqfte.DestinationAgent TRANSFER_EVENT • com.ibm.wmqfte.MqmdUser TRANSFER_EVENT or CALL_REQUEST • com.ibm.wmqfte.OriginatingUser TRANSFER_EVENT or CALL_REQUEST • com.ibm.wmqfte.OriginatingHost TRANSFER_EVENT or CALL_REQUEST • com.ibm.wmqfte.TransferId TRANSFER or CALL_REQUEST • com.ibm.wmqfte.JobName TRANSFER or CALL_REQUEST Setting the value of this property to `false` causes these metadata entries to be stored in the metadata table.	true

Property name	Description	Default value
wmqfte.database.host	DB2 only: For WebSphere MQ V7.5, the host name of the database server to connect to using a Type 4 JDBC driver. If a value for this property is specified, then a value for wmqfte.database.port must also be specified. If both properties are not defined, the database logger connects by using the default Type 2 JDBC driver. If a value for this property is specified, then a credentials file for this logger (file path defined by the wmqfte.database.credentials.file property) must exist, and be accessible to define the user name and password for connecting to the database, even if the database is on the local system.	No default value
wmqfte.database.name	The name of the database that contains the WebSphere MQ Managed File Transfer log tables.	No default value
wmqfte.database.type	The database management system in use: Db2 or Oracle. Set this value to db2 or oracle.	db2
wmqfte.database.port	DB2 only: For WebSphere MQ V7.5, the port number of the database server to connect to using a Type 4 JDBC driver. If a value for this property is specified, then a value for wmqfte.database.host must also be specified. If both properties are not defined, the database logger connects by using the default Type 2 JDBC driver. If a value for this property is specified, then a credentials file for this logger (file path defined by the wmqfte.database.credentials.file property) must exist, and be accessible to define the user name and password for connecting to the database, even if the database is on the local system.	No default value
wmqfte.database.schema	The database schema that contains the WebSphere MQ Managed File Transfer logging tables. In most cases the default value is appropriate, but you might need to specify an alternative value depending on your own site-specific database considerations.	FTELOG
wmqfte.database.native.library.path	The path that contains the native libraries that are needed by your chosen database driver (if any). For example, the Type 2 driver for DB2 on AIX systems requires libraries from /opt/IBM/db2/V9.5/lib32/. As an alternative to this property, you can set the java.library.path system property by using other methods. On Solaris and HP-UX systems, before running the **fteStartLogger** command, you must also set and export the LD_LIBRARY_PATH environment variable to include the path.	No default value
wmqfte.file.logger.fileDirectory	The directory where the file logger log files are located.	mqft/logs/*coordination_dir*/loggers/ *logger_name*/logs

Property name	Description	Default value
wmqfte.file.logger.fileSize	The maximum size that a log file is allowed to grow to. The size value is a positive integer, greater than zero, followed by one of the following units: KB, MB, GB, m (minutes), h (hours), d (days), w (weeks). For example, `wmqfte.file.logger.fileSize=5MB` Specifies a maximum file size of 5MB. `wmqfte.file.logger.fileSize=2d` Specifies a maximum file size of 2 days of data.	10MB
wmqfte.file.logger.fileCount	The maximum number of log files to create. When the amount of data exceeds the maximum amount that can be stored in this number of files, the oldest file is deleted so that the number of files never exceeds the value that is specified.	3

Property name	Description	Default value

wmqfte.file.logger.mode	The logger mode in use: circular, or linear. Set this value to `CIRCULAR`, or `LINEAR`.
	
	CIRCULAR - The file logger writes information to a file until that file reaches its maximum size as defined by using the `wmqfte.file.logger.fileSize` property. When the maximum size is reached, the file logger starts a new file. The maximum number of files that are written in this mode is controlled by the value that is defined by using the `wmqfte.file.logger.fileCount` property. When this maximum number of files is reached, the file logger deletes the first file and re-creates it for use as the currently active file. If the value defined in the `wmqfte.file.logger.fileSize` property is a fixed size byte unit (for example, KB, MB, or GB) then the upper limit on disk space that is used in this mode equals fileSize multiplied by fileCount. If the value defined in the `wmqfte.file.logger.fileSize` property is a time unit (for example, `m`, `h`, `d`, or `w`) then the maximum size depends on the throughput of log messages in your system over these time periods. The log file naming convention that is used when running in this mode is: *logger_namenumber-timestamp*.`log` where:
	
	• *logger_name* is the name that is given to the logger in the `fteCreateLogger` command.
	
	• *number* is the number of the file within the set.
	
	• *timestamp* is the timestamp of when the file was created.
	
	For example,
	`LOGGER1-20111216123430147.log`
	
	LINEAR - The file logger writes information to a file until that file reaches its maximum size as defined by using the `wmqfte.file.logger.fileSize` property. When the maximum size is reached the file logger starts a new file. Previously written files are not deleted, which allows them to be kept as a historical record of log messages. Files are not deleted when running in `linear` mode, so the `wmqfte.file.logger.fileCount` property is ignored because there is no upper limit to the number of files that can be created. Because there is no upper limit when running in this mode, it is necessary to track the amount of disk space that is used by the log files to avoid running low on disk space. The log file naming convention that is used when running in this mode is: *logger_name-timestamp*.`log` where:
	
	• *logger_name* is the name that is given to the logger in the `fteCreateLogger` command.
	
	• *timestamp* is the timestamp of when the file was created.
	
	For example,
	`LOGGER-20111216123430147.log`
		No default value

Property name	Description	Default value
wmqfte.max.retry.interval	The maximum time, in seconds, between retries when the logger encounters a persistent error. Some error conditions (for example, loss of database connection) prevent the logger from continuing. When this type of condition occurs, the logger rolls back the current transaction, waits for a period, and then retries. The time that the logger waits is initially very short, so that transitory errors can be overcome quickly. However, each time the logger retries, the time that it waits is increased. This prevents too much unnecessary work from taking place when the error condition is longer-lasting, for example when a database is taken down for maintenance. Use this property to set a limit to the length of the wait so that a retry occurs in a reasonable time of the error condition being resolved.	600
loggerQMgrRetryInterval	The interval, in seconds, between checks on the availability of the queue manager by the logger's process controller.	30
maxRestartCount	The maximum number of restarts that can happen within the time interval specified by the value of the maxRestartInterval property. When this value is exceeded the logger's process controller stops restarting the logger, and instead performs an action that is based on the value of the maxRestartDelay property.	4
maxRestartInterval	The interval, in seconds, that the logger's process controller measures logger restarts over. If the number of restarts in this interval exceeds the value of the maxRestartCount property, the logger's process controller stops restarting the logger. Instead the logger's process controller performs an action that is based on the value of the maxRestartDelay property.	120
maxRestartDelay	Determines the behavior of the logger's process controller when the rate of logger restarts exceeds the value of the maxRestartCount and maxRestartInterval properties. If you specify a value less than or equal to zero, the logger's process controller is stopped. If you specify a value greater than zero, this is the number of seconds to wait before the restart history information held by the logger's process controller is reset and the logger is restarted.	-1
wmqfte.oracle.port	The port that the logger uses to connect to the Oracle instance. This port is also known as a TNS listener.	1521
wmqfte.oracle.host	The host that the logger uses to connect to the Oracle instance.	localhost

Property name	Description	Default value
trace	Optional property. Trace specification when the logger is to be run with trace enabled at logger start. The trace specification is a comma-separated list of classes, the equals character, and a trace level. For example, `com.ibm.wmqfte.databaselogger, com.ibm.wmqfte.databaselogger.operation=all`. You can specify multiple trace specifications in a colon-separated list. For example, `com.ibm.wmqfte.databaselogger=moderate:com.ibm.wmqfte.databaselogger.operation=all`	None
traceFiles	Optional property. The total number of trace files to keep. This value applies to the process controller of a logger, as well as the logger itself.	5
traceSize	Optional property. The maximum size in MB of each trace file, before trace wraps onto the next file. This value applies to the process controller of the logger, and the logger itself.	20

Related reference:

"Environment variables in WebSphere MQ Managed File Transfer properties" on page 537
In WebSphere MQ V7.5, it is possible for environment variables to be used in Managed File Transfer properties that represent file or directory locations. This allows the locations of files or directories used when running parts of the product, to vary depending on environment changes. For example, which user is running the process.

Alternative configurations for the stand-alone logger

Typically a stand-alone logger, whether it is a file or a database type, is on the same system as the coordination queue manager and is connected to the coordination queue manager in WebSphere MQ bindings mode. However, it can also be installed on the same system as any queue manager which has connectivity to the coordination queue manager. The stand-alone logger receives messages using a subscription, which the stand-alone logger creates automatically. This is the configuration described in the installation instructions.

However, if you have site-specific considerations, you can configure a stand-alone logger to receive messages in two other ways, controlled by the wmqfte.message.source.type property. This property is described in Database Logger properties.

Administrative subscription

By default, a stand-alone logger creates its own subscription to the SYSTEM.FTE/Log/# topic, using the default durable subscription options and a managed subscription (that is, the queue manager controls the backing queue used to hold the messages before they are passed to the application). If other options are required on the subscription or the queue, you can instead create a subscription yourself, set the options that you require, and configure the stand-alone logger to use that subscription instead. Remember to add permission for the stand-alone logger to use the subscription that you create.

An example of using this configuration is to partition the log space by using two wildcard subscriptions, to send logs from agents whose name begins with FINANCE into one database and logs from agents beginning with ACCOUNTING into another. This type of configuration requires two stand-alone logger instances, each with its own `logger.properties` file referring to the required subscription and its own command queue and reject queue.

To collect log messages only from agents whose names begin with ACCOUNTING, create a subscription object on your coordination queue manager with a topic string of SYSTEM.FTE/Log/ACCOUNTING*.

Set the **Wildcard usage** value to **Character level wildcard**. You must also add entries to the `logger.properties` file for your logger. For example, if you create a subscription object called ACCOUNTING.LOGS with these settings, add the following entries to the `logger.properties` file:

```
wmqfte.message.source.type=administrative subscription
wmqfte.message.source.name=ACCOUNTING.LOGS
```

The stand-alone logger handles log messages that start with the topic string of SYSTEM.FTE/Log/ only. You can specify a more restrictive topic string, but you cannot specify a less restrictive string. If you do specify a less restrictive string in error, all publications that relate to a topic string other than SYSTEM.FTE/Log/ go to the reject queue, and the stand-alone logger produces the error message BFGDB0002E. This error message implies that there is a problem with the stand-alone logger configuration.

Queue

The typical topology is where the stand-alone logger runs on the same system as the coordination queue manager. If this is not possible, you can create a subscription on the coordination queue manager using a queue on another queue manager as the subscription destination (either using a remote queue definition or by using the DESTQMGR property of the subscription). The logger can then run on the system hosting the second queue manager and read the messages from the queue. To ensure transactional integrity, the stand-alone logger must always connect to its queue manager in bindings mode. You must define the reject queue and command queue on the same queue manager that the stand-alone logger connects to. The queue managers must be at WebSphere MQ Version 7 or later.

For example, to collect log messages which are being placed on the queue USER.QUEUE by a subscription, add these entries to the `logger.properties` file:

```
wmqfte.message.source.type=queue
wmqfte.message.source.name=USER.QUEUE
```

Installing the WebSphere MQ Managed File Transfer JEE database logger

Follow these instructions to install and configure the JEE database logger.

About this task

For more information about the JEE database logger, see the topic "Configuring a WebSphere MQ Managed File Transfer logger" on page 133.

Note: You cannot run a JEE database logger at the same time as a stand-alone logger, unless these loggers are using separate instances of the database.

Procedure
1. Before installing the JEE database logger, you must prepare your environment. Use the instructions in the topic "Preparing to install the WebSphere MQ Managed File Transfer JEE database logger" on page 153.
2. You install the JEE database logger in a Java Platform, Enterprise Edition (JEE)-compliant application server. For instructions, see the following topics:
   - "Installing the WebSphere MQ Managed File Transfer JEE database logger with WebSphere Application Server Version 7.0" on page 155
   - "Installing the WebSphere MQ Managed File Transfer JEE database logger with WebSphere Application Server Community Edition" on page 159

Related tasks:

"Preparing to install the WebSphere MQ Managed File Transfer JEE database logger"
Follow these instructions to prepare your environment before installing the JEE database logger.

"Installing the WebSphere MQ Managed File Transfer JEE database logger with WebSphere Application Server Version 7.0" on page 155
Follow these instructions to install and configure the Java Platform, Enterprise Edition (JEE) database logger with WebSphere Application Server Version 7.

"Installing the WebSphere MQ Managed File Transfer JEE database logger with WebSphere Application Server Community Edition" on page 159
Follow these instructions to install and configure the JEE database logger with WebSphere Application Server Community Edition.

"Configuring user access for the JEE database logger" on page 163
When you configure the WebSphere MQ Managed File Transfer Java Platform, Enterprise Edition (JEE) database logger, you need user accounts to access WebSphere MQ, your database, and your operating system. The number of operating system users that is required depend on the number of systems you are using to host these components.

"Migrating from the stand-alone database logger to the JEE database logger" on page 164
You can migrate from the stand-alone database logger to the JEE database logger. You must stop the stand-alone database logger and install the JEE database logger. To avoid losing or duplicating log entries you must stop messages being published to the SYSTEM.FTE topic before stopping the stand-alone database logger, and restart it after you have installed the JEE database logger. Back up your database before migration. .

Related reference:

"Authorities for the logger" on page 412
The operating system user who runs the logger requires certain WebSphere MQ authorities on the logger queues and the SYSTEM.FTE topic.

Preparing to install the WebSphere MQ Managed File Transfer JEE database logger

Follow these instructions to prepare your environment before installing the JEE database logger.

About this task

For more information about the JEE database logger, see the topic "Configuring a WebSphere MQ Managed File Transfer logger" on page 133.

Procedure

1. Install your database software using the documentation for your database. If JDBC support is an optional component for your database, you must install this component.
2. Create a database using the tools provided by your database. The database must have a tablespace and bufferpool page size of at least 8K. The default schema name is FTELOG. If you use a schema name other than FTELOG, you must edit the provided SQL file appropriate to your database, `ftelog_tables_db2.sql` or `ftelog_tables_oracle.sql`, to reflect this before proceeding to the next step.
3. Create the required database tables using your database's tools. The files `ftelog_tables_db2.sql` and `ftelog_tables_oracle.sql` contain SQL commands that you can run to create the tables. The SQL files are in the `MQ_INSTALLATION_PATH`/mqft/sql directory on distributed platforms. Choose the file appropriate to your database. Details of how to use and customize the SQL commands are described in comments at the start of the files. The stand-alone database logger is able to work with a schema name other than the default FTELOG if required by your local naming conventions.
4. If you have changed the schema name from FTELOG, you must change the schema name in the EAR file. For more information, see "Changing the schema name in your Java Platform, Enterprise Edition database logger" on page 154.

Configuring WebSphere MQ File Transfer Edition   **153**

5. Create a reject queue in WebSphere MQ. Because the logger never discards log messages, if the logger encounters a message that it cannot handle, it places the message on the reject queue for examination and possible reprocessing. Do not use the queue manager's dead letter queue for this purpose, because rejected messages do not have a DLH header and because rejected messages must not be combined with messages put to the dead letter queue for other reasons. The **fteCreateLogger** command creates a reject queue. The default name for this reject queue is SYSTEM.FTE.LOG.RJCT.*logger_name*

6. Follow the instructions in the topic Configuring user access for the JEE logger.

What to do next

Now you can install the JEE database logger in a JEE-compliant application server. Use the instructions in the following topics, based on the application server you are using:

Changing the schema name in your Java Platform, Enterprise Edition database logger

The Java Platform, Enterprise Edition (JEE) database logger can use a database that has a non-default schema name. You must change the schema name in the JEE database logger EAR file.

About this task

To change the name of the schema that your JEE database logger uses, complete the following steps:

Procedure

1. Extract the JPA JAR file from the EAR file by using the following command:

   jar -xvf *ear_file* lib/*jpa_file*

   where:
   - *ear_file* is com.ibm.wmqfte.databaselogger.jee.oracle.ear or com.ibm.wmqfte.databaselogger.jee.ear depending on whether you are using Db2 or Oracle.
   - *jpa_file* is com.ibm.wmqfte.web.jpa.oracle.jar or com.ibm.wmqfte.web.jpa.jar depending on whether you are using Db2 or Oracle.

2. Extract the persistence.xml file from the JPA JAR file by using the following command:

   jar -xvf lib/*jpa_file* META_INF/persistence.xml

   where:
   - *jpa_file* is com.ibm.wmqfte.web.jpa.oracle.jar or com.ibm.wmqfte.web.jpa.jar depending on whether you are using Db2 or Oracle.

3. Edit the persistence.xml file to change the following line:

   <property name="openjpa.jdbc.Schema" value="*schema_name*" />

   where
   - *schema_name* is the schema name you want to use.

4. Update JPA JAR with the modified persistence.xml file by using the following command:

   jar -uvf lib/*jpa_file* META_INF/persistence.xml

   where:

- *jpa_file* is com.ibm.wmqfte.web.jpa.oracle.jar or com.ibm.wmqfte.web.jpa.jar depending on whether you are using Db2 or Oracle.
5. Update the EAR file with the modified JPA JAR file by using the following command:

   jar -uvf *ear_file* lib/*jpa_file*

   where:
   - *ear_file* is com.ibm.wmqfte.databaselogger.jee.oracle.ear or com.ibm.wmqfte.databaselogger.jee.ear depending on whether you are using Db2 or Oracle.
   - *jpa_file* is com.ibm.wmqfte.web.jpa.oracle.jar or com.ibm.wmqfte.web.jpa.jar depending on whether you are using Db2 or Oracle.

What to do next

Use the modified EAR file to install the JEE database logger.

Related tasks:

"Installing the WebSphere MQ Managed File Transfer JEE database logger with WebSphere Application Server Version 7.0"
Follow these instructions to install and configure the Java Platform, Enterprise Edition (JEE) database logger with WebSphere Application Server Version 7.

"Installing the WebSphere MQ Managed File Transfer JEE database logger with WebSphere Application Server Community Edition" on page 159
Follow these instructions to install and configure the JEE database logger with WebSphere Application Server Community Edition.

Installing the WebSphere MQ Managed File Transfer JEE database logger with WebSphere Application Server Version 7.0

Follow these instructions to install and configure the Java Platform, Enterprise Edition (JEE) database logger with WebSphere Application Server Version 7.

Before you begin

Before you install the JEE database logger application, follow the instructions in the topics "Preparing to install the WebSphere MQ Managed File Transfer JEE database logger" on page 153 and "Setting the native library path in WebSphere Application Server Version 7.0" on page 182.

About this task

For more information about the JEE database logger, see "Configuring a WebSphere MQ Managed File Transfer logger" on page 133.

Procedure

1. Set up the XA JDBC provider:
   a. Select **Resources** > **JDBC** > **JDBC Providers** from the WebSphere Application Server Version 7.0 administration console navigation.
   b. Create a JDBC provider using the console wizard, by clicking **New**.
   c. At Step 1 of the wizard, select the database that you are using from the **Database type** list, and the associated provider type from the **Provider type** list. From the **Implementation type** list, select **XA data source**. Click **Next**.
   d. At Step 2 of the wizard, ensure that the directory location of the required database jar files is set correctly. Click **Next**.
   e. Click **Finish** on the summary page to create the JDBC provider.
2. Create authentication aliases. You create one alias for the data source and another for WebSphere MQ:

a. Select **Security** > **Global security** from the WebSphere Application Server Version 7.0 administration console navigation.

   b. Under the **Authentication** heading, expand **Java Authentication and Authorization Service**.

   c. Click **J2C authentication data**. The authentication alias page opens.

   d. Create an authentication alias for your data source:

      1) Click **New**.

      2) Enter the details for **Alias**, **User ID**, **Password**, and **Description**. The details that are entered in the **User ID** and **Password** fields must match the details that you entered when you created your database user. For more information, see Configuring user access for the JEE database logger.

      3) Click **OK**.

   e. Create an authentication alias for WebSphere MQ:

      1) Click **New**.

      2) Enter the details for **Alias**, **User ID**, **Password**, and **Description**. The details that are entered in the **User ID** and **Password** fields must match your user and password settings for your WebSphere MQ installation.

      3) Click **OK**.

3. Create a data source:

   a. Select **Resources** > **JDBC** > **Data sources** from the WebSphere Application Server Version 7.0 administration console navigation.

   b. Select the **Scope** drop-down list and change the scope to the appropriate value. For example, `Node=yourNode, Server=yourServer`.

   c. Create a data source using the console wizard, by clicking **New**.

   d. At Step 1 of the wizard, in the **Data source name** field, enter `wmqfte-database` and in the **JNDI name** field, enter `jdbc/wmqfte-database`. Click **Next**.

   e. At Step 2 of the wizard, use the **Select an existing JDBC provider** dropdown list to select the JDBC provider created in the previous steps. Click **Next**.

   f. **Db2:** At Step 3 of the wizard, in the **Driver type** field, enter 4.

   g. **Db2:** Enter the details in the **Database name**, **Server name**, and **Port number** fields, and click **Next**.

      Oracle: Enter the connection URL in the **URL** field and choose the correct data store helper in the **Data store helper class name** field.

      Oracle RAC: When connecting to an Oracle Real Application Cluster, the connection URL must include the host information necessary to connect to all available instances of the database.

   h. At Step 4 of the wizard, select the name of the data source authentication alias that you defined in step 2d from the **Authentication alias for XA recovery** list. Select the same name from the **Component-managed authentication alias** and **Container-managed authentication alias** lists.

   i. Click **Finish** on the summary page to create the data source.

4. Optional: Verify the configuration of the data source:

   a. Select **Resources** > **JDBC** > **Data sources** from the WebSphere Application Server Version 7.0 administration console navigation.

   b. Click the **Test Connection** button.

5. Create a topic.

   a. From the WebSphere Application Server Version 7.0 administration console navigation, click **Resources** > **JMS** > **Topics**.

   b. Select the **Scope** drop-down list and change the scope to the appropriate value. For example, `Node=yourNode, Server=yourServer`.

   c. Click **New**.

d. Click **WebSphere MQ messaging provider**.

e. On the Administration panel of the property page for the topic, choose unique values for the **Name** and **JNDI name** fields, that you will reference later on in the configuration.

f. In the WebSphere MQ topic panel, enter `SYSTEM.FTE/Log/#` in the **Topic name** field.

6. Create an activation specification:

a. From the WebSphere Application Server Version 7.0 administration console navigation, click **Resources** > **JMS** > **Activation specifications**.

b. Select the **Scope** drop-down list and change the scope to the appropriate value. For example, `Node=yourNode, Server=yourServer`.

c. Click **New**.

d. Click **WebSphere MQ messaging provider**.

e. In Step 1 of the wizard, choose unique values for the **Name** and **JNDI name** fields, that you will again reference later on in the configuration.

f. In Step 1.1, enter the JNDI name for the topic that you set up in step 5 in the **Destination JNDI name** field.

g. From the **Destination type** list, select **Topic**.

h. In Step 1.2 of the wizard, select **Durable Subscription**. Enter `SYSTEM.FTE.DATABASELOGGER.AUTO` in the **Subscription name** field.

i. In Step 2 of the wizard, select **Enter all the required information into this wizard**.

j. In Step 2.1, enter your queue manager name in the **Queue manager or queue sharing group name** field.

k. In Step 2.2, select your chosen transport method from the **Transport** list. If you select **Bindings**, no other information is required. If you select **Client** or **Bindings then client**, enter the details for **Hostname**, **Port**, and **Server connection channel**.

l. Optional: Click **Test Connection** if you wish to confirm the queue manager is present. However, you can expect to receive `NOT_AUTHORIZED` until you have referenced the authentication alias in step 6n.

m. Click **Save**.

n. Click the name of the Activation Specification that you created. In the General Properties section of the Configuration tab, scroll down to the Advanced panel and enter a unique name to identify your MQ connection in the **Client ID** field. You must complete this step or your connection is rejected by MQ with the `JMSCC0101` error code.

o. If you chose **Client** as your transport method, scroll down to the Security Settings panel and select the authentication alias that you defined in step 8 from the **Authentication alias** list.

p. Click **Apply**.

q. In the Additional Properties section of the **Configuration** tab, click **Advanced Properties**. In the Connection Consumer section of the Advanced Properties panel, enter 1 into the **Maximum server sessions** field.

Note: Ensure you complete this step before proceeding. Failure to do so can cause the logger to fail to operate correctly.

r. In the Additional Properties section of the **Configuration** tab, click **Advanced Properties**. Set the value of **Stop endpoint if message delivery fails** to a minimum of 1.

If the value of the _numberOfFailedAttemptsBeforeReject property is set to more than 1 (see 9j for more information), set **Stop endpoint if message delivery fails** to at least the value of the _numberOfFailedAttemptsBeforeReject property. This prevents the endpoint from stopping when a message that cannot be processed (for example, a malformed transfer log message) is received. For more information, see "Logger error handling and rejection" on page 370.

7. Create a queue connection factory.

a. From the WebSphere Application Server Version 7.0 administration console navigation, click **Resources** > **JMS** > **Queue connection factories**.

b. Select the **Scope** drop-down list and change the scope to the appropriate value. For example, `Node=yourNode, Server=yourServer`.

c. Click **New**.

d. Click **WebSphere MQ messaging provider**.

e. In Step 1 of the wizard, choose unique values for the **Name** and **JNDI name** fields, that you will again reference later on in the configuration.

f. In Step 2, select **Enter all the required information into this wizard**.

g. In Step 2.1, enter your queue manager name in the **Queue manager or queue sharing group name** field.

h. In Step 2.2, select your chosen transport method from the **Transport** list. If you select **Bindings**, no other information is required. If you select **Client** or **Bindings then client**, enter the details for **Hostname**, **Port**, and **Server connection channel**.

i. Optional: Click **Test Connection** if you wish to confirm the queue manager is present. However, you can expect to receive `NOT_AUTHORIZED` until you have referenced the authentication alias in step 7h.

j. If you selected **Client** or **Bindings then client** as your transport method, click the name of the queue connection factory you have just created. Scroll down to the Security Settings panel of the **Configuration** tab and select the authentication alias that you defined in step 2e from the **Authentication alias for XA recovery** and **Container-managed authentication alias** lists.

8. Create a reject queue in WebSphere Application Server:

a. From the WebSphere Application Server Version 7.0 administration console navigation, click **Resources** > **JMS** > **Queues**.

b. Select the **Scope** drop-down list and change the scope to the appropriate value. For example, `Node=yourNode, Server=yourServer`.

c. Click **New**.

d. Click **WebSphere MQ messaging provider**.

e. Choose unique values for the **Name** and **JNDI name** fields, that you will again reference later on in the configuration.

f. Enter `SYSTEM.FTE.LOG.RJCT.`*logger_name* in the **Queue name** field. Ensure you have created this queue on your coordination queue manager.

g. Enter your queue manager name in the **Queue manager name** field.

h. Click **OK**.

9. Install the JEE database logger application:

a. From the WebSphere Application Server Version 7.0 administration console, select **Applications** > **New Application**.

b. Select the **Scope** drop-down list and change the scope to the appropriate value. For example, `Node=yourNode, Server=yourServer`.

c. From the options list, select **New Enterprise Application**.

d. On the **Preparing for the application installation** page, select the `com.ibm.wmqfte.databaselogger.jee.ear` file or the `com.ibm.wmqfte.databaselogger.jee.oracle.ear` file from the *MQ_INSTALLATION_PATH*/mqft/web directory of the WebSphere MQ Managed File Transfer Server installation, and click **Next**.

e. On the following screen, select **Detailed** to show all installation options and parameters, and click **Next**.

f. Click **Next** through wizard steps 1-4 to accept the default values.

g. In step 5 of the wizard, **Bind listeners for message driven beans**, scroll to the **Listener Bindings** section. Click **Activation Specification**. Enter the required values for the following fields:

Target Resource JNDI name
    The JNDI name that you specified when creating an activation specification in step 6d.

Destination JNDI name
    The JNDI name that you specified when creating a topic in step 5d.

Click **Next**.

h. In step 6 of the wizard, **Map resource references to resources**, enter the details in the **Target Resource JNDI name** field. This name is the JNDI name that you specified for the reject queue connection factory in step 7c. Click **Next**.

i. In step 7 of the wizard, **Map resource environment entry references to resources**, enter the details in the **Target Resource JNDI name** field. This name is the JNDI name of the reject queue that you created in step 8d. Click **Next**.

j. In step 8 of the wizard, **Map environment entries for EJB modules**, accept the default value of 1. Click **Next**.

   Oracle RAC: When connecting to an Oracle Real Application Cluster you must set the value of the _numberOfFailedAttemptsBeforeReject property to **at least** 2. This property determines the number of times that the logger attempts to process an audit message after a failure occurs. In a case of database failover at least one failure is likely to occur. To avoid unnecessarily moving a message to the reject queue, increasing this value allows a second attempt to be made, which usually results in success as a connection is made to the new database instance. If you find during testing that messages are still moved to the reject queue during failover of your database instance, increase this value further: the timing of the switch between instances might cause more than one failure for the same message. However, be aware that increasing this value affects all failure cases (for example, a malformed message) and not just database failover, so increase the value with care to avoid unnecessary retries.

k. In step 9 of the wizard, **Metadata for modules**, click **Next**.

l. In step 10 of the wizard, **Summary**, click **Finish**.

10. You can now start the application from the WebSphere Application Server Version 7.0 administration console:

   a. Select **Applications** > **Application Types** > **WebSphere enterprise applications** from the console navigation.

   b. Select the check box for the **Logger** enterprise application from the collection table, and click **Start**.

Results

Installing the WebSphere MQ Managed File Transfer JEE database logger with WebSphere Application Server Community Edition

Follow these instructions to install and configure the JEE database logger with WebSphere Application Server Community Edition.

Before you begin

Before installing the JEE database logger application, follow the instructions in the topic "Preparing to install the WebSphere MQ Managed File Transfer JEE database logger" on page 153.

About this task

For more information about the JEE database logger, see the topic "Configuring a WebSphere MQ Managed File Transfer logger" on page 133.

Procedure

1. Deploy the IBM WebSphere MQ resource adapter, `wmq.jmsra.rar`.

Note: If you have already deployed the WebSphere MQ Managed File Transfer Web Gateway in your WebSphere Application Server Community Edition environment, you already have a IBM WebSphere MQ resource adapter. In this case you need to uninstall that instance of the resource adapter and redeploy using a plan file that contains the combined resources for both the Web Gateway and the JEE database logger.

- To deploy the IBM WebSphere MQ resource adapter for a JEE database logger using a coordination queue manager QM_JUPITER, perform the following steps. This example applies when your WebSphere Application Server Community Edition instance is running on the same system as the IBM WebSphere MQ queue manager that you want to connect to.

    a. Create a plan file that defines a connection to the MQMFT coordination queue manager. The following example plan file defines a connection to a queue manager called QM_JUPITER, and a reference to a queue called SYSTEM.FTE.LOG.RJCT.LOGGER1 on that queue manager.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<connector xmlns="http://geronimo.apache.org/xml/ns/j2ee/connector">
  <resourceadapter>
    <resourceadapter-instance>
      <resourceadapter-name>WMQ</resourceadapter-name>
      <workmanager>
        <gbean-link>DefaultWorkManager</gbean-link>
      </workmanager>
    </resourceadapter-instance>
    <outbound-resourceadapter>
      <connection-definition>
        <connectionfactory-interface>javax.jms.ConnectionFactory</connectionfactory-interface>
        <connectiondefinition-instance>
          <name>jms/WMQFTEJEEDBLoggerRejectQueueCF</name>
          <config-property-setting name="queueManager">QM_JUPITER</config-property-setting>
          <config-property-setting name="transportType">BINDINGS</config-property-setting>
          <connectionmanager>
            <xa-transaction>
          <transaction-caching/>
            </xa-transaction>
            <single-pool>
              <max-size>10</max-size>
              <min-size>1</min-size>
              <blocking-timeout-milliseconds>5000</blocking-timeout-milliseconds>
              <idle-timeout-minutes>2</idle-timeout-minutes>
              <match-all />
            </single-pool>
          </connectionmanager>
        </connectiondefinition-instance>
      </connection-definition>
    </outbound-resourceadapter>
  </resourceadapter>
  <adminobject>
    <adminobject-interface>javax.jms.Queue</adminobject-interface>
    <adminobject-class>com.ibm.mq.connector.outbound.MQQueueProxy</adminobject-class>
    <adminobject-instance>
      <message-destination-name>jms/WMQFTEJEEDBLoggerRejectQueue</message-destination-name>
      <config-property-setting name="baseQueueManagerName">QM_JUPITER</config-property-setting>
      <config-property-setting name="baseQueueName">SYSTEM.FTE.LOG.RJCT.LOGGER1</config-property-setting>
    </adminobject-instance>
  </adminobject>
</connector>
```

    To use this plan file in your environment change QM_JUPITER to the name of your coordination queue manager.

    b. Open the WebSphere Application Server CE administration console.

    c. From the **Common Console Actions** list on the Welcome page, click **Deploy New Applications** > **Deploy New**.

    d. In the **Archive** field, enter *mq_install_root*/java/lib/jca/wmq.jmsra.rar

    e. In the **Plan** field, type the path to the plan file you created in Step 1a.

- If your WebSphere Application Server Community Edition instance is running on a different system to the IBM WebSphere MQ queue manager that you want to connect to, perform the following steps to deploy the IBM WebSphere MQ resource adapter.

a. Create a plan file that defines a connection to the WMQFTE coordination queue manager. The following example plan file defines a connection to a queue manager, QM_SATURN, that is located on a different system to your WebSphere Application Server Community Edition installation, and a reference to a queue called SYSTEM.FTE.LOG.RJCT.LOGGER1 on that queue manager. The host name of QM_SATURN is `saturn.example.com`. The port of QM_SATURN is 1415. The channel of QM_SATURN is SYSTEM.DEF.SVRCONN.

Because the application server and the queue manager are on different systems, you must use a client mode connection to the queue manager. The following plan file sets the value of the `<config-property-setting>` element that has the name `transportType` to CLIENT.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<connector xmlns="http://geronimo.apache.org/xml/ns/j2ee/connector">
  <resourceadapter>
    <resourceadapter-instance>
      <resourceadapter-name>WMQ</resourceadapter-name>
      <workmanager>
        <gbean-link>DefaultWorkManager</gbean-link>
      </workmanager>
    </resourceadapter-instance>
    <outbound-resourceadapter>
      <connection-definition>
        <connectionfactory-interface>javax.jms.ConnectionFactory</connectionfactory-interface>
        <connectiondefinition-instance>
          <name>jms/WMQFTEJEEDBLoggerRejectQueueCF</name>
          <config-property-setting name="queueManager">QM_SATURN</config-property-setting>
          <config-property-setting name="transportType">CLIENT</config-property-setting>
          <config-property-setting name="channel">SYSTEM.DEF.SVRCONN</config-property-setting>
          <config-property-setting name="hostName">saturn.example.com</config-property-setting>
          <config-property-setting name="port">1415</config-property-setting>
          <connectionmanager>
            <xa-transaction>
          <transaction-caching/>
            </xa-transaction>
            <single-pool>
              <max-size>10</max-size>
              <min-size>1</min-size>
              <blocking-timeout-milliseconds>5000</blocking-timeout-milliseconds>
              <idle-timeout-minutes>2</idle-timeout-minutes>
              <match-all />
            </single-pool>
          </connectionmanager>
        </connectiondefinition-instance>
      </connection-definition>
    </outbound-resourceadapter>
  </resourceadapter>
  <adminobject>
    <adminobject-interface>javax.jms.Queue</adminobject-interface>
    <adminobject-class>com.ibm.mq.connector.outbound.MQQueueProxy</adminobject-class>
    <adminobject-instance>
      <message-destination-name>jms/WMQFTEJEEDBLoggerRejectQueue</message-destination-name>
      <config-property-setting name="baseQueueManagerName">QM_SATURN</config-property-setting>
      <config-property-setting name="baseQueueName">SYSTEM.FTE.LOG.RJCT.LOGGER1</config-property-setting>
    </adminobject-instance>
  </adminobject>
</connector>
```

To use this plan file in your environment change QM_SATURN to the name of your coordination queue manager. Change the value of the host name, port and channel to the values for your coordination queue manager.

b. Copy the file *mq_install_root*/java/lib/jca/wmq.jmsra.rar from the system where IBM WebSphere MQ is installed to the system where WebSphere Application Server CE is installed.

c. Open the WebSphere Application Server CE administration console.

d. From the **Common Console Actions** list on the Welcome page, click **Deploy New Applications** > **Deploy New**.

e. In the **Archive** field, type the path to the copy of the `wmq.jmsra.rar` file that you obtained.

f. In the **Plan** field, type the path to the plan file you created.

2. You must define a database connector so that the JEE database logger application has access to the required database from within the WebSphere Application Server Community Edition environment.

Note: If you have already deployed the WebSphere MQ Managed File Transfer Web Gateway in your WebSphere Application Server Community Edition environment, you already have a database connector defined. In this case you do not need to repeat these steps.

Carry out the following steps from the WebSphere Application Server Community Edition administration console:

a. Depending on the level of WebSphere Application Server Community Edition that you are using, from the **Console Navigation** either select **Services** > **Database Pools**, or select **Resources** > **Datasources**.

b. Create a database pool using the Geronimo database pool wizard. In the **Name of Database Pool** field, type `jdbc/wmqfte-database`.

c. For the **Database Type** select `DB2 XA` or `Oracle Thin`, as appropriate for your database.

d. Click **Next**.

e. In the **Driver jar** field, select the appropriate jar for your database.

f. In the **Database Name** field, type the name of the database you are connecting to for transfer status information.

g. In the **User Name** field, type the user name for connecting to and authenticating with your database.

h. In the **Password** and **Confirm Password** fields, type the password for authenticating with your database.

i. In the **Port Number** field, type the port number you are using if it is not the default port.

j. Ensure that the value for **Driver Type** is 4.

k. Select `XA` from the **Transaction Type** list.

l. Click **Deploy**.

3. Update the WebSphere MQ Managed File Transfer JEE database logger application `openejb-jar.xml` file for your environment. Use a Java SDK jar utility to complete the following steps:

a. Extract the EJB jar file from the supplied EAR file by running the following command:

```
jar -xf ear_file_name com.ibm.wmqfte.databaselogger.jee.ejb.jar
```

where *ear_file_name* is `com.ibm.wmqfte.databaselogger.jee.ear` or `com.ibm.wmqfte.databaselogger.jee.oracle.ear` depending on whether you are using Db2 or Oracle. The EAR file is located in the *MQ_INSTALLATION_PATH*/mqft/web directory of the WebSphere MQ Managed File Transfer Server installation.

b. Extract the META-INF/openejb-jar.xml file from the previously extracted EJB jar file, `com.ibm.wmqfte.databaselogger.jee.ejb.jar`, by running the following command:

```
jar -xf com.ibm.wmqfte.databaselogger.jee.ejb.jar META-INF/openejb-jar.xml
```

c. Use a text editor to edit the extracted META-INF/openejb-jar.xml file. Change the following `activation-config-property` values to match your environment:

queueManager
The name of the IBM WebSphere MQ queue manager that is used by the JEE database logger.

hostName
The host name to use to connect to the specified IBM WebSphere MQ queue manager. This value is not required if you are connecting to the queue manager in bindings mode.

transportType
Whether to connect to the specified IBM WebSphere MQ queue manager in client or bindings mode.

port   Not required if you specified a **transportType** of bindings. The port to use to connect to the specified IBM WebSphere MQ queue manager.

channel

Not required if you specified a **transportType** of bindings. The server channel to use to connect to the specified IBM WebSphere MQ queue manager.

  d. Update the EJB jar file with the modified META-INF/openejb-jar.xml file, by running the following command:

```
jar -uf com.ibm.wmqfte.databaselogger.jee.ejb.jar META-INF/openejb-jar.xml
```

  e. Update the supplied ear file with the updated EJB jar file, by running the following command:

```
jar -uf ear_file_name com.ibm.wmqfte.databaselogger.jee.ejb.jar
```

where *ear_file_name* is com.ibm.wmqfte.databaselogger.jee.ear or com.ibm.wmqfte.databaselogger.jee.oracle.ear depending on your database.

4. To deploy the EAR file to the application server, complete the following steps from the WebSphere Application Server Community Edition administration console.

  a. Select: **Applications** > **Deploy New** from the **Console Navigation** menu.

  b. In the **Archive** field, specify the EAR file: com.ibm.wmqfte.databaselogger.jee.ear or com.ibm.wmqfte.databaselogger.jee.oracle.ear depending on your database.

  c. Leave the **Plan** field blank.

  d. Ensure the **Start application after install** box is selected.

  e. Click **Install**. The JEE database logger application is installed and started.

Configuring user access for the JEE database logger

When you configure the WebSphere MQ Managed File Transfer Java Platform, Enterprise Edition (JEE) database logger, you need user accounts to access WebSphere MQ, your database, and your operating system. The number of operating system users that is required depend on the number of systems you are using to host these components.

About this task

The number and type of user accounts you need to run the Java Platform, Enterprise Edition (JEE) database logger depend on the number of systems you use. User accounts are required to access the following three environments:

- Local operating system
- WebSphere MQ
- Database

You can install the JEE database logger, WebSphere MQ and your database on a single system, or across several systems. The components can be installed in the following example topologies:

JEE database logger, WebSphere MQ, and the database all on the same system
You can define a single operating system user for use with all three components. The logger uses Bindings mode to connect to WebSphere MQ and a native connection to connect to the database.

JEE database logger and WebSphere MQ on one system, the database on a separate system
You create two users for this configuration: an operating system user on the system running the logger, and an operating system user with remote access to the database on the database server. The logger uses Bindings mode to connect to WebSphere MQ and a client connection to access the database.

JEE database logger on one system, WebSphere MQ on another system, the database on a further system
You create three users for this configuration: An operating system user to start the application server, a WebSphere MQ user to access the queues and topics being used, and a database server user to access and insert into the database tables. The logger uses Client mode to access WebSphere MQ and a client connection to access the database.

As an example, the rest of these instructions assume that the user is called `ftelog`, but you can use any user name, new or existing. Configure the user permissions as follows:

Procedure

1. Ensure that the operating system user has its own group, and is not also in any groups with wide-ranging permissions on the coordination queue manager. The user should not be in the mqm group. On certain platforms, the staff group is automatically given queue manager access as well; the logger user should not be in the staff group. You can view authority records for the queue manager itself and for objects in it using the WebSphere MQ Explorer. Right-click the object and select **Object Authorities > Manage Authority Records**. At the command line, you can use the commands dspmqaut (display authority) or dmpmqaut (dump authority).

2. Use the Manage Authority Records window in the WebSphere MQ Explorer or the setmqaut (grant or revoke authority) command to add authorities for the WebSphere MQ user's own group (on UNIX, WebSphere MQ authorities are associated with groups only, not individual users). The authorities required are as follows:
   - CONNECT and INQUIRE on the queue manager (the WebSphere MQ Java libraries require INQUIRE permission to operate).
   - SUBSCRIBE permission on the SYSTEM.FTE topic.
   - PUT permission on the SYSTEM.FTE.LOG.RJCT.*logger_name* queue.

   The reject and command queue names given are the default names. If you chose different queue names when you configured the logger queues, add the permissions to those queue names instead.

3. Perform the database user configuration that is specific to the database you are using.
   - If your database is Db2, carry out the following steps:

     Note: There are several mechanisms for managing database users with Db2. These instructions apply to the default scheme based on operating system users.
     - Ensure that the `ftelog` user is not in any Db2 administration groups (for example, db2iadm1, db2fadm1, or dasadm1)
     - Give the user permission to connect to the database and permission to select, insert, and update on the tables that you created as part of Step 2: create the required database tables
   - If your database is Oracle, carry out the following steps:
     - Ensure that the `ftelog` user is not in any Oracle administration groups (for example, ora_dba on Windows or dba on Unix)
     - Give the user permission to connect to the database and permission to select, insert and update on the tables that you created as part of Step 2: create the required database tables

Migrating from the stand-alone database logger to the JEE database logger

You can migrate from the stand-alone database logger to the JEE database logger. You must stop the stand-alone database logger and install the JEE database logger. To avoid losing or duplicating log entries you must stop messages being published to the SYSTEM.FTE topic before stopping the stand-alone database logger, and restart it after you have installed the JEE database logger. Back up your database before migration. .

About this task

Procedure

1. Before stopping the database, run the following MQSC command against your coordination queue manager: `ALTER QM PSMODE(COMPAT)` This stops messages being published to the SYSTEM.FTE/Log topic. Wait until the logger has processed all of the messages on its subscription. By default, this subscription is called SYSTEM.FTE.LOGGER.AUTO.

2. Stop the database logger using the **fteStopDatabaseLogger** command.
3. Back up the database using the tools supplied with the database software.
4. Delete the subscription belonging to the stand-alone database logger. By default, this subscription is called SYSTEM.FTE.LOGGER.AUTO.
5. If your database schema is at an earlier version, you must migrate the schema to each subsequent level in order. For example, if your database schema is at V7.0.1 and you are migrating to V7.0.4, you must migrate your schema from V7.0.1 to V7.0.2, then from V7.0.2 to V7.0.3, and then from V7.0.3 to V7.0.4. Migrate your database schema from version *old* to version *new*, where *old* and *new* are variables that describe a schema version, by performing the one of the following actions for each version of the schema that you must migrate through:

   •

   • If your database is not Db2 or you created your database with a page size of more than 8K, you can migrate the schema in the same way as for other versions, by completing the following steps.

   • If you are migrating between database tables in any other circumstances complete the following steps:
       a. Choose the file that is appropriate to your database platform and has a name that includes the string *old-new*. This file is located in the *MQ_INSTALLATION_PATH*/mqft/sql directory of the Remote Tools and Documentation installation.
       b. If you have made modifications to the initial schema, review the migration file to ensure that the file will be compatible with your modified database.
       c. Run the SQL file against your database.
6. Install the JEE database logger EAR file.
7. Deploy the JEE database logger. For more information, see "Installing the WebSphere MQ Managed File Transfer JEE database logger" on page 152.
8. Run the following MQSC command against your coordination queue manager: ALTER QMGR PSMODE(ENABLED) This enables publishing of messages to the SYSTEM.FTE/Log topic.

Results

Configuring the Web Gateway

You must configure the WebSphere MQ Managed File Transfer Web Gateway SupportPac to work with your existing WebSphere MQ Managed File Transfer environment. The process of configuration is specific to the application server you are using. Before configuring a Web Gateway, create a web agent on the same system as the application server.

Before you begin

Before configuring or using the Web Gateway, refer to "Scenarios for the Web Gateway" on page 278 and "How the Web Gateway fits into your WebSphere MQ Managed File Transfer topology" on page 280. These topics explain the purpose and components of the Web Gateway.

Setting up a database for use with file spaces

Before you can use file spaces you must set up database tables for the Web Gateway to store file space information in. You can create these tables in your existing log database, or create a new database to contain the tables.

About this task

Follow these instructions to create the database tables that the Web Gateway requires to work with file spaces.

Procedure

1. If you do not have database software installed, install your database software using the documentation for your database. If JDBC support is an optional component for your database you must install this component.

2. If you do not have a database or you want to use a different database to the database that is used by the database logger, create a database using the database tools. The default schema name is FTEWEB. If you use a schema name other than FTEWEB, you must edit the provided SQL files, `webgateway_db2.sql`, `webgateway_oracle.sql` or `webgateway_zos.sql` to reflect this before proceeding to the next step. If you want to create the Web Gateway tables in the same database as the database logger tables, the two sets of tables must not have the same schema name.

3. Create the required database tables using the database tools. The files `webgateway_db2.sql`, `webgateway_oracle.sql` or `webgateway_zos.sql` contain SQL commands you can run to create the tables. The files are in the *MQ_INSTALLATION_PATH*/mqft/web/sql directory of a WebSphere MQ Managed File Transfer Service installation.

   Information about how to use and customize the SQL commands are described in the comments at the start of the files.

   Note: If you are migrating from WebSphere MQ File Transfer Edition V7.0.x to a later version of WebSphere MQ Managed File Transfer, there are no changes to the database schema for the Web Gateway. There is no SQL migration file to run against your database.

Related tasks:

"Configuring the database logger for use with the Web Gateway" on page 187
The following example shows the result of requesting the status of a transfer when the database logger is not correctly configured.

Related reference:

"Database tables used by the Web Gateway" on page 897
The WebSphere MQ Managed File Transfer Web Gateway uses the following database tables to configure and secure user file spaces.

Changing the schema name in the Web Gateway

The Web Gateway can use a database that has a non-default schema name. You must change the schema name in the Web Gateway EAR file.

About this task

The default schema name is FTEWEB. To change the name of the schema that the Web Gateway uses, complete the following steps:

Procedure

1. Extract the JAR file using the following command:

   `jar -xvf com.ibm.wmqfte.web.ear lib/com.ibm.wmqfte.web.jpa.fs.jar`

   The JAR file is found in *<product_install_location>*/mqft/web/com.ibm.wmqfte.web.ear.

2. Extract the `persistence.xml` file from the JPA JAR file by using the following command:

   `jar -xvf lib/com.ibm.wmqfte.web.jpa.fs.jar META-INF/persistence.xml`

3. Edit the `META-INF/persistence.xml` file to change the following line:

   `<property name="openjpa.jdbc.Schema" value="`*schema_name*`" />`

   where

   • *schema_name* is your chosen schema name. The default schema name is FTEWEB

4. Update JPA JAR with the modified `persistence.xml` file by using the following command:

   `jar -uvf lib/com.ibm.wmqfte.web.jpa.fs.jar META-INF/persistence.xml`

5. Update the EAR file with the modified JPA JAR file by using the following command:

```
jar -uvf com.ibm.wmqfte.web.ear lib/com.ibm.wmqfte.web.jpa.fs.jar
```

Preparing to deploy the Web Gateway

Before deploying the WebSphere MQ Managed File Transfer Web Gateway, you must set up your application server environment and dependent modules. This section describes the setup tasks for WebSphere MQ and two different application servers.

Before you begin

Before configuring or using the Web Gateway, refer to "Scenarios for the Web Gateway" on page 278 and "How the Web Gateway fits into your WebSphere MQ Managed File Transfer topology" on page 280. These topics explain the purpose and components of the Web Gateway.

Before you deploy the Web Gateway application, you must complete the required security steps. For more information, see "Required security for the Web Gateway" on page 111.

To complete your Web Gateway topology, you also need a web agent and a database logger. For more information, see "fteCreateWebAgent (create a WebSphere MQ Managed File Transfer web agent)" on page 480 and "Configuring a WebSphere MQ Managed File Transfer logger" on page 133.

Related tasks:

"Preparing to deploy the Web Gateway with WebSphere Application Server Version 7.0" on page 178
Use these instructions to define required resources before deploying the WebSphere MQ Managed File Transfer Web Gateway enterprise application to WebSphere Application Server Version 7.0. You must customize the example deployment plan for your environment.

"Preparing to deploy the Web Gateway with WebSphere Application Server Community Edition"
Use these instructions to set up your environment before deploying the WebSphere MQ Managed File Transfer Service Web Gateway enterprise application to WebSphere Application Server Community Edition. Customize the example deployment plan for your environment.

Preparing to deploy the Web Gateway with WebSphere Application Server Community Edition

Use these instructions to set up your environment before deploying the WebSphere MQ Managed File Transfer Service Web Gateway enterprise application to WebSphere Application Server Community Edition. Customize the example deployment plan for your environment.

Before you begin

Before configuring or using the Web Gateway, refer to "Scenarios for the Web Gateway" on page 278 and "How the Web Gateway fits into your WebSphere MQ Managed File Transfer topology" on page 280. These topics explain the purpose and components of the Web Gateway.

To check that you are using an application server version that is supported for use with the Web Gateway, refer to the web page IBM WebSphere MQ System Requirements.

Note: The user that your application server runs as must be the same as, or in the same group as, the user that your web agent runs as.

Before starting your application server setup, complete the following tasks to prepare your IBM WebSphere MQ environment for working with the Web Gateway.

Determine which user ID the application server uses to connect to IBM WebSphere MQ. This user ID must be given the **Set identity context** permission in your IBM WebSphere MQ environment. For

example, if the application server is running as `appuser1`, who is a member of group `appgrp`, and connecting to a local IBM WebSphere MQ queue manager called `qm1` using a bindings mode connection, then run the following command:

```
setmqaut -m qm1 -g appgrp +setid -t qmgr
```

You must also give the user ID the **Set identity context** permission on the web agent command queue. For example, if the application server is running as `appuser1`, who is a member of group `appgrp`, and the web agent is called `WEBAGENT` and it connects to a local IBM WebSphere MQ queue manager called `qm2` using a bindings mode connection, then run the following command:

```
setmqaut -m qm2 -g appgrp +setid -t queue -n SYSTEM.FTE.COMMAND.WEBAGENT
```

About this task

WebSphere Application Server Community Edition can be obtained from the following web page: http://www.ibm.com/software/webservers/appserv/community

Before deploying the Web Gateway application, you must set up the dependent components. These components are the IBM WebSphere MQ resource adapter, a database written to by a WebSphere MQ Managed File Transfer database logger, a database connector, and a security realm. You must also update the `web.xml` file and the deployment plan for your environment.

The Web Gateway also requires a WebSphere MQ Managed File Transfer web agent installed on the same system as the application and run as the same user, or a user in the same group, as the application server. For instructions on how to create and configure this agent, see "fteCreateWebAgent (create a WebSphere MQ Managed File Transfer web agent)" on page 480.

Procedure

1. Deploy the IBM WebSphere MQ resource adapter. If your WebSphere Application Server Community Edition instance is running on the same system as the IBM WebSphere MQ queue manager that you want to connect to, see "Deploying the IBM WebSphere MQ resource adapter on the same system as the application server" on page 169. If your WebSphere Application ServerCommunity Edition instance is running on a different system from the IBM WebSphere MQ queue manager that you want to connect to, see "Deploying the IBM WebSphere MQ resource adapter on a different system from the application server" on page 169.
2. Define a database connector to connect to the log database. For more information, see "Defining a database connector to connect to the log database" on page 171.
3. Define a database connector to connect to the file space database. For more information, see "Defining a database connector to connect to the file space database" on page 171
4. Define a security realm. For more information, see "Defining a security realm" on page 172.
5. Update the `web.xml` file. For more information, see "Updating the web.xml file" on page 173.
6. Update the `openejb-jar.xml` file. For more information, see "Updating the openejb-jar.xml to configure the Web Gateway to use file spaces" on page 175.
7. If you must deploy the Web Gateway in a non-default environment or are using your own security realm, you must either update the supplied deployment plan or provide a separate deployment plan. For more information, see "Update the deployment plan" on page 176.
8. Optional: If you want to deploy the Web Gateway administrative console in a non-default environment update the supplied deployment plan in the `com.ibm.wmqfte.web.admin.war` file. For more information, see "Update the deployment plan for the administrative console" on page 177.

Results

You can now deploy the Web Gateway EAR file to the application server. Carry out the steps in the topic "Deploying the Web Gateway with WebSphere Application Server Community Edition" on page 183.

Deploying the IBM WebSphere MQ resource adapter on the same system as the application server:
About this task

If your WebSphere Application Server Community Edition instance is running on the same system as the IBM WebSphere MQ queue manager that you want to connect to, perform the following steps to deploy the IBM WebSphere MQ resource adapter.

Procedure

1. Create a plan file that defines a connection to the queue manager of the source agent. The following example plan file defines a connection to a queue manager called QM_JUPITER.

```
<?xml version="1.0" encoding="UTF-8"?>
<connector xmlns="http://geronimo.apache.org/xml/ns/j2ee/connector">
  <resourceadapter>
    <resourceadapter-instance>
      <resourceadapter-name>WMQ</resourceadapter-name>
      <workmanager>
        <gbean-link>DefaultWorkManager</gbean-link>
      </workmanager>
    </resourceadapter-instance>
    <outbound-resourceadapter>
      <connection-definition>
        <connectionfactory-interface>
                javax.jms.ConnectionFactory
        </connectionfactory-interface>
        <connectiondefinition-instance>
          <name>jms/WMQFTEWebAgentConnectionFactory</name>
          <config-property-setting name="queueManager">
                    QM_JUPITER
          </config-property-setting>
          <config-property-setting name="transportType">
                      BINDINGS
          </config-property-setting>              <connectionmanager>
            <no-transaction />
            <no-pool/>
          </connectionmanager>
        </connectiondefinition-instance>
      </connection-definition>
    </outbound-resourceadapter>
  </resourceadapter>
</connector>
```

To use this plan file in your environment change QM_JUPITER to the name the queue manager of your source agent. The sections of the XML file that must be edited are highlighted in **bold** typeface.

2. Open the WebSphere Application Server CE administration console.
3. From the **Common Console Actions** list on the Welcome page, click **Deploy New Applications** > **Deploy New**.
4. In the **Archive** field, type `mq-install-root`/java/lib/jca/wmq.jmsra.rar
5. In the **Plan** field, type the path to the plan file you created in Step 1.
6. Optional: If you receive the following error: `HTTP Status 403 - The request body was too large to be cached during the authentication process`, you must increase the maximum post size. On the WebSphere Application Server CE administration console click **Server** > **Web Server** > **Tomcat Web Connector** > **Edit** and change the value of **maxPostSize** to -1 (unlimited).

What to do next

Next define a database connector to connect to the log database. For more information, see "Defining a database connector to connect to the log database" on page 171.

Deploying the IBM WebSphere MQ resource adapter on a different system from the application server:

About this task

If your WebSphere Application Server Community Edition instance is running on a different system from
the IBM WebSphere MQ queue manager that you want to connect to, perform the following steps to
deploy the IBM WebSphere MQ resource adapter

Procedure

1. Create a plan file that defines a connection to the queue manager of the source agent. The following
   example plan file defines a connection to a queue manager, QM_SATURN, that is located on a
   different system to your WebSphere Application Server Community Edition installation. The host
   name of QM_SATURN is `saturn.example.com`. The port of QM_SATURN is 1415. The channel of
   QM_SATURN is SYSTEM.DEF.SVRCONN.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<connector xmlns="http://geronimo.apache.org/xml/ns/j2ee/connector">
  <resourceadapter>
    <resourceadapter-instance>
      <resourceadapter-name>WMQ</resourceadapter-name>
      <workmanager>
        <gbean-link>DefaultWorkManager</gbean-link>
      </workmanager>
    </resourceadapter-instance>
    <outbound-resourceadapter>
      <connection-definition>
        <connectionfactory-interface>
                  javax.jms.ConnectionFactory
        </connectionfactory-interface>
        <connectiondefinition-instance>
          <name>jms/WMQFTEWebAgentConnectionFactory</name>
          <config-property-setting name="channel">
                    SYSTEM.DEF.SVRCONN
          </config-property-setting>
          <config-property-setting name="queueManager">
                    QM_SATURN
          </config-property-setting>
          <config-property-setting name="hostName">
                    saturn.example.com
          </config-property-setting>
          <config-property-setting name="port">
                    1414
          </config-property-setting>
          <config-property-setting name="transportType">
                    CLIENT
          </config-property-setting>
          <connectionmanager>
            <no-transaction />
            <no-pool/>
          </connectionmanager>
        </connectiondefinition-instance>
      </connection-definition>
    </outbound-resourceadapter>
  </resourceadapter>
</connector>
```

   To use this plan file in your environment change QM_SATURN to the name of the queue manager of
   your source agent. Change the value of the host name, port, and channel to the values for the queue
   manager of your source agent. The sections of the XML file that must be edited are highlighted in
   bold typeface.

2. Copy the file *mq-install-root*/java/lib/jca/wmq.jmsra.rar from the system where IBM WebSphere
   MQ is installed to the system where WebSphere Application Server Community Edition is installed.

3. Open the WebSphere Application Server Community Edition administration console.

4. From the **Common Console Actions** list on the Welcome page, click **Deploy New Applications** >
   Deploy New.

5. In the **Archive** field, type the path to the copy of the wmq.jmsra.rar file that you obtained.

6. In the **Plan** field, type the path to the plan file you created.

What to do next

Next define a database connector to connect to the log database. For more information, see "Defining a database connector to connect to the log database."

Defining a database connector to connect to the log database:
Before you begin

For transfer status information, the Web Gateway application requires access to a database written by a WebSphere MQ Managed File Transfer database logger. Before defining a database connector to this database, you must first set up the database and database logger. For instructions on how to set up the database and use the database logger application, see the topic "Configuring a WebSphere MQ Managed File Transfer logger" on page 133.

About this task

To access this database from within a WebSphere Application Server Community Edition environment, a database connector must be defined. To define a database connector, perform the following steps from the WebSphere Application Server Community Edition administration console:

Procedure
1. Depending on the level of WebSphere Application Server Community Edition that you are using, from the **Console Navigation** either select **Services** > **Database Pools**, or select **Resources** > **Datasources**.
2. Create a database pool using the Geronimo database pool wizard. In the **Name of Database Pool** field, type `jdbc/wmqfte-database`.
3. For the **Database Type** select either `DB2 XA` or `Oracle Thin`, as appropriate for your database.
4. Click **Next**.
5. In the **Driver jar** field, select the appropriate jar file for your database.
6. In the **Database Name** field, type the name of the database you are connecting to for transfer status information.
7. In the **User Name** field, type the user name for connecting to and authenticating with your database.
8. In the **Password** and **Confirm Password** fields, type the password for authenticating with your database.
9. In the **Server Name** field, type the host name or IP address of the host that the database driver needs to connect to.
10. In the **Port Number** field, type the port number you are using if it is not the default port.
11. Ensure that the value for **Driver Type** is 4.
12. Select `XA` from the **Transaction Type** list.
13. Click **Deploy**.

What to do next

Next define a database connector to connect to the file space database. For more information, see "Defining a database connector to connect to the file space database."

Defining a database connector to connect to the file space database:
Before you begin

Before you define this database connector you must create the database and tables that the Web gateway requires to work with file spaces. For more information, see "Setting up a database for use with file spaces" on page 165.

About this task

The Web Gateway application requires access to a database, to store information about the user file spaces that you create and use. This database can be the same database as the database used by the WebSphere MQ Managed File Transfer database logger, which is referred to in "Defining a database connector to connect to the log database" on page 171. Even if you use the same database for your file space information, you must create a second database connector as described in the following steps. To define a database connector, perform the following steps from the WebSphere Application Server Community Edition console:

Procedure

1. Depending on the level of WebSphere Application Server Community Edition that you are using, from the **Console Navigation** either select **Services** > **Database Pools**, or select **Resources** > **Datasources**.
2. Create a database pool using the Geronimo database pool wizard. In the **Name of Database Pool** field, type `jdbc/wmqfte-filespace`.
3. For the **Database Type** select either `DB2 XA` or `Oracle Thin`, as appropriate for your database.
4. Click **Next**.
5. In the **Driver jar** field, select the appropriate jar file for your database.
6. In the **Database Name** field, type the name of the database you are connecting to for file space information.
7. In the **User Name** field, type the user name for connecting to and authenticating with your database.
8. In the **Password** and **Confirm Password** fields, type the password for authenticating with your database.
9. In the **Port Number** field, type the port number you are using if it is not the default port.
10. Ensure that the value for **Driver Type** is `4`.
11. Select `XA` from the **Transaction Type** list.
12. Click **Deploy**.

What to do next

Next define a security realm. For more information, see "Defining a security realm."

Defining a security realm:
About this task

By default, for the Web Gateway application, a security realm called `WMQFTESecurityRealm` is required. Define the realm with groups named *administrators*, *employees*, and *managers*. Define at least one user for each group. To define a security realm, from the WebSphere Application Server Community Edition administration console:

Procedure

1. Select **Security** > **Security Realms** from the **Console Navigation**.
2. On the panel that is displayed, click **Add new security realm**.
3. In the **Name of Security Realm** field, type `WMQFTESecurityRealm`.
4. For the **Realm Type**:
   - If a simple setup is required then perform the following steps:
     a. Create a file that contains user and password information. The format of each line is `username=password`. For example,
        ```
        fteadmin=password1
        fteuser=password2
        ```

b. Create a file that contains group information. The format of each line is `group=user,user`. For example,

```
administrators=fteadmin
employees=fteadmin,fteuser
managers=fteuser
```

c. For the **Realm Type**, select **Properties File Realm** and click **Next**.

d. Enter the required information in the following fields.

Users File URI
The location of the properties file, created in Step 4a, that contains user and password information. Path separators must be specified as a forward slash (/) character on all platforms. The path to this file is relative to the WebSphere Application Server Community Edition installation directory.

Groups File URI
The location of a properties file, created in Step 4b, that contains group information. Path separators must be specified as a forward slash (/) character on all platforms. The path to this file is relative to the WebSphere Application Server Community Edition installation directory.

Digest Algorithm
The message digest algorithm used on the passwords. Example values are `MD5` and `SHA1`. Leave this field empty for a simple setup or if no digest algorithm is used.

Digest Encoding
The encoding to use for digest algorithms. Example values are `hex` and `base64`. This value is only used if a **Digest Algorithm** is specified. If no encoding is specified, `hex` is used.

e. Click the **Next** button. The **Advanced Configuration** panel is displayed. Leave the check boxes clear.

f. Click the **Test a login** button. On the **Test a login** panel, enter a valid user name and password for one of the users specified in the file that you defined in the **Users File URI** field. Click the **Next** button.

g. On the panel that is displayed, click the **Deploy Realm** button.

- If a more advanced setup is required, see the information in the WebSphere Application Server Community Edition documentation.

What to do next

Next update the `web.xml` file. For more information, see "Updating the web.xml file."

Updating the web.xml file:
About this task

Update the Web Gateway application `web.xml` file for your environment, using a Java SDK jar utility to complete the following steps:

Procedure

1. Extract the Web Gateway application from the supplied EAR file, , by running the following command:

2. Extract the `WEB-INF/web.xml` file from the previously extracted Web Gateway application, `com.ibm.wmqfte.web.war`, by running the following command:

   `jar -xf com.ibm.wmqfte.web.war WEB-INF/web.xml`

3. Use a text editor to edit the extracted `WEB-INF/web.xml`. Change the following parameters:

agentName
> Required. The name of the web agent that acts as the source for transfers initiated by the Web Gateway. This agent must be installed on the same system as the application server where you are deploying the Web Gateway application and run as the same user, or a user in the same group, as the application server. For information about how to create this agent, see the topic: "fteCreateWebAgent (create a WebSphere MQ Managed File Transfer web agent)" on page 480.
>
> The agent name is not case-sensitive and must conform to the IBM WebSphere MQ object naming conventions. For more information, see "Object naming conventions for WebSphere MQ Managed File Transfer" on page 664.

coordinationQMgr
> Required. The name of the coordination queue manager that is used by the Web Gateway for logging of transfer information.

fileSpaceRoot
> Optional. The root directory path for file spaces created and managed by the Web Gateway. Each file space is located in a subdirectory, under this root directory, with the same name as the file space. If you leave the value of this parameter blank, the application server home directory is used as the default file space root. If you change the value of this parameter after creating file spaces, the location of those file spaces will remain unchanged.

webGatewayName
> Required. The name of the Web Gateway that you are deploying.
>
> The name of the Web Gateway is not case-sensitive and must conform to the WebSphere MQ Managed File Transfer object naming conventions. For more information, see "Object naming conventions for WebSphere MQ Managed File Transfer" on page 664.

tempFileUploadDir
> Optional. The directory path for the storage of temporary files related to transfers initiated by the Web Gateway. The upload directory for temporary files is used to temporarily store files when they are uploaded to the Web Gateway. When the upload to the Web Gateway is complete, the web agent transfers the files from the upload directory for temporary files to the destination agent. If you do not provide a value for this parameter, the application server temporary directory (the value of java.io.tmpdir) is used.

maxTempFileUploadSpace
> Optional. The maximum amount of space, in MB, that a user is allowed for storing temporary files related to Web Gateway-initiated transfers. When a user uploads files to an agent they are temporarily stored on the file system until they have been transferred. This parameter can be used to limit the amount of space an upload user can use at any one time. If you do not provide a value for this parameter, the amount of temporary file storage available to a user is unlimited.

defaultMQMDUserID
> You must map user names to MQMD user IDs. If you do not do this, users cannot perform file transfers using the Web Gateway. There are two ways to map users to MQMD user IDs. You must perform one or both of the following actions:
> - Set this parameter to the default IBM WebSphere MQ Message Descriptor (MQMD) user ID to associate with a requesting user when there is no specific MQMD user ID defined for the user.
> - Use the Web Gateway user administration API to define mappings between users and MQMD user IDs.
>
>   For more information on defining mappings between users and MQMD user IDs, see "XML format for mapping web user ID to an MQMD user ID" on page 892, "Web Gateway administration API reference" on page 885, and "Example: Mapping web user IDs to MQMD user IDs" on page 312.

CSRFProtection

Enables CSRF (cross-site forgery request) protection in the Web Gateway. When set to `true` (case-insensitive), any POST or DELETE requests processed by the Web Gateway must specify either the `x-fte-csrf-token` HTTP header or 'csrf-token' form property. The value of the header or property must match the value of the current JSESSIONID. If you leave the value of this parameter blank or set it to any other value, the Web Gateway will not perform CSRF validation and requests are not required to include the csrf header or form property.

For more information about including the correct CSRF token in Web Gateway requests, see "HTTP headers and HTML form fields for using the Web Gateway" on page 867.

4. Update the Web Gateway application with the modified `WEB-INF/web.xml`, by running the following command:

   `jar -uf com.ibm.wmqfte.web.war WEB-INF/web.xml`

5. Update the supplied ear file with the updated Web Gateway application, by running the following command:

   `jar -uf fo02.ear com.ibm.wmqfte.web.war`

What to do next

Next update the `openejb-jar.xml` file. For more information, see "Updating the openejb-jar.xml to configure the Web Gateway to use file spaces."

Updating the openejb-jar.xml to configure the Web Gateway to use file spaces:
About this task

If you want to use the file space functionality of the Web Gateway, update the WebSphere MQ Managed File Transfer Web Gateway application `openejb-jar.xml` file for your environment. Use a Java SDK jar utility to complete the following steps:

Procedure

1. Extract the EJB jar file from the supplied EAR file, `com.ibm.wmqfte.web.ear`, by running the following command:

   `jar -xf com.ibm.wmqfte.web.ear com.ibm.wmqfte.web.mdb.jar`

   The EAR file is located in the *MQ_INSTALLATION_PATH*/mqft/web directory of the WebSphere MQ Managed File Transfer Service installation.

2. Extract the `META-INF/openejb-jar.xml` file from the previously extracted EJB jar file, `com.ibm.wmqfte.web.mdb.jar`, by running the following command:

   `jar -xf com.ibm.wmqfte.web.mdb.jar META-INF/openejb-jar.xml`

3. Use a text editor to edit the extracted `META-INF/openejb-jar.xml` file. Change the following `activation-config-property` values to match your environment:

   queueManager
   The name of the IBM WebSphere MQ queue manager that is used by the web agent.

   hostName
   The host name to use to connect to the specified IBM WebSphere MQ queue manager.

   transportType
   The connection method used to communicate with the specified IBM WebSphere MQ queue manager. The value of this property can be either `CLIENT` or `BINDINGS`.

   port The port to use to connect to the specified IBM WebSphere MQ queue manager. This property is only required if the transportType is set to `CLIENT`.

channel

The server channel to use to connect to the specified IBM WebSphere MQ queue manager. This property is only required if the transportType is set to `CLIENT`.

destination

The name of the WebSphere MQ Managed File Transfer Web Gateway queue that is used by the Web Gateway. For example, if your Web Gateway is called JUPITER.GATEWAY, set this property to SYSTEM.FTE.WEB.JUPITER.GATEWAY.

4. Update the EJB jar file with the modified META-INF/openejb-jar.xml file, by running the following command:

   `jar -uf com.ibm.wmqfte.web.mdb.jar META-INF/openejb-jar.xml`

5. Update the supplied ear file with the updated EJB jar file, by running the following command:

   `jar -uf com.ibm.wmqfte.web.ear com.ibm.wmqfte.web.mdb.jar`

What to do next

Next update the deployment plan. For more information, see "Update the deployment plan."

Update the deployment plan:
About this task

If you must deploy the Web Gateway for a non-default environment or are using your own security realm, you must either update the supplied deployment plan or provide a separate deployment plan. The supplied deployment plan is in the Web Gateway application file `com.ibm.wmqfte.web.war`, in the file `WEB-INF/geronimo-web.xml`. Update the supplied deployment plan for your environment, using a Java SDK jar utility to complete the following steps:

Procedure

1. Extract the Web Gateway application from the supplied EAR file, `fo02.ear`, by running the following command:

   `jar -xf fo02.ear com.ibm.wmqfte.web.war`

2. Extract the WEB-INF/geronimo-web.xml file from the previously extracted Web Gateway application, `com.ibm.wmqfte.web.war`, by running the following command:

   `jar -xf com.ibm.wmqfte.web.war WEB-INF/geronimo-web.xml`

3. Use a text editor to edit the extracted WEB-INF/geronimo-web.xml. The following example deployment plan shows a sample security configuration for WebSphere Application Server Community Edition:

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!-- Licensed Materials - Property of IBM  Copyright IBM Corp. 2010 All Rights Reserved.
     US Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.   -->
<web:web-app xmlns:app="http://geronimo.apache.org/xml/ns/j2ee/application-2.0"
            xmlns:client="http://geronimo.apache.org/xml/ns/j2ee/application-client-2.0"
            xmlns:conn="http://geronimo.apache.org/xml/ns/j2ee/connector-1.2"
            xmlns:dep="http://geronimo.apache.org/xml/ns/deployment-1.2"
            xmlns:ejb="http://openejb.apache.org/xml/ns/openejb-jar-2.2"
            xmlns:name="http://geronimo.apache.org/xml/ns/naming-1.2"
            xmlns:pers="http://java.sun.com/xml/ns/persistence"
            xmlns:pkgen="http://openejb.apache.org/xml/ns/pkgen-2.1"
            xmlns:sec="http://geronimo.apache.org/xml/ns/security-2.0"
            xmlns:web="http://geronimo.apache.org/xml/ns/j2ee/web-2.0.1">
  <dep:environment>
    <dep:moduleId>
      <dep:groupId>ibm</dep:groupId>
      <dep:artifactId>com.ibm.wmqfte.web.war</dep:artifactId>
      <dep:version>7.5</dep:version>
      <dep:type>car</dep:type>
    </dep:moduleId>
    <dep:dependencies>
      <dep:dependency>
        <dep:artifactId>wmq.jmsra.rar</dep:artifactId>
        <dep:type>rar</dep:type>
      </dep:dependency>
    </dep:dependencies>
  </dep:environment>
  <web:context-root>/wmq</web:context-root>
  <!--      Sample security configuration for WAS CE deployment        -->
  <!-- With the following settings, WAS must be configured as follows:    -->
```

```
<!-- 1 - A security realm must be defined called 'WMQFTESecurityRealm'         -->
<!-- 2 - For each group add a <sec:principal> element into each <sec:role>  -->
<!--     for the roles required for that group.  For example:               -->
<!-- <sec:principal class="org.apache.geronimo.security.realm.providers.GeronimoGroupPrincipal" name="[Group Name]"/> -->
<web:security-realm-name>WMQFTESecurityRealm</web:security-realm-name>
<sec:security>
  <sec:role-mappings>
    <sec:role role-name="wmqfte-admin">
      <!-- Add groups here that are to have the highest administration roles -->
      <sec:principal class="org.apache.geronimo.security.realm.providers.GeronimoGroupPrincipal"
                     name="administrators"/>
    </sec:role>
    <sec:role role-name="wmqfte-filespace-create">
      <!-- Add groups here that are to have the ability to create a file space  -->
      <sec:principal class="org.apache.geronimo.security.realm.providers.GeronimoGroupPrincipal"
                     name="managers"/>
      <sec:principal class="org.apache.geronimo.security.realm.providers.GeronimoGroupPrincipal"
                     name="administrators"/>
    </sec:role>
    <sec:role role-name="wmqfte-filespace-modify">
      <!-- Add groups here that are to have the ability to modify properties of a file space  -->
      <sec:principal class="org.apache.geronimo.security.realm.providers.GeronimoGroupPrincipal"
                     name="managers"/>
      <sec:principal class="org.apache.geronimo.security.realm.providers.GeronimoGroupPrincipal"
                     name="administrators"/>
    </sec:role>
    <sec:role role-name="wmqfte-filespace-permissions">
      <!-- Add groups here that are to have the ability to modify the user permissions of a file space  -->
      <sec:principal class="org.apache.geronimo.security.realm.providers.GeronimoGroupPrincipal"
                     name="managers"/>
      <sec:principal class="org.apache.geronimo.security.realm.providers.GeronimoGroupPrincipal"
                     name="employees"/>
    </sec:role>
    <sec:role role-name="wmqfte-filespace-delete">
      <!-- Add groups here that are to have the ability to delete a file space  -->
      <sec:principal class="org.apache.geronimo.security.realm.providers.GeronimoGroupPrincipal"
                     name="administrators"/>
    </sec:role>
    <sec:role role-name="wmqfte-agent-upload">
      <!-- Add groups here that are to have the ability to upload a file to a file space  -->
      <sec:principal class="org.apache.geronimo.security.realm.providers.GeronimoGroupPrincipal"
                     name="employees"/>
    </sec:role>
    <sec:role role-name="wmqfte-filespace-user">
      <!-- Add groups here that are to have the ability to view information from a file space  -->
      <sec:principal class="org.apache.geronimo.security.realm.providers.GeronimoGroupPrincipal"
                     name="employees"/>
    </sec:role>
    <sec:role role-name="wmqfte-audit">
      <!-- Add groups here that are to have the ability to view information from the transfer logs  -->
      <sec:principal class="org.apache.geronimo.security.realm.providers.GeronimoGroupPrincipal"
                     name="employees"/>
    </sec:role>
  </sec:role-mappings>
</sec:security>
</web:web-app>
```

Add groups into the sections of the XML file highlighted in **bold** typeface to give the groups permission to perform certain actions. For more information about Web Gateway roles, see the topic "User roles for the Web Gateway" on page 113.If you are using your own security realm update the deployment plan `web:security-realm-name` element to reference that realm and update the roles to reference a group name that is defined for the realm.

4. Optional: If you want to use a non-default context root for your Web Gateway, you can edit the `<web:context-root>` element in the `WEB-INF/geronimo-web.xml` file.

5. Update the Web Gateway application with the modified `WEB-INF/geronimo-web.xml`, by running the following command:

   `jar -uf com.ibm.wmqfte.web.war WEB-INF/geronimo-web.xml`

6. Update the supplied ear file with the updated Web Gateway application, by running the following command:

   `jar -uf fo02.ear com.ibm.wmqfte.web.war`

What to do next

Next, if you are using a non-default context root, update the deployment plan. For more information, see "Update the deployment plan for the administrative console."

Update the deployment plan for the administrative console:

About this task

You can use the Web Gateway administrative console to manage file spaces and user mappings from a web browser. For more information, see "Web Gateway administrative console" on page 303.

If you want to deploy the Web Gateway administrative console with a non-default context root, you must update the supplied deployment plan to contain the non-default context root. The supplied deployment plan is in the administrative console application file com.ibm.wmqfte.web.admin.war, in the file WEB-INF/geronimo-web.xml. Update the supplied deployment plan for your environment, using a Java SDK jar utility to complete the following steps:

Procedure

1. Extract the administrative console application from the supplied EAR file, fo02.ear, by running the following command:

   ```
   jar -xf fo02.ear com.ibm.wmqfte.web.war
   ```

2. Extract the WEB-INF/geronimo-web.xml file from the previously extracted administrative console application, com.ibm.wmqfte.web.admin.war, by running the following command:

   ```
   jar -xf com.ibm.wmqfte.web.admin.war WEB-INF/geronimo-web.xml
   ```

3. Use a text editor to edit the extracted WEB-INF/geronimo-web.xml. The following example deployment plan shows a sample security configuration for WebSphere Application Server Community Edition:

   ```
   <?xml version="1.0" encoding="UTF-8" standalone="no"?>
   <!-- Licensed Materials - Property of IBM  Copyright IBM Corp. 2010 All Rights Reserved.
        US Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.   -->
   <web:web-app xmlns:app="http://geronimo.apache.org/xml/ns/j2ee/application-2.0"
            xmlns:client="http://geronimo.apache.org/xml/ns/j2ee/application-client-2.0"
            xmlns:conn="http://geronimo.apache.org/xml/ns/j2ee/connector-1.2"
            xmlns:dep="http://geronimo.apache.org/xml/ns/deployment-1.2"
            xmlns:ejb="http://openejb.apache.org/xml/ns/openejb-jar-2.2"
            xmlns:name="http://geronimo.apache.org/xml/ns/naming-1.2"
            xmlns:pers="http://java.sun.com/xml/ns/persistence"
            xmlns:pkgen="http://openejb.apache.org/xml/ns/pkgen-2.1"
            xmlns:sec="http://geronimo.apache.org/xml/ns/security-2.0"
            xmlns:web="http://geronimo.apache.org/xml/ns/j2ee/web-2.0.1">
     <dep:environment>
       <dep:moduleId>
         <dep:groupId>ibm</dep:groupId>
         <dep:artifactId>com.ibm.wmqfte.web.admin.war</dep:artifactId>
         <dep:version>7.0.3.0</dep:version>
         <dep:type>car</dep:type>
       </dep:moduleId>
     </dep:environment>
     <web:context-root><b>wmqfteconsole</b></web:context-root>
     <web:security-realm-name>WMQFTESecurityRealm</web:security-realm-name>
   </web:web-app>
   ```

   Edit the text in the XML file that is highlighted in **bold** typeface to change the context root of the administrative console.

4. Update the Web Gateway application with the modified WEB-INF/geronimo-web.xml, by running the following command:

   ```
   jar -uf com.ibm.wmqfte.web.admin.war WEB-INF/geronimo-web.xml
   ```

5. Update the supplied ear file with the updated Web Gateway application, by running the following command:

   ```
   jar -uf fo02.ear com.ibm.wmqfte.web.war
   ```

Preparing to deploy the Web Gateway with WebSphere Application Server Version 7.0

Use these instructions to define required resources before deploying the WebSphere MQ Managed File Transfer Web Gateway enterprise application to WebSphere Application Server Version 7.0. You must customize the example deployment plan for your environment.

Before you begin

Before configuring or using the Web Gateway, refer to "Scenarios for the Web Gateway" on page 278 and "How the Web Gateway fits into your WebSphere MQ Managed File Transfer topology" on page 280. These topics explain the purpose and components of the Web Gateway.

To check that you are using an application server version that is supported for use with the Web Gateway, refer to the web page IBM WebSphere MQ System Requirements.

Note: The user that your application server runs as must be the same as, or in the same group as, the user that your web agent runs as.

Before starting your application server setup, complete the following tasks to prepare your IBM WebSphere MQ environment for working with the Web Gateway.

Determine which user ID the application server uses to connect to IBM WebSphere MQ. This user ID must be given the **Set identity context** permission in your IBM WebSphere MQ environment. For example, if the application server is running as appuser1, who is a member of group appgrp, and connecting to a local IBM WebSphere MQ queue manager called qm1 using a bindings mode connection, then run the following command:

```
setmqaut -m qm1 -g appgrp +setid -t qmgr
```

You must also give the user ID the **Set identity context** permission on the web agent command queue. For example, if the application server is running as appuser1, who is a member of group appgrp, and the web agent is called WEBAGENT and it connects to a local IBM WebSphere MQ queue manager called qm2 using a bindings mode connection, then run the following command:

```
setmqaut -m qm2 -g appgrp +setid -t queue -n SYSTEM.FTE.COMMAND.WEBAGENT
```

About this task

Before deploying the Web Gateway application, you must carry out the following tasks to set up the application server environment. For transfer status information, the Web Gateway application requires access to a database that is written to by a WebSphere MQ Managed File Transfer database logger. See "Configuring a WebSphere MQ Managed File Transfer logger" on page 133 for instructions on how to set up the database and use the database logger application. To access this database from within a WebSphere Application Server Version 7.0 environment you must define a Java Database Connectivity (JDBC) provider and data source.

The Web Gateway also requires a WebSphere MQ Managed File Transfer web agent installed on the same system as the application and run as the same user, or a user in the same group, as the application server. For instructions on how to create and configure this agent, see "fteCreateWebAgent (create a WebSphere MQ Managed File Transfer web agent)" on page 480.

Note: Several times during the following steps, the WebSphere Application Server Version 7.0 administrative console prompts you to save your configuration. When you are prompted, save your configuration.

Procedure

1. If you plan to connect the Web Gateway or web agent to a queue manager in bindings mode, you must set the native library path. For information about how to set the native library path in WebSphere Application Server Version 7.0, see "Setting the native library path in WebSphere Application Server Version 7.0" on page 182.
2. Enable the correct level of security in WebSphere Application Server Version 7.0. To do this perform the following steps:
   a. Select **Security** > **Global security**.

b. Ensure that **Enable administrative security** is selected.

c. Ensure that **Enable application security** is selected.

d. Ensure that **Use Java 2 security to restrict application access to local resources** is not selected.

e. Click **Apply**.

3. Define a JNDI queue connection factory:

a. Select **Resources** > **JMS** > **Queue connection factories** from the WebSphere Application Server Version 7.0 administration console navigation.

b. Select the **Scope** drop-down list and change the scope to the appropriate value. For example, `Node=yourNode, Server=yourServer`.

c. Create a queue connection factory using the console wizard, by clicking **New**.

d. Select **WebSphere MQ messaging provider**, and click **OK**.

e. At Step 1 of the wizard, in the **Name** field, enter `WMQFTEWebAgentConnectionFactory` and in the **JNDI name** field, enter `jms/WMQFTEWebAgentConnectionFactory`. Click **Next**.

f. At Step 2 of the wizard, select **Enter all the required information into this wizard**, and click **Next**.

g. At Step 2.1 of the wizard, in the **Queue manager or queue sharing group name** field, enter the name of the queue manager that the Web Gateway agent connects to, and click **Next**.

h. At Step 2.2 of the wizard, enter the connection details of the queue manager that the Web Gateway agent connects to, and click **Next**.

i. At Step 3 of the wizard, click **Test Connection**. Click **Next**.

j. At Step 4 of the wizard, review the summary information and click **Finish**.

k. On the **Queue connections factories** panel, select the resource you created.

l. In the **Advanced** section, ensure that the **Support distributed two phase commit protocol** check box is selected.

   Note: Ensure you have completed this step before proceeding. Failure to do so can cause the Web Gateway to fail to operate correctly.

4. Define a JNDI queue:

a. Select **Resources** > **JMS** > **Queues** from the WebSphere Application Server Version 7.0 administration console navigation.

b. Select the **Scope** drop-down list and change the scope to the appropriate value. For example, `Node=yourNode, Server=yourServer`.

c. Create a queue using the console wizard, by clicking **New**.

d. Select **WebSphere MQ messaging provider**, and click **OK**.

e. At Step 1 of the wizard, in the **Name** field, enter `WMQFTEWebAgentRequestQueue`. In the **JNDI name** field, enter `jms/WMQFTEWebAgentRequestQueue`. In the **Queue name** field, enter `SYSTEM.FTE.WEB.`*gateway_name*. The variable *gateway_name* is the name that you choose to give to the Web Gateway instance. In the **Queue manager or queue sharing group name** field, enter the name of the queue manager that the Web Gateway agent connects to, and click **OK**.

5. Define an activation specification:

a. Select **Resources** > **JMS** > **Activation specification** from the WebSphere Application Server Version 7.0 administration console navigation.

b. Select the **Scope** dropdown list and change the scope to the appropriate value. For example, `Node=yourNode, Server=yourServer`.

c. Create an activation specification using the console wizard, by clicking **New**.

d. Select **WebSphere MQ messaging provider**, and click **OK**.

e. At Step 1 of the wizard, in the **Name** field, enter `WMQFTEActivationSpec` and in the **JNDI name** field, enter `jms/WMQFTEActivationSpec`. Click **Next**.

f. At Step 1.1 of the wizard, in the **Destination JNDI name** field, enter `jms/WMQFTEWebAgentRequestQueue`, from the **Destination type** dropdown list, select `Queue`, and click **Next**.

g. At Step 2 of the wizard, select **Enter all the required information into this wizard**, and click **Next**.

h. At Step 2.1 of the wizard, in the **Queue manager or queue sharing group name** field, enter the name of the queue manager that the Web Gateway agent connects to, and click **Next**.

i. At Step 2.2 of the wizard, enter the connection details of the queue manager that the Web Gateway agent connects to, and click **Next**.

j. At Step 3 of the wizard, click **Test Connection**. Click **Next**.

k. At Step 4 of the wizard, review the summary information and click **Finish**.

l. Click the name of the Activation Specification that you have just created. In the Additional Properties section of the **Configuration** tab, click **Advanced Properties**. In the Connection Consumer section of the Advanced Properties panel, enter 1 into the **Maximum server sessions** field.

> **Note:** Ensure you have completed this step before proceeding. Failure to do so can cause the Web Gateway to fail to operate correctly.

6. Define a JDBC provider. If you have already deployed a JEE database logger, this data source is already defined at your selected scope.

   a. Select **Resources** > **JDBC** > **JDBC Providers** from the WebSphere Application Server Version 7.0 administration console navigation.

   b. Select the **Scope** dropdown list and change the scope to the appropriate value. For example, `Node=yourNode, Server=yourServer`.

   c. Create a JDBC provider using the console wizard, by clicking **New**.

   d. At Step 1 of the wizard, the values you provide depend on the type of database you are using.
      - If you are using Db2, select **DB2** from the **Database type** list, **DB2 Universal JDBC Driver Provider** from the **Provider type** list and **XA Data Source** from the **Implementation type** list. Click **Next**.
      - If you are using Oracle, select **Oracle** from the **Database type** list, **Oracle JDBC Driver** from the **Provider type** list and **XA Data Source** from the **Implementation type** list. Click **Next**.

   e. At Step 2 of the wizard, ensure that the directory location of the required database jar files is set correctly. Click **Next**.

   f. Click **Finish** on the summary page to create the JDBC provider.

7. Define a data source, so that the Web Gateway application can retrieve transfer status information. If you have already deployed a JEE database logger, this data source is already defined at your selected scope.

   a. Select **Resources** > **JDBC** > **Data sources** from the WebSphere Application Server Version 7.0 administration console navigation.

   b. Select the **Scope** dropdown list and change the scope to the appropriate value. For example, `Node=yourNode, Server=yourServer`.

   c. Create a data source using the console wizard, by clicking **New**.

   d. At Step 1 of the wizard, in the **Data source name** field, enter `wmqfte-database` and in the **JNDI name** field, enter `jdbc/wmqfte-database`. Click **Next**.

   e. At Step 2 of the wizard, use the **Select an existing JDBC provider** dropdown list to select the JDBC provider created in the previous steps. Click **Next**.

   f. **Db2:** At Step 3 of the wizard, in the **Driver type** field, enter 4.

   g. **Db2:** Enter the required details in the **Database name**, **Server name**, and **Port number** fields, and click **Next**.

Oracle: Enter the required connection URL in the **URL** field and choose the correct data store helper in the **Data store helper class name** field.

    h. At Step 4 of the wizard, if you have configured the authentication on your database, supply the required **Component-managed authentication alias** and **Container-managed authentication alias** in the respective dropdown boxes, and click **Next**.

    i. Click **Finish** on the summary page to create the data source.

8. Define a second data source, so that the Web Gateway application can store information about the user file spaces that you create and use:

    a. Create the database and database tables that are required to work with file spaces. For more information, see "Setting up a database for use with file spaces" on page 165.

    b. Repeat steps 7a to 7i, but for step 7d type `wmqfte-filespace` into the **Data source name** field and `jdbc/wmqfte-filespace` into the **JNDI name** field, and click **Next**.

9. Optional: If you have already configured your database you can verify the configuration of the data sources:

    a. Select **Resources** > **JDBC** > **Data sources** from the WebSphere Application Server Version 7.0 administration console navigation.

    b. Click the **Test Connection** button.

Results

You can now deploy the Web Gateway EAR file to the application server. Carry out the steps in the topic "Deploying the Web Gateway with WebSphere Application Server Version 7.0" on page 184.

Setting the native library path in WebSphere Application Server Version 7.0

If you deploy the Web Gateway application or the Java Platform, Enterprise Edition database logger application on WebSphere Application Server Version 7.0, and you want to use bindings mode connections between the application and WebSphere MQ, you must configure the WebSphere MQ messaging provider with the location of the WebSphere MQ native libraries on the system.

About this task

If you do not set the native library path in your application server, you might receive the following error message in the WebSphere Application Server Version 7.0 system out log:

```
A connection could not be made to WebSphere MQ for the following reason:
  CC=2;RC=2495;AMQ8568: The native JNI library 'mqjbnd' was not found. [3=mqjbnd]
```

Use the WebSphere Application Server Version 7.0 administrative console to complete the following steps:

Procedure

1. In the navigation pane, expand **Resources** > **JMS** > **JMS Providers**.
2. Select the WebSphere MQ messaging provider that is at the correct scope for the connection factory or activation specification that creates the bindings mode connection.

   Note: Native path information at `Server` scope is used in preference to native path information at higher scopes, and native path information at `Node` scope is used in preference to native path information at `Cell` scope.
3. Under General Properties, in the **Native library path** field, enter the full name of the directory that contains the WebSphere MQ native libraries. For example, on Linux enter `/opt/mqm/java/lib`. Enter only one directory name.
4. Click **OK**.
5. Restart the application server to refresh the configuration.
6. Required: Restart the application server a second time to load the libraries.

Deploying the WebSphere MQ Managed File Transfer Web Gateway

The WebSphere MQ Managed File Transfer Web Gateway SupportPac must be deployed to an application server that is compatible with Java Platform, Enterprise Edition 5. The deployment process for different application servers varies. This section outlines the deployment process for two application servers.

Related tasks:

"Deploying the Web Gateway with WebSphere Application Server Version 7.0" on page 184
Use these instructions to deploy the Web Gateway enterprise application to WebSphere Application Server Version 7.0.

"Deploying the Web Gateway with WebSphere Application Server Community Edition"
Use these instructions to deploy the WebSphere MQ Managed File Transfer Web Gateway enterprise application to WebSphere Application Server Community Edition. Customize the example deployment plan for your environment.

Deploying the Web Gateway with WebSphere Application Server Community Edition

Use these instructions to deploy the WebSphere MQ Managed File Transfer Web Gateway enterprise application to WebSphere Application Server Community Edition. Customize the example deployment plan for your environment.

Before you begin

Before configuring or using the Web Gateway, refer to "Scenarios for the Web Gateway" on page 278 and "How the Web Gateway fits into your WebSphere MQ Managed File Transfer topology" on page 280. These topics explain the purpose and components of the Web Gateway.

Before deploying the Web Gateway application, you must carry out the tasks described in the topic "Preparing to deploy the Web Gateway with WebSphere Application Server Community Edition" on page 167.

About this task

To deploy the EAR file to the application server, carry out the following steps from the WebSphere Application Server Community Edition administration console.

Procedure

1. From the **Console Navigation**, select **Applications** > **Deploy New**.
2. In the **Archive** field, specify the EAR file: `fo02.ear`
3. In the **Plan** field, either specify your own deployment plan file, or leave the value blank to choose the default deployment plan `geronimo-web.xml`.
4. Ensure that **Start application after install** is selected.
5. Click **Install**. The Web Gateway application is installed and started.

Results

You can now start to use the Web Gateway, for example by deploying a web application that uses the Web Gateway to submit file transfers and transfer status requests. To use the sample application provided with the Web Gateway, follow the instructions in the topic "Sample web page" on page 327.

To check your Web Gateway installation, use the installation verification application that is provided with the Web Gateway. For instructions, see "Verifying your Web Gateway installation" on page 187.

Related tasks:
"Enabling trace with WebSphere Application Server Community Edition" on page 383
If the Web Gateway application is running in WebSphere Application Server Community Edition, follow
these instructions to enable trace of the Web Gateway application. Trace is produced by the Web Gateway
application when it receives and processes requests.

Deploying the Web Gateway with WebSphere Application Server Version 7.0

Use these instructions to deploy the Web Gateway enterprise application to WebSphere Application
Server Version 7.0.

Before you begin

Before deploying the Web Gateway application, you must follow the instructions in the topic "Preparing
to deploy the Web Gateway with WebSphere Application Server Version 7.0" on page 178 to set up the
application server environment.

About this task

Before configuring or using the Web Gateway, refer to "Scenarios for the Web Gateway" on page 278 and
"How the Web Gateway fits into your WebSphere MQ Managed File Transfer topology" on page 280.
These topics explain the purpose and components of the Web Gateway.

Procedure

1. From the WebSphere Application Server Version 7.0 administration console, select **Applications** >
   New Application.
2. From the options list, select **New Enterprise Application**.
3. On the **Preparing for the application installation** page, select the `com.ibm.wmqfte.web.ear` file from
   the `MQ_INSTALLATION_PATH`/mqft/web directory of the WebSphere MQ Managed File Transfer Server
   installation, and click **Next**.
4. On the following screen, select **Detailed** to show all installation options and parameters, and click
   Next.
5. Click **Next** in each of steps 1 - 5 to accept the default values.
6. In step 6 (**Initialize parameters for servlets**), supply values for the following parameters:

   agentName
   > The name of the WebSphere MQ Managed File Transfer agent that acts as the source for Web
   > Gateway-initiated transfers. This agent must be configured as a web agent and be installed
   > on the same system as the application server where you are deploying the Web Gateway
   > application. You must provide a value for this parameter.
   >
   > You must create a web agent, it is not created by the deployment process. For information
   > about how to create a web agent, see "fteCreateWebAgent (create a WebSphere MQ Managed
   > File Transfer web agent)" on page 480.

   coordinationQMgr
   > The name of the coordination queue manage that is used by the Web Gateway for logging of
   > transfer information. You must provide a value for this parameter.

   fileSpaceRoot
   > The root directory path for file spaces created and managed by the Web Gateway. Each file
   > space is located in a subdirectory, under this root directory, with the same name as the file
   > space. If you change the value of this parameter after creating file spaces, the location of
   > those file spaces remains unchanged. If you leave the value of this parameter blank, the
   > application server home directory is used as the default file space root.
   >
   > **Note:** Use a new, empty directory as your file space root.

webGatewayName

The name of the Web Gateway that you are deploying. You must provide a value for this parameter.

The name of the Web Gateway is not case-sensitive and must conform to the WebSphere MQ object naming conventions. For more information, see "Object naming conventions for WebSphere MQ Managed File Transfer" on page 664.

tempFileUploadDir

The directory path for the storage of temporary files related to Web Gateway-initiated transfers. The temporary file upload directory is used to temporarily store files when they are uploaded to the Web Gateway. When the upload to the Web Gateway is complete, the web agent transfers the files from the temporary file upload directory to the destination agent. If you do not provide a value for this parameter, the application server temporary directory (the value of java.io.tmpdir) is used.

maxTempFileUploadSpace

The maximum amount of space, in MB, that a user is allowed for storing temporary files related to Web Gateway-initiated transfers. When a user uploads files to an agent they are temporarily stored on the file system until they have been transferred. This parameter can be used to limit the amount of space an upload user can use at any one time. If you do not provide a value for this parameter, the amount of temporary file storage available to a user is unlimited.

defaultMQMDUserID

The default WebSphere MQ Message Descriptor (MQMD) user ID to associate with a requesting user when there is no specific MQMD user ID defined for the user. You can define mappings between users and MQMD user IDs by using the MQMFT Web Gateway user administration API. If you do not provide a value for this parameter, then a user that does not have an MQMD user ID defined cannot perform a file upload.

For more information about defining mappings between users and MQMD user IDs, see the topics "Web Gateway administration API reference" on page 885 and "Example: Mapping web user IDs to MQMD user IDs" on page 312.

▶ V 7.5.0.1 CSRFProtection

Enables CSRF (cross-site forgery request) protection in the Web Gateway. When set to `true`(case-insensitive), any POST or DELETE requests processed by the Web Gateway must specify either the `x-fte-csrf-token` HTTP header or 'csrf-token' form property. The value of the header or property must match the value of the current JSESSIONID. If you leave the value of this parameter blank or set it to any other value, the Web Gateway will not perform CSRF validation and requests are not required to include the csrf header or form property.

For more information about including the correct CSRF token in Web Gateway requests, see "HTTP headers and HTML form fields for using the Web Gateway" on page 867.

Note: If you want, you can change these values after deployment without redeploying the application. To change these values, go to **Applications** > **Application types** > **WebSphere enterprise applications** > **WebSphere MQ FTE Web Gateway** > **Initialize parameters for servlets**.

7. Click **Next**.

8. In step 7 (**Bind listeners for message-driven beans**), in the **Target Resource JNDI Name** field, enter `jms/WMQFTEActivationSpec`. Click **Next**.

9. Click **Next** in each of steps 8 - 10 to accept the default values.

10. In step 11 (**Map resource references to resources**) perform the following steps:

   a. For both items in the **javax.jms.QueueConnectionFactory** section, in the **Target Resource JNDI Name** field, enter `jms/WMQFTEWebAgentConnectionFactory`.

b. In the **javax.sql.DataSource** section, locate the entry where the **Resource Reference** field has a value of jdbc/wmqfte-filespace. In the **Target Resource JNDI Name** field, enter jdbc/wmqfte-filespace.

c. In the **javax.sql.DataSource** section, locate the entry where the **Resource Reference** field has a value of jdbc/wmqfte-database. In the **Target Resource JNDI Name** field, enter jdbc/wmqfte-database.

Click **Next**.

11. Click **Next** in each of steps 12 - 13 to accept the default values.

12. In step 14 (**Map security roles to users or groups**) map the required users or groups to the roles defined in the enterprise application. For example:

a. Select wmqfte-admin, wmqfte-filespace-create, wmqfte-filespace-modify, and wmqfte-filespace-delete from the table.

b. Click **Map groups**.

c. Click **Search**.

d. Select the group administrators from the list and click the first arrow button.

e. Click **OK**.

f. Select wmqfte-filespace-create, wmqfte-filespace-modify, and wmqfte-filespace-permissions from the table.

g. Click **Map groups**.

h. Click **Search**.

i. Select the group managers from the list and click the first arrow button.

j. Click **OK**.

k. Select wmqfte-filespace-permissions, wmqfte-agent-upload, wmqfte-filespace-user, and wmqfte-audit from the table.

l. Click **Map groups**.

m. Click **Search**.

n. Select the group employees from the list and click the first arrow button.

o. Click **OK**.

For more information about Web Gateway roles, see "User roles for the Web Gateway" on page 113. Click **Next**.

13. Optional: If you want to use a non-default context root for your Web Gateway, in step 13 (**Map context roots for Web modules**), you can change the context root of the Web Gateway.

14. Optional: If you want to use a non-default context root for your Web Gateway administrative console, in step 13 (**Map context roots for Web modules**), you can change the context root of the administrative console.

15. Click **Finish** on the summary page to install the enterprise application.

16. You can now start the application from the WebSphere Application Server Version 7.0 administration console:

a. Select **Applications** > **Application Types** > **WebSphere enterprise applications** from the console navigation.

b. Select the check box for the **Web Gateway** enterprise application from the collection table, and click **Start**.

Results

You can now start to use the Web Gateway, for example by deploying a web application that uses the Web Gateway to submit file transfers and transfer status requests. To use the sample application provided with the Web Gateway, follow the instructions in the topic "Sample web page" on page 327.

To check your Web Gateway installation, use the installation verification application that is provided with the Web Gateway. For instructions, see "Verifying your Web Gateway installation."

Related tasks:

"Enabling trace with WebSphere Application Server Version 7.0" on page 384
If the Web Gateway application is running in WebSphere Application Server Version 7.0, follow these instructions to enable trace of the Web Gateway application. Trace is produced by the Web Gateway application when it receives and processes requests.

Configuring the database logger for use with the Web Gateway

The following example shows the result of requesting the status of a transfer when the database logger is not correctly configured.

About this task

1. This HTTP request submits a transfer query:

   ```
   GET HTTP/1.1 /transfer/414d51204d554e474f2afed834435bc6edaf323520204cee
   Host: example.com
   User-Agent: mozilla
   ```

2. The Web Gateway returns an HTTP response with the following format:

   ```
   HTTP/1.1 500 Internal Server Error
   Server: WAS/6.0
   Content-length: 93
   Content-type: text/plain

   BFGWI0018E: The request could not be completed due to an internal
   web application server error.
   ```

To configure the database logger so that the request is processed correctly, perform the following steps:

Procedure

1. Install the WebSphere MQ Managed File Transfer database logger. For more information on how to install and configure the database logger, see "Configuring a WebSphere MQ Managed File Transfer logger" on page 133.

2. If you already have the WebSphere MQ Managed File Transfer database logger installed, ensure that your database tables are up to date. Use the SQL files provided in the following directories to update your database tables:

   - On distributed platforms: *MQ_INSTALLATION_PATH*/mqft/sql

Related tasks:

"Installing the WebSphere MQ Managed File Transfer JEE database logger" on page 152
Follow these instructions to install and configure the JEE database logger.

"Installing the WebSphere MQ Managed File Transfer stand-alone database logger" on page 141
Complete these steps to install and configure the stand-alone database logger.

Verifying your Web Gateway installation

Follow these instructions to check that your WebSphere MQ Managed File Transfer Web Gateway application is deployed correctly.

Before you begin

Before verifying your Web Gateway configuration, you must follow the instructions to deploy the Web Gateway application. See "Configuring the Web Gateway" on page 165.

About this task

Procedure

1. Ensure that you are logged on to the application server environment with a user ID that has the `wmqfte-admin` security role. For more information, see "User roles for the Web Gateway" on page 113.
2. In a web browser, type the following URI:

   `http://host/wmqfte/ivt?logdbschema=FTELOG&webdbschema=FTEWEB`

   If you defined a context root for the Web Gateway application other than the default value of `wmqfte`, use the following URI:

   `http://host/context_root/ivt?logdbschema=FTELOG&webdbschema=FTEWEB`

   Note: During configuration of the Web Gateway, you set up database tables for storing information about file spaces and transfer history. The Web Gateway installation verification application assumes that you used the default values for the database schema names. If you defined database schema names other than the default values of FTELOG for the transfer history database and FTEWEB for the file space information database, you must change the schema names that are specified in the URI. Use the following query terms to specify the database schema names:

   logdbschema
   > Schema name for the transfer history database

   webdbschema
   > Schema name for the file space information database

   For example, if your transfer history database has a schema name of MYLOG and your file space information database has a schema name of MYWEB, use the following URI:

   `http://host/wmqfte/ivt?logdbschema=MYLOG&webdbschema=MYWEB`

   For more information about setting up databases, see "Setting up a database for use with file spaces" on page 165 and "Configuring the database logger for use with the Web Gateway" on page 187.

Results

The web browser displays a page that lists configuration information for your Web Gateway installation, and the results of testing some basic Web Gateway functions. For more information, see "The Web Gateway installation verification application."

The Web Gateway installation verification application

WebSphere MQ Managed File Transfer provides a Web Gateway installation verification application. Use this application to view configuration values for your Web Gateway installation and test basic Web Gateway functions.

For information about how to access the installation verification application, see "Verifying your Web Gateway installation" on page 187. The application displays two types of information: configuration values for your Web Gateway installation, and the results of testing basic Web Gateway functions.

Configuration values

When you deploy the Web Gateway in an application server, you provide values for several initialization parameters. If you are using WebSphere Application Server Version 7.0, you provide these values using the **Initialize parameters for servlets** step in the administration console. If you are using WebSphere Application Server Community Edition, you set these values in the `web.xml` file.

Under the heading **Web Gateway configuration information**, the application lists the values for the following Web Gateway settings:

Servlet information
> The name and version of the Web Gateway servlet that you have deployed.

Web Gateway name
> The name of the Web Gateway that you deployed. You provided this value for the `webGatewayName` initialization parameter.

Context root
> The context root that you defined for the Web Gateway application. In WebSphere Application Server Community Edition, this is the value of the `<web:context-root>` element in the `WEB-INF/geronimo-web.xml` file. In WebSphere Application Server Version 7.0, this value is set in the **Map context roots for Web modules** step when you install the Web Gateway application. The default value is `wmqfte`.

File space root directory
> The root directory path for file spaces created and managed by the Web Gateway. You provided this value for the `fileSpaceRoot` initialization parameter.

Temporary file upload root directory
> The directory path for the storage of temporary files related to Web Gateway-initiated transfers. You provided this value for the `tempFileUploadDir` initialization parameter.

Maximum size of temporary file upload directory
> The maximum amount of space, in MB, that a user is allowed for storing temporary files related to Web Gateway-initiated transfers. You provided this value for the `maxTempFileUploadSpace` initialization parameter.

MQMFT web agent name
> The name of the WebSphere MQ Managed File Transfer agent that acts as the source for Web Gateway-initiated transfers. You provided this value for the `agentName` initialization parameter. This is the name that you specified for your web agent, using the `-agentName` parameter, when you ran the `fteCreateWebAgent` command.

Coordination queue manager name
> The name of the coordination queue manager that is used by the Web Gateway for logging of transfer information. You provided this value for the `coordinationQMgr` initialization parameter.

Default MQMD user ID
> The default WebSphere MQ Message Descriptor (MQMD) user ID to associate with a requesting user when there is no specific MQMD user ID defined for the user. You provided this value for the `defaultMQMDUserID` initialization parameter.

▶V 7.5.0.1 **CSRF Protection**
> Indicates whether the Web Gateway is currently configured to perform CSRF token validation to prevent cross-site request forgery attacks. You provided this value for the `CSRFProtection` initialization parameter.

Application server information
> The name and version of the application server hosting the Web Gateway application.

Web Gateway tests

Under the heading **Results of Web Gateway tests**, the installation verification application shows the results of several tests. If a test fails, a WebSphere MQ Managed File Transfer error code and message are displayed in the **Information** column. For more information about error messages, see Diagnostic messages. The following tests are listed:

Upload file to temporary storage

Tests the directory that is named in the **Temporary file upload root directory** field. The application tests that the directory exists and is readable and writeable, and that data written to the directory can be read back.

Upload file to file space storage

Tests the directory that is named in the **File space root directory** field. The application tests that the directory exists and is readable and writeable, and that data written to the directory can be read back.

Transfer history database access

Tests that the connection to the transfer history database exists. If you are using WebSphere Application Server Version 7, the application tests the data source that you configured when deploying the Web Gateway. For more information, see "Preparing to deploy the Web Gateway with WebSphere Application Server Version 7.0" on page 178. If you are using WebSphere Application Server Community Edition, the application tests the database pool that you configured when deploying the Web Gateway. For more information, see "Preparing to deploy the Web Gateway with WebSphere Application Server Community Edition" on page 167. The application checks that the database can be accessed using the credentials that you supplied when you set up the data source or database pool.

The application also checks that the required database tables exist. For more information, see "Setting up a database for use with file spaces" on page 165 and "Configuring the database logger for use with the Web Gateway" on page 187.

The final part of the test checks that Java Persistence API (JPA) objects have been correctly defined.

File space information database access

Tests that the connection to the file space information database exists. If you are using WebSphere Application Server Version 7, the application tests the data source that you configured when deploying the Web Gateway. For more information, see "Preparing to deploy the Web Gateway with WebSphere Application Server Version 7.0" on page 178. If you are using WebSphere Application Server Community Edition, the application tests the database pool that you configured when deploying the Web Gateway. For more information, see "Preparing to deploy the Web Gateway with WebSphere Application Server Community Edition" on page 167. The application checks that the database can be accessed using the credentials that you supplied when you set up the data source or database pool.

The application also checks that the required database tables exist. For more information, see "Setting up a database for use with file spaces" on page 165 and "Configuring the database logger for use with the Web Gateway" on page 187.

The final part of the test checks that Java Persistence API (JPA) objects have been correctly defined.

Configuring the Connect:Direct bridge

Configure the Connect:Direct bridge to transfer files between a WebSphere MQ Managed File Transfer network and a Connect:Direct network. The components of the Connect:Direct bridge are a Connect:Direct node and a WebSphere MQ Managed File Transfer agent that is dedicated to communicating with that node. This agent is referred to as the Connect:Direct bridge agent.

About this task

Complete the following steps to configure the Connect:Direct bridge:

Procedure

1. "Choose the operating systems for the Connect:Direct bridge agent and node."
2. "Choose and configure a Connect:Direct node."
3. "Create and configure a Connect:Direct bridge agent" on page 192.
4. "Configure the `ConnectDirectNodeProperties.xml` file to include information about the remote Connect:Direct nodes" on page 192.
5. "Configure a secure connection between the Connect:Direct bridge agent and the Connect:Direct node" on page 193.

Choose the operating systems for the Connect:Direct bridge agent and node
Before you begin

The agent and node that make up the Connect:Direct bridge must be on the same system, or have access to the same file system, for example through a shared NFS mount. This file system is used to temporarily store files during file transfers that involve the Connect:Direct bridge, in a directory defined by the **cdTmpDir** parameter. The Connect:Direct bridge agent and the Connect:Direct bridge node must be able to address this directory using the same path name. For example, if the agent and node are on separate Windows systems, the systems must use the same drive letter to mount the shared file system. The following configurations allow the agent and the node to use the same path name:

- The agent and node are on the same system, which is either running Windows or Linux for System x
- The agent is on Linux for System x, and the node is on UNIX
- The agent is on one Windows system, and the node is on another Windows system

The following configurations do not allow the agent and the node to use the same path name:

- The agent is on Linux for System x, and the node is on Windows
- The agent is on Windows, and the node is on UNIX

Consider this restriction when planning your installation of the Connect:Direct bridge.

For more details of the operating system versions supported for the Connect:Direct bridge, see the web page WebSphere MQ System Requirements.

Procedure

1. Choose a system running either Windows or Linux on System x to install the Connect:Direct bridge agent on.
2. Choose an operating system that is supported by Connect:Direct for Windows or Connect:Direct for UNIX to install the Connect:Direct bridge node on.

Choose and configure a Connect:Direct node
Before you begin

You must have a Connect:Direct node installed before following these instructions.

Procedure

1. Choose a Connect:Direct node for the WebSphere MQ Managed File Transfer agent to communicate with.
2. Check the network map for your chosen Connect:Direct node. If the network map contains any entries for remote nodes running on a Windows operating system, you must ensure that these entries specify that the nodes are running on Windows.

a. If the Connect:Direct node that you have selected for the Connect:Direct bridge is running on Windows, use the Connect:Direct Requester to edit the network map. Ensure that the **Operating System** field for any remote nodes that are running on Windows is set to **Windows**.

Create and configure a Connect:Direct bridge agent
About this task

A Connect:Direct bridge agent is a WebSphere MQ Managed File Transfer agent that is dedicated to communicating with a Connect:Direct node.

Procedure
1. Create a Connect:Direct bridge agent using the `fteCreateCDAgent` command.
   a. You must provide a value for the `cdNode` parameter. This parameter specifies the name that the agent uses for the Connect:Direct node that is part of the Connect:Direct bridge. Use the name of the Connect:Direct node that you chose in the previous section.
   b. Provide values for the `cdNodeHost` and `cdNodePort` parameters, which define the Connect:Direct node that the agent communicates with. If you do not provide a value for the `cdNodeHost` parameter, the host name or IP address of the local system is used. If you do not provide a value for the `cdNodePort` parameter, the value 1363 is used.
   c. Optional: Use the information in "fteCreateCDAgent (create a Connect:Direct bridge agent)" on page 437 to determine whether you need to specify a value for the `cdTmpDir` parameter.
2. Map the user credentials used by WebSphere MQ Managed File Transfer to user credentials on a Connect:Direct node. You can map credentials by using one of the following methods:
   - Create a `ConnectDirectCredentials.xml` file to define credential mapping information. For more information, see "Mapping credentials for Connect:Direct by using the ConnectDirectCredentials.xml file" on page 193.
   - Write a user exit to perform credential mapping for your Connect:Direct bridge. For more information, see "Mapping credentials for Connect:Direct by using exit classes" on page 195.

Configure the `ConnectDirectNodeProperties.xml` file to include information about the remote Connect:Direct nodes
Before you begin

You must have created a Connect:Direct bridge agent before following these instructions.

Procedure

Edit the template `ConnectDirectNodeProperties.xml` in the Connect:Direct bridge agent configuration directory. For each Connect:Direct node or group of nodes that you want to define information about, perform the following steps:
1. Inside the nodeProperties element, create a node element.
2. Add a name attribute to the node element. Specify the value of this attribute as a pattern to match the name of one or more remote Connect:Direct nodes.
3. Optional: Add a pattern attribute to the node element that specifies what type of pattern the value in the name attribute is. Valid values are `regex` and `wildcard`. The default option is `wildcard`.
4. Add a type attribute to the node element that specifies the operating system that the remote Connect:Direct nodes specified by the name attribute run on. The following values are valid:
   - `Windows` - the node runs on Windows
   - `UNIX` - the node runs on UNIX or Linux

   The value of this attribute is not case sensitive. Transfers to remote nodes on other operating systems are not supported by the Connect:Direct bridge.

For more information, see "Connect:Direct node properties file format" on page 582.

Configure a secure connection between the Connect:Direct bridge agent and the Connect:Direct node

About this task

By default, the Connect:Direct bridge agent uses the TCP/IP protocol to connect to the Connect:Direct node. If you want a secure connection between your Connect:Direct bridge agent and the Connect:Direct node, you can use the SSL protocol or the TLS protocol.

Procedure

Configure a secure connection. For an example of how to do this, see "Configuring an SSL or TLS connection between the Connect:Direct bridge agent and the Connect:Direct node" on page 119.

Mapping credentials for Connect:Direct

Map user credentials in WebSphere MQ Managed File Transfer to user credentials on a Connect:Direct node by using the default credential mapping function of the Connect:Direct bridge agent or by writing your own user exit. WebSphere MQ Managed File Transfer provides a sample user exit that performs user credential mapping.

Mapping credentials for Connect:Direct by using the ConnectDirectCredentials.xml file

Map user credentials in WebSphere MQ Managed File Transfer to user credentials on Connect:Direct nodes by using the default credential mapping function of the Connect:Direct bridge agent. WebSphere MQ Managed File Transfer provides an XML file that you can edit to include your credential information.

About this task

Once a Connect:Direct bridge agent has been created using the **fteCreateCDAgent** command, a `ConnectDirectCredentials.xml` file needs to be manually created. Before you can use a Connect:Direct bridge agent, you must edit this file to include host, user, and credential information. For more information, see "Connect:Direct credentials file format" on page 579. By default, this file is loaded from the current user's home directory, `/home/fteuser/ConnectDirectCredentials.xml` for example. If you wish to use another location then you must specify it via the `<credentialsFile>` element in the `ConnectDirectNodeProperties.xml` file.

Procedure

1. Ensure that the `name` attribute in the element `<tns:pnode name="Connect:Direct node host" pattern="wildcard">` contains the value of the name of the Connect:Direct node that the Connect:Direct bridge agent connects to. This value must be the same value that you specify for the **fteCreateCDAgent -cdNode** parameter. The value of the `pattern` attribute can be either `wildcard` or `regex`. If this attribute is not specified, the default is `wildcard`.

2. Insert user ID and credential information into the file as child elements of `<tns:pnode>`. You can insert one or more instances of the following `<tns:user>` element into the file:

```
<tns:user name="name"
          pattern="pattern"
          ignorecase="ignorecase"
          cdUserId="cdUserId"
          cdPassword="cdPassword"
          pnodeUserId="pnodeUserId"
          pnodePassword="pnodePassword">
</tns:user>
```

where:

- *name* is a pattern to match the MQMD user ID associated with the MQMFT transfer request.
- *pattern* specifies whether the pattern specified for the `name` attribute is a wildcard expression or a Java regular expression. The value of the `pattern` attribute can be either `wildcard` or `regex`. If this attribute is not specified, the default is `wildcard`.
- *ignorecase* specifies whether to treat the pattern specified by the `name` attribute as case sensitive. If this attribute is not specified, the default is `true`.
- *cdUserId* is the user ID that is used by the Connect:Direct bridge agent to connect to the Connect:Direct node specified by the `name` attribute of `<tns:pnode>` element. If possible, ensure that *cdUserId* is a Connect:Direct administrator user ID. If *cdUserId* cannot be a Connect:Direct administrator, ensure that the user ID has the following functional authorities at the Connect:Direct bridge node:
  - For a Windows node set the following authorities. This example is formatted with carriage returns to aid readability:

    ```
    View Processes in the  value: yes
     TCQ
    Issue the copy         value: yes
     receive, copy send,
     run job, and run task
     Process statements
    Issue the submit       value: yes
     Process statement
    Monitor, submit,       value: all
     change,and delete all
     Processes
    Access Process         value: all
     statistics
    Use the trace tool or  value: yes
     issue traceon and
     traceoff commands
    Override Process       value: yes
     options such as file
     attributes and remote
     node ID
    ```
  - For a UNIX node set the following parameters in the `userfile.cfg` file:

    ```
    pstmt.copy             value: y
    pstmt.upload           value: y
    pstmt.download         value: y
    pstmt.runjob           value: y
    pstmt.runtask          value: y
    cmd.submit             value: y
    pstmt.submit           value: y
    cmd.chgproc            value: y
    cmd.delproc            value: y
    cmd.flsproc            value: y
    cmd.selproc            value: a
    cmd.selstats           value: a
    cmd.trace              value: y
    snode.ovrd             value: y
    ```
- *cdPassword* is the password associated with the user ID specified by the `cdUserId` attribute.
- You can optionally specify the `pnodeUserId` attribute. The value of this attribute is the user ID that is used by the Connect:Direct node specified by the `name` attribute of `<tns:pnode>` element to submit the Connect:Direct process. If you do not specify the `pnodeUserId` attribute, the Connect:Direct node uses the user ID specified by the `cdUserId` attribute to submit the Connect:Direct process.
- You can optionally specify the attribute `pnodePassword`. The value of this attribute is the password associated with the user ID specified by the `pnodeUserId` attribute.

  If no user element matches the MQMD user ID, the transfer fails.
3. Optional: You can include one or more `<tns:snode>` elements as child elements of the `<tns:user>` element. The `<tns:snode>` element specifies credentials that are used by the Connect:Direct node that

is part of the Connect:Direct bridge. These credentials are the user ID and password that the Connect:Direct bridge node uses to connect to the Connect:Direct node that is the source or destination of the file transfer. Insert one or many of the following elements into the file:

```
<tns:snode name="name"
           pattern="pattern"
           userId="userId"
           password="password"/>
```

where:

- *name* is a pattern to match the name of the Connect:Direct node that is the source or destination of the file transfer.
- *pattern* specifies whether the pattern specified for the `name` attribute is a wildcard expression or a Java regular expression. The value of the pattern attribute can be either `wildcard` or `regex`. If this attribute is not specified, the default is `wildcard`.
- *userId* is the user ID that is used by the Connect:Direct node specified by the `name` attribute of the `<tns:pnode>` element to connect to a Connect:Direct node that matches the pattern specified by the `name` attribute of `<tns:snode>`.
- *password* is the password associated with the user ID specified by the `userId` attribute.

If no `<tns:snode>` element matches the secondary node of the file transfer, this does not cause the transfer to fail. The transfer is started and no user ID and password are specified for use with the snode.

Results

When searching for a pattern match for user names or Connect:Direct node names the Connect:Direct bridge agent searches from the start of the file to the end of the file. The first match that is found is the one that is used.

Related tasks:

"Configuring the Connect:Direct bridge" on page 190
Configure the Connect:Direct bridge to transfer files between a WebSphere MQ Managed File Transfer network and a Connect:Direct network. The components of the Connect:Direct bridge are a Connect:Direct node and a WebSphere MQ Managed File Transfer agent that is dedicated to communicating with that node. This agent is referred to as the Connect:Direct bridge agent.

Related reference:

"Connect:Direct credentials file format" on page 579
The `ConnectDirectCredentials.xml` file in the agent configuration directory defines the user names and credential information that the Connect:Direct agent uses to authorize itself with a Connect:Direct node.

"fteCreateCDAgent (create a Connect:Direct bridge agent)" on page 437
The fteCreateCDAgent command creates a WebSphere MQ Managed File Transfer agent and its associated configuration for use with the Connect:Direct bridge.

Mapping credentials for Connect:Direct by using exit classes

If you do not want to use the default credential mapping function of the Connect:Direct bridge agent, you can map user credentials in WebSphere MQ Managed File Transfer to user credentials on a Connect:Direct node by writing your own user exit. Configuring your own credential mapping user exits disables the default credential mapping function.

About this task

User exits that you create for mapping Connect:Direct credentials must implement the interface `com.ibm.wmqfte.exitroutine.api.ConnectDirectCredentialExit`. For more information, see "CDCredentialExit.java interface" on page 930.

Configuring an SSL or TLS connection between the Connect:Direct bridge agent and the Connect:Direct node

Configure the Connect:Direct bridge agent and the Connect:Direct node to connect to each other through the SSL protocol by creating a keystore and a truststore, and by setting properties in the Connect:Direct bridge agent properties file.

About this task

These steps include instructions for getting your keys signed by a certificate authority. If you do not use a certificate authority, you can generate a self-signed certificate. For more information about generating a self-signed certificate, see Working with SSL or TLS on UNIX and Windows systems.

These steps include instructions for creating a new keystore and truststore for the Connect:Direct bridge agent. If the Connect:Direct bridge agent already has a keystore and truststore that it uses to connect securely to WebSphere MQ queue managers, you can use the existing keystore and truststore when connecting securely to the Connect:Direct node. For more information, see "Configuring SSL encryption for WebSphere MQ Managed File Transfer" on page 109.

Procedure

For the Connect:Direct node, complete the following steps:
1. Generate a key and signed certificate for the Connect:Direct node. You can do this by using the IBM Key Management tool that is provided with WebSphere MQ. For more information, see Working with SSL or TLS.
2. Send a request to a certificate authority to have the key signed. You receive a certificate in return.
3. Create a text file; for example, /test/ssl/certs/CAcert, that contains the public key of your certification authority.
4. Install the Secure+ Option on the Connect:Direct node. If the node already exists, you can install the Secure+ Option by running the installer again, specifying the location of the existing installation, and choosing to install only the Secure+ Option.
5. Create a new text file; for example, /test/ssl/cd/keyCertFile/*node_name*.txt.
6. Copy the certificate that you received from your certification authority and the private key, located in /test/ssl/cd/privateKeys/*node_name*.key, into the text file. The contents of /test/ssl/cd/ keyCertFile/*node_name*.txt must be in the following format:

```
-----BEGIN CERTIFICATE-----
MIICnzCCAgigAwIBAgIBGjANBgkqhkiG9w0BAQUFADBeMQswCQYDVQQGEwJHQjES
MBAGA1UECBMJSGFtcHNoaXJlMRAwDgYDVQQHEwdIdXJzbGV5MQwwCgYDVQQKEwNJ
Qk0xDjAMBgNVBAsTBU1RSVBMQswCQYDVQQDEwJDQTAeFw0xMTAzMDExNjIwNDZa
Fw0yMTAyMjYxNjIwNDZaMFAxCzAJBgNVBAYTAkdCMRIwEAYDVQQIEwlIYW1wc2hp
cmUxDDAKBgNVBAoTA0lCTTEOMAwGA1UECxMFTVFGVEUxDzANBgNVBAMTBmJpbmJh
ZzCBnzANBgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEAvgP1QIklU9ypSKD1XoODo1yk
EyMFXBOUpZRrDVxjoSEC0vtWNcJ199e+Vc4UpNybDyBu+NkDlMNofX4QxeQcLAFj
WnhakqCiQ+JIAD5AurhnrwChe0MV3kjA84GKH/rOSVqtl984mu/lDyS819XcfSSn
cOOMsK1KbneVSCIV2XECAwEAAaN7MHkwCQYDVR0TBAIwADAsBglghkgBhvhCAQ0E
HxYdT3BlblNTTCBHZW5lcmF0ZWQgQ2VydGlmaWNhdGUwHQYDVR0OBBYEFNXMIpSc
csBXUniW4A3UrZnCRsv3MB8GA1UdIwQYMBaAFDXY8rmj4lVz5+FVAoQb++cns+B4
MA0GCSqGSIb3DQEBBQUAA4GBAFc7klXa4pGKYgwchxKpE3ZF6FNwy4vBXS216/ja
8h/vl8+iv01OCL8t0ZOKSU95fyZLzOPKnCH7v+ItFSE3CIiEk9Dlz2U6WO9lICwn
l7PL72TdfaL3kabwHYVf17IVcuL+VZsZ3HjLggP2qHO9ZuJPspeT9+AxFVMLiaAb
8eHw
-----END CERTIFICATE-----
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: DES-EDE3-CBC,64A02DA15B6B6EF9

57kqxLOJ/gRUOIQ6hVK2YN13B4E1jAi1gSme0I5ZpEIG8CHXISKB7/0cke2FTqsV
lvI99QyCxsDWoMNt5fj51v7aPmVeS60bOm+UlGre8B/Ze18JVj2O4K2Uh72rDCXE
```

```
5e6eFxSdUM207sQDy20euBVELJtM2kOkL1ROdoQQSlU3XQNgjw/t3ZIx5hPXWEQT
rjRQO64BEhb+PzzxPF8uwzZ9IrUK9BJ/UUnqC6OdBR87IeA4pnJD1Jvb2ML7EN9Z
5Y+50hTKI8OGvBvWXO4fHyvIX5aslwhBoArXIS1AtNTrptPvoaP1zyIAeZ6OCVo/
SFo+A2UhmtEJeOJaZG2XZ3H495fAw/EHmjehzIACwukQ9nSIETgu4A1+CV64RJED
aYBCM8UjaAkbZDH5gn7+eBov0ssXAXWDyJBVhUOjXjvAj/e1h+kcSF1hax5D//AI
66nRMZzboSxNqkjcVd8wfDwP+bEjDzUaaarJTS7lIFeLLw7eJ8MNAkMGicDkycL0
EPBU9X5QnHKLKOfYHN/1WgUk8qt3UytFXXfzTXGF3EbsWbBupkT5e5+lYcX8OVZ6
sHFPNlHluCNy/riUcBy9iviVeodX8IomOchSyO5DKl8bwZNjYtUP+CtYHNFU5BaD
I+1uUOAeJ+wjQYKT1WaeIGZ3VxuNITJul8y5qDTXXfX7vxM5OoWXa6U5+AYuGUMg
/itPZmUmNrHjTk7ghT6i1IQOaBowXXKJBlMmq/6BQXN2IhkD9ys2qrvM1hdi5nAf
egmdiG50loLnBRqWbfR+DykpAhK4SaDi2F52Uxovw3Lhiw8dQP7lzQ==
-----END RSA PRIVATE KEY-----
```

7. Start the Secure+ Admin Tool.

   • On Linux or UNIX systems, run the command **spadmin.sh**.

   • On Windows systems, click **Start** > **Programs** > **Sterling Commerce Connect:Direct** > **CD Secure+ Admin Tool**

   The CD Secure+ Admin Tool starts.

8. In the CD Secure+ Admin Tool, double-click the **.Local** line to edit the main SSL or TLS settings.

   a. Select **Enable SSL Protocol** or **Enable TLS Protocol**, depending on which protocol you are using.

   b. Select **Disable Override**.

   c. Select at least one Cipher Suite.

   d. If you want two-way authentication, change the value of **Enable Client Authentication** to Yes.

   e. In the **Trusted Root Certificate** field, enter the path to the public certificate file of your certification authority, /test/ssl/certs/CAcert.

   f. In the **Key Certificate File** field, enter the path to the file that you created, /test/ssl/cd/keyCertFile/*node_name*.txt.

9. Double-click the **.Client** line to edit the main SSL or TLS settings.

   a. Select **Enable SSL Protocol** or **Enable TLS Protocol**, depending on which protocol you are using.

   b. Select **Disable Override**.

For the Connect:Direct bridge agent, perform the following steps:

10. Create a truststore. You can do this by creating a dummy key and then deleting the dummy key. You can use the following commands:

    ```
    keytool -genkey -alias dummy -keystore /test/ssl/fte/stores/truststore.jks
    ```

    ```
    keytool -delete -alias dummy -keystore /test/ssl/fte/stores/truststore.jks
    ```

11. Import the public certificate of the certification authority into the truststore. You can use the following command:

    ```
    keytool -import -trustcacerts -alias myCA
            -file /test/ssl/certs/CAcert
            -keystore /test/ssl/fte/stores/truststore.jks
    ```

12. Edit the Connect:Direct bridge agent properties file. Include the following lines anywhere in the file:

    ```
    cdNodeProtocol=protocol
    cdNodeTruststore=/test/ssl/fte/stores/truststore.jks
    cdNodeTruststorePassword=password
    ```

    In the example in this step, *protocol* is the protocol you are using, either SSL or TLS, and *password* is the password that you specified when you created the truststore.

13. If you want two-way authentication, create a key and certificate for the Connect:Direct bridge agent.

    a. Create a keystore and key. You can use the following command:

       ```
       keytool -genkey -keyalg RSA -alias agent_name
               -keystore /test/ssl/fte/stores/keystore.jks
               -storepass password -validity 365
       ```

    b. Generate a signing request. You can use the following command:

```
keytool -certreq -v -alias agent_name
        -keystore /test/ssl/fte/stores/keystore.jks -storepass password
        -file /test/ssl/fte/requests/agent_name.request
```

   c. Import the certificate you receive from the preceding step into the keystore. The certificate must be in x.509 format. You can use the following command:

```
keytool -import -keystore /test/ssl/fte/stores/keystore.jks
        -storepass password -file certificate_file_path
```

   d. Edit the Connect:Direct bridge agent properties file. Include the following lines anywhere in the file:

```
cdNodeKeystore=/test/ssl/fte/stores/keystore.jks
cdNodeKeystorePassword=password
```

In the example in this step, *password* is the password that you specified when you created the keystore.

Related tasks:

"Configuring the Connect:Direct bridge" on page 190
Configure the Connect:Direct bridge to transfer files between a WebSphere MQ Managed File Transfer network and a Connect:Direct network. The components of the Connect:Direct bridge are a Connect:Direct node and a WebSphere MQ Managed File Transfer agent that is dedicated to communicating with that node. This agent is referred to as the Connect:Direct bridge agent.

Specifying the Connect:Direct process to start by using the ConnectDirectProcessDefinition.xml file

Specify which Connect:Direct process to start as part of a WebSphere MQ Managed File Transfer transfer. WebSphere MQ Managed File Transfer provides an XML file that you can edit to specify process definitions.

About this task

The **fteCreateCDAgent** command creates the file ConnectDirectProcessDefinitions.xml in the agent configuration directory *MQ_DATA_PATH*/mqft/config/*coordination_queue_manager*/agents/ *cd_bridge_agent_name*. Before you can call user-defined Connect:Direct processes from the Connect:Direct bridge agent, you must set up process definitions by editing this file.

For each process that you want to specify to call as part of a transfer through the Connect:Direct bridge, perform the following steps:

Procedure

1. Define the Connect:Direct process that you want the Connect:Direct bridge agent to call as part of the transfer and save the process template in a file.
2. Open the *MQ_DATA_PATH*/mqft/config/*coordination_queue_manager*/agents/*cd_bridge_agent_name*/ ConnectDirectProcessDefinitions.xml file in a text editor.
3. Create a <processSet> element.
4. Inside the <processSet> element, create a <condition> element.
5. Inside the <condition> element, create one or more elements that define a condition that the transfer request must match to call the Connect:Direct process you defined in Step 1. These elements can be either <match> elements or <defined> elements.
   - Use a <match> element to specify that the value of a variable must match a pattern. Create the <match> element with the following attributes:
     - variable - the name of the variable whose value is compared. The variable is an intrinsic symbol. For more information, see "Substitution variables for use with user-defined Connect:Direct processes" on page 686.
     - value - the pattern to compare to the value of the specified variable.

- Optional: `pattern` - the type of pattern used by the value of the `value` attribute. This pattern type can be `wildcard` or `regex`. This attribute is optional and the default is `wildcard`.
- Use a `<defined>` element to specify that a variable must have a value defined. Create the `<defined>` element with the following attribute:
  - `variable` - the name of the variable that must have a value defined. The variable is an intrinsic symbol. For more information, see "Substitution variables for use with user-defined Connect:Direct processes" on page 686.

The conditions specified within the `<condition>` element are combined with a logical AND. All conditions must be met for the Connect:Direct bridge agent to call the process specified by this `<processSet>` element. If you do not specify a `<condition>` element, the process set matches all transfers.

6. Inside the `<processSet>` element, create a `<process>` element.
7. Inside the `<process>` element, create a `<transfer>` element. The transfer element specifies the Connect:Direct process that the Connect:Direct bridge agent calls as part of the transfer. Create the `<transfer>` element with the following attribute:
   - `process-` - the location of the Connect:Direct process that you defined in step 1. The location of this file is specified with an absolute path or relative to the *MQ_DATA_PATH*/mqft/config/ *coordination_queue_manager*/agents/*cd_bridge_agent_name* directory.

Results

When searching for a condition match, the Connect:Direct bridge agent searches from the start of the file to the end of the file. The first match that is found is the one that is used.

Related tasks:

"Configuring the Connect:Direct bridge" on page 190
Configure the Connect:Direct bridge to transfer files between a WebSphere MQ Managed File Transfer network and a Connect:Direct network. The components of the Connect:Direct bridge are a Connect:Direct node and a WebSphere MQ Managed File Transfer agent that is dedicated to communicating with that node. This agent is referred to as the Connect:Direct bridge agent.

Related reference:

"Connect:Direct process definitions file format" on page 584
The `ConnectDirectProcessDefinitions.xml` file in the Connect:Direct bridge agent configuration directory specifies the user-defined Connect:Direct process to start as part of the file transfer.

"fteCreateCDAgent (create a Connect:Direct bridge agent)" on page 437
The fteCreateCDAgent command creates a WebSphere MQ Managed File Transfer agent and its associated configuration for use with the Connect:Direct bridge.

Configuring WebSphere MQ Managed File Transfer agents with MSCS

WebSphere MQ Managed File Transfer (MFT) agent MSCS setup is supported, if the platform is one supported by MFT and running one of the versions of Windows.

About this task

Complete the following steps to achieve failover of an MFT agent:

Procedure

1. Install WebSphere MQ Managed File Transfer locally on each machine in the cluster. See "How to install" on page 17 for links to the installation instructions for your platform.
2. Create the agent as normal on the primary machine, ensuring that you create all of the objects on the queue manager for this agent.

3. Set up the agent on the primary machine as a Windows service, under control of the cluster, as for the existing service. See "Starting an agent as a Windows service" on page 201 for details on how to do this.

4. Create the same agent on the other machine, but do not start the agent, or set the agent up as a service. This ensures that the file structure for logs, properties, and so on, exists on the second machine.

   If there is a failover, the agent service can transfer over to the backup machine, and continues working, with the file structure as expected.

Administering WebSphere MQ Managed File Transfer

Use WebSphere MQ Managed File Transfer commands to administer WebSphere MQ Managed File Transfer. You can also use the WebSphere MQ Explorer for some of the administrative tasks.

Starting a WebSphere MQ Managed File Transfer agent

Before you can use a WebSphere MQ Managed File Transfer agent for a file transfer, you must first start the agent.

About this task

You can start a WebSphere MQ Managed File Transfer from the command line. In this case, the agent process stops when you log off the system.

On Windows, UNIX and Linux, you can configure an agent so that it continues running when you log off from the system and can continue to receive file transfers.

▶ V 7.5.0.6 ◀ Note that, from IBM WebSphere MQ Version 7.5.0, Fix Pack 6, if an agent encounters an unrecoverable error when it is running, a first failure data capture (FDC) is generated and the agent is stopped.

Procedure
- To start an agent from the command line, use the **fteStartAgent** command. For more information, see "**fteStartAgent** (start a WebSphere MQ Managed File Transfer agent)" on page 529.
- To configure an agent so that it continues running when you log off from the system:
  - On Windows, configure the agent to run as a Windows service. For more information, see "Starting an agent as a Windows service."
  - On UNIX and Linux, configure the agent to start automatically during a reboot by using a script file. For more information, see "Starting an agent at UNIX system startup" on page 203.

Starting an agent as a Windows service

You can start an agent as a Windows service so that when you log off Windows, your agent continues running and can receive file transfers.

About this task

On Windows, when you start an agent from the command line, the agent process runs using the user name you used to log on to Windows. When you log off the system, the agent process stops. To prevent the agent stopping, you can configure an agent to run as a Windows service. Running as a Windows service also allows you to configure agents to be started automatically when the Windows environment starts or is restarted.

Complete the following steps to start an agent that runs as a Windows service. You must be running WebSphere MQ Managed File Transfer on one of the supported Windows versions to run the agent as a Windows service. For the list of supported environments, refer to the WebSphere MQ System Requirements.

The exact steps depend on whether you have already created an agent or whether you are creating an agent. Both options are described in the following steps.

Procedure

1. If you are creating an MQMFT agent, use the **fteCreateAgent**, **fteCreateWebAgent**, **fteCreateCDAgent**, or **fteCreateBridgeAgent** command. Specify the **-s** parameter to run the agent as a Windows service. In the following example, the agent AGENT1 is created, which has an agent queue manager QMGR1. The Windows service runs using a user name of `fteuser`, which has an associated password `ftepassword`.

   `fteCreateAgent -agentName AGENT1 -agentQMgr QMGR1 -s -su fteuser -sp ftepassword`

   You can optionally specify a name for the service after the **-s** parameter. If you do not specify a name, the service is named mqmftAgent<*AGENT*><*QMGR*>, where *AGENT* is the agent name you specified and *QMGR* is your agent queue manager name. In this example, the default name for the service is `mqmftAgentAGENT1QMGR1`.

   Note: The Windows user account that you specify using the **-su** parameter must have the **Log on as a service** rights. For information about how to configure this, see "Guidance for running an agent or logger as a Windows service" on page 366.

   For more information, see "fteCreateAgent (create a WebSphere MQ Managed File Transfer agent)" on page 429, "fteCreateWebAgent (create a WebSphere MQ Managed File Transfer web agent)" on page 480, "fteCreateCDAgent (create a Connect:Direct bridge agent)" on page 437, or "fteCreateBridgeAgent (create and configure WebSphere MQ Managed File Transfer protocol bridge agent)" on page 432.

2. If you followed the previous step to create an agent, run the MQSC commands that are generated by the **fteCreateAgent**, **fteCreateWebAgent**, **fteCreateCDAgent**, or **fteCreateBridgeAgent** command. These commands create the WebSphere MQ queues that are needed by the agent. For example, for an agent named *AGENT1*, an agent queue manager named *QMGR1* and a coordination queue manager named *COORDQMGR1*, run the following command:

   `runmqsc QMGR1 < MQ_DATA_PATH\mqft\config\COORDQMGR1\agents\AGENT1\AGENT1_create.mqsc`

3. If you did not follow the previous steps to create an agent and instead want to configure an existing agent to run as a Windows service, first stop your agent if it is running, and then modify its configuration.

   a. The following example uses an agent named AGENT1. Run the following command:

      `fteStopAgent AGENT1`

   b. Use the **fteModifyAgent** command to configure the agent to run as a Windows service:

      `fteModifyAgent -agentName AGENT1 -s -su fteuser -sp ftepassword`

      For more information, see "fteModifyAgent (modify a WebSphere MQ Managed File Transfer agent)" on page 507.

4. Start your agent using the **fteStartAgent** command. Alternatively, you can use the Windows Services tool, which is available from Administrative Tools in the Control Panel, selected from the Windows desktop start menu, to start the service.

   `fteStartAgent AGENT1`

   The service continues to run even if you log off Windows. To ensure that the service also restarts when Windows restarts after a shutdown, the **Startup Type** field in the Windows Services tool is set to **Automatic** by default. Change this to **Manual** if you do not want the service to restart when Windows restarts.

5. Optional: To stop the agent, either use the fteStopAgent command or use the Windows Services tool. For example, from the command line, run the following command:

   `fteStopAgent AGENT1`

   • When you run the **fteStopAgent** command as a service, the command always runs using the **-i** parameter regardless of whether you specified this parameter. The **-i** parameter stops the agent immediately without completing any transfers that are in progress. This is caused by a limitation of the Windows service.

What to do next

If you have problems starting your Windows service, see "Guidance for running an agent or logger as a Windows service" on page 366. This topic also describes the location of the Windows service log files.

Starting an agent at UNIX system startup

A WebSphere MQ Managed File Transfer agent can be configured to start at system startup on UNIX. When you log off a UNIX system your agent continues running and can receive file transfers.

When you have created and configured an agent using one of these WebSphere MQ Managed File Transfer commands; **fteCreateAgent**, **fteCreateCDAgent**, **fteCreateWebAgent**, or **fteCreateBridgeAgent**, you can configure it to start automatically during a reboot on UNIX machines by using a script file that simply executes:

```
su -l mqmft_user -c mq_install_root/bin/fteStartAgent agent_name
```

Where *mq_install_root* is the root directory of the required WebSphere MQ Managed File Transfer 7.5 installation, the default is: /opt/mqm and *agent_name* is the name of the WebSphere MQ Managed File Transfer agent to be started. The usage of this script file varies depending on the specific UNIX operating system.

Linux

For Linux systems there are multiple ways that you can start applications during the system boot process. In general, we recommend that you follow these steps:

1. Create a file called /etc/rc.mqmft with contents:

   ```
   #!/bin/sh
   su -l mqmft_user "-c mq_install_root/bin/fteStartAgent agent_name"
   ```

   Where *mqmft_user* is the user id under which the agent process is to run. This user id must be a member of the mqm group.

2. Make the file executable, for example:

   ```
   chmod 755 /etc/rc.mqmft
   ```

3. Next add the following line to /etc/inittab:

   ```
   mqmft:5:boot:/etc/rc.mqmft
   ```

Other ways to start an agent during boot on Linux include adding the script lines to the /etc/rc.d/rc.local file, or on Linux SuSe, adding the script lines to the /etc/init.d/boot.local file. You should select the method that works best for your environment. Here is some more information on other ways to start an agent during startup on specific Linux distributions that are supported:

SLES 10 and 11

For SUSE Linux Enterprise Server (SLES) 10 and 11 systems, follow these steps:

1. As the system root user id, create your own /etc/init.d/rc.rclocal file.

2. Add the following lines to the rc.rclocal file:

   ```
   #!/bin/sh
   ### BEGIN INIT INFO
   # Provides: rc.rclocal
   # Required-Start: $network $syslog
   # Required-Stop: $network $syslog
   # Default-Stop: 0 1 2 6
   # Description: MQMFT agent startup
   ### END INIT INFO
   su -l mqmft_user "-c mq_install_root/bin/fteStartAgent agent_name"
   ```

3. Run the following commands:

```
chmod 755 rc.rclocal
chkconfig --add rc.rclocal
```

Solaris

On Solaris, follow these steps:

1. Run the following command, and keep track of the path returned:

   `which sh`

   For example, the path might be /usr/bin/sh

2. As the system root user ID, create your own /etc/init.d/startmqmft file.

3. Edit this file and add the script lines to it, using the returned path from step 1 as the first line in the script:

   ```
   #!/usr/bin/sh
   su mqmft_user mq_install_root/bin/fteStartAgent agent_name
   ```

4. Make the file executable, for example:

   `chmod 755 /etc/init.d/startmqmft`

5. Symlink the file to the rc3.d directory:

   `ln -s /etc/init.d/startmqmft /etc/rc3.d/S98startmqmft`

   The prefix S means default state Started for Solaris. 98 is a sequence number. The suffix is the filename from init.d

HP-UX

On HP-UX, follow these steps:

1. Create a file called /sbin/init.d/mqmft with contents:

   ```
   #!/bin/sh
   su -l mqmft_user -c mq_install_root/bin/fteStartAgent agent_name
   ```

2. Create a file called /etc/rc.config.d/mqmft with contents:

   `MQMFT=1`

3. Symlink the file to the rc3.d directory and start the agent:

   `ln -s /sbin/init.d/mqmft /sbin/rc3.d/S84mqmft`

Starting an agent on a 4690 OS system

To start an agent, you must first configure it as a background application. Background applications can be configured to start automatically when the system becomes the acting master controller or file server, and to stop when the system is no longer the acting master controller or file server.

About this task

Agents that are available to be started are already defined by running the f:\adxetc\mft75\bin\ftecfg.bat utility to load a new configuration. For each configured agent, there is an agent_name.rsp response file in the f:\adxetc\mft75 directory. Each of these response files contains a command to start a particular agent when passed to the ADXCHAIN.386 system command.

Procedure

Complete the following steps to start an agent that runs as a background service:

1. From the SYSTEM MAIN MENU panel, select **4** (**Installation and Update Aids**).
2. From the INSTALLATION AND UPDATE AIDS panel, select **1** (**Change Configuration Data**).
3. From the CONFIGURATION panel, select **2** (**Controller Configuration**).

4. If you are asked whether you are configuring a store system that uses the IBM Multiple Controller Feature, press **Enter** to select **Yes**.

5. If the LAN CONFIGURATION panel is displayed, select the options that are appropriate for your environment, and press **Enter**.

6. If the SNA CONFIGURATION panel is displayed, select the options that are appropriate for your environment, and press **Enter**.

7. If you are prompted to enter store controller IDs, specify the appropriate controller IDs, and press **Enter**.

8. Select the store controller that you want to configure, and press **Enter**.

9. From the list of controller configuration items, select **Background Application**, and press **Enter**.

10. From the BACKGROUND APPLICATION panel, select **1** (Define a Background Application).

11. On the DEFINE BACKGROUND APPLICATION panel, specify the following entries:
    - Initial message: `MQMFT`
    - Program name: `ADX_SPGM:ADXCHAIN.386`
    - Parameter list: `@f:\adxetc\mft75\`*agent_name*`.rsp`

    Note: The *agent_name* is restricted to a maximum of 23 characters, and the parameter list entry is restricted to a maximum of 45 characters. The parameter list path to the response file must be specified exactly in the format that is shown, meaning no change to uppercase characters or forward slashes. Defining a logical name to specify the path to the response file is not permitted.

12. Press **PgDn** to see more options, and specify whether the application should be started or stopped when the system becomes the acting master or stops being the acting master.

13. Press **PgDn** again to see further options, and specify whether the application should be started or stopped when the system becomes the acting file server or stops being the acting file server.

14. Press **Enter** to save the changes.

15. Press **Esc** to return to the CONFIGURATION panel.

16. From the CONFIGURATION panel, select **4** (**Activate Configuration**).

17. From the ACTIVATE CONFIGURATION panel, select **2** (**Controller Configuration**). The controller configuration is activated.

18. Re-IPL the store controller.

Starting a new file transfer

You can start a new file transfer from the WebSphere MQ Explorer or from the command line and you can choose to transfer either a single file or multiple files in a group.

About this task

You can also start a file transfer by putting a file transfer message on the command queue of the source agent. An example command queue name is `SYSTEM.FTE.COMMAND.AGENT01`. You must ensure that the message reaches the command queue of the correct source agent; if the message is received by an agent that does not match the source information in the XML, the message is rejected.

The transfer request XML must conform to the `FileTransfer.xsd` schema and use the <request> element as the root element. See File transfer request message format for information about the structure and content of a transfer request message. How you put the transfer request message on an agent command queue is task-specific. For example, you can use the WebSphere MQ Java API to put a message on the queue programmatically.

To start a new file transfer from the command line, see fteCreateTransfer command.

To start a new file transfer by using the **Create New Managed File Transfer** wizard in WebSphere MQ Explorer, use the following steps:

Procedure

1. In the Navigator view, click **Managed File Transfer**. **Managed File Transfer Central** is displayed in the Content view.

2. All of your coordination queue managers are displayed in the Navigator view. Expand the name of the coordination queue manager that the agent you want to use for the transfer is registered against. If you are currently connected to a coordination queue manager other than the one you want to use for the transfer, right-click that coordination queue manager name in the Navigator view and click **Disconnect**. Then right-click the name of the coordination queue manager you want to use and click **Connect**.

3. Start the **Create New Managed File Transfer** wizard by using either of the following methods:

   a. Right-click the name of any of the following nodes in the Navigator view: the relevant coordination queue manager, **Transfer Templates**, **Transfer Log**, or **Pending Transfers**. Then click **New Transfer** to start the wizard.

   b. Click **File** > **New** > **Other** > **Managed File Transfer Wizards** > **New Transfer Wizard**

4. Follow the instructions on the wizard panels. There is also context-sensitive help provided for each panel. To access the context-sensitive help on Windows, press F1. On Linux, press `Ctrl+F1` or `Shift+F1`.

Using transfer definition files

You can specify a transfer definition file which can be used to create a file transfer. The transfer definition file is an XML file that defines some or all of the information required to create the transfer.

Transfer definition files are useful when you want to specify multiple source files and multiple destination files in a single transfer operation. You can use a transfer definition file to submit a complex file transfer. You can reuse and share the transfer definition file.

You can use two formats for a transfer definition file, and while these formats vary slightly, both conform to the `FileTransfer.xsd` schema. You can find this schema in the `samples\schema` directory of the WebSphere MQ Managed File Transfer installation.

The following two formats of transfer definition files are supported:

* A definition of the source and destination files for a transfer. This definition uses a <transferSpecifications> element as the root.
* A definition of the entire transfer, including source and destination files and the source and destination agents. This definition uses a <request> element as the root.
  – Files with this format can be generated from the **fteCreateTransfer** command by using the **-gt** parameter.

The following example shows a transfer definition file format that specifies only the source and destination files for a transfer:

```
<?xml version="1.0" encoding="UTF-8"?>
<transferSpecifications xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
  <item checksumMethod="MD5" mode="text">
    <source recursive="false" disposition="leave">
      <file>textTransferTest.txt</file>
    </source>
    <destination type="directory" exist="overwrite">
      <file>c:\targetfiles</file>
    </destination>
  </item>
</transferSpecifications>
```

To submit this format of transfer definition file you must specify the source and destination agents on the command line:

```
fteCreateTransfer -sa AGENT1 –sm agent1qm -da AGENT2 –dm agent2qm -td
  c:\definitions\example1.xml
```

The following example is a transfer definition file format that specifies all information required for a transfer:

```
<?xml version="1.0" encoding="UTF-8"?>
<request version="3.00" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
  <managedTransfer>
    <originator>
      <hostName>example.com.</hostName>
      <userID>fteuser</userID>
    </originator>
    <sourceAgent agent="AGENT1" QMgr="agent1qm"/>
    <destinationAgent agent="AGENT2" QMgr="agent2qm"/>
    <transferSet>
      <item mode="binary" checksumMethod="MD5">
        <source recursive="false" disposition="leave">
          <file>c:\sourcefiles\*.jpg</file>
        </source>
        <destination type="directory" exist="error">
          <file>/targetfiles/images</file>
        </destination>
      </item>
    </transferSet>
  </managedTransfer>
</request>
```

You can generate a file with this format by using the **-gt** parameter on the **fteCreateTransfer** command. When you submit a transfer definition file with this format, you do not need to specify anything else on the command line:

```
fteCreateTransfer -td c:\definitions\example2.xml
```

You can override the source and destination agent information about the command line by passing in the normal parameters in addition to the transfer definition file. For example:

```
fteCreateTransfer -da AGENT9 -dm agent9qm -td c:\definitions\example2.xml
```

This example uses the command-line options to override the destination agent defined inside the transfer definition file with **AGENT9** and the destination queue manager defined in the transfer definition file as **agent9qm**.

Both of the formats described can contain one or more <item> elements. For further information about the <item> element, see File transfer request message format. Each of these transfer items defines a source and destination file pair with additional attributes to control the behavior of the transfer. For example, you can specify the following behavior:

- Whether the transfer uses a checksum
- Whether the transfer is text or binary
- Whether to delete the source file after the transfer has completed
- Whether to overwrite the destination file if the file exists

An advantage of using transfer definition files is that you can specify additional options that are not available from the command line. For example, when you are carrying out message-to-file transfers, you can specify the groupId attribute by using a transfer definition file. This attribute specifies the WebSphere MQ group ID of the messages that are read from the queue. Another advantage of transfer definition files is that you can specify different options for each file pair. For example, you can specify whether a checksum is used, or whether the file is transferred in text or binary mode, on a file-by-file basis. If you use the command line, the same options apply for every file in a transfer.

For example:

```
<item checksumMethod="none" mode="binary">
  <source disposition="leave">
    <file>c:\sourcefiles\source1.doc</file>
  </source>
  <destination type="file" exist="error">
    <file>c:\destinationfiles\destination1.doc</file>
  </destination>
</item>

<item checksumMethod="MD5" mode="text">
  <source disposition="delete">
    <file>c:\sourcefiles\source2.txt</file>
  </source>
  <destination type="file" exist="overwrite">
    <file encoding="UTF8" EOL="CRLF">c:\destinationfiles\destination2.txt</file>
  </destination>
</item>

<item checksumMethod="none" mode="text">
  <source recursive="false" disposition="leave">
    <file>c:\originfiles\source3.txt</file>
  </source>
  <destination type="file" exist="overwrite">
    <file>c:\targetfiles\destination3.txt</file>
  </destination>
</item>
```

Creating a scheduled file transfer

You can schedule a new file transfer either from the WebSphere MQ Explorer or from the command line. The scheduled transfer can contain single files or multiple files in a group. You can perform a scheduled file transfer once or repeat the transfer multiple times.

About this task

You can set up a file transfer schedule to occur once, or to occur at the following intervals:
- Every minute
- Hourly
- Daily
- Weekly
- Monthly
- Yearly

You can then specify the occurrences to stop at the following points:
- At a defined time and date
- After a defined number of occurrences

Alternatively, you can specify that the occurrences continue forever.

To create a new scheduled file transfer using the command line, use the scheduling parameters (**-tb**, **-ss**, **-oi**, **-of**, **-oc**, and **-es**) for the fteCreateTransfer command.

To create a new scheduled file transfer using the **Create New Managed File Transfer** wizard in WebSphere MQ Explorer, use the following steps:

Procedure

1. In the Navigator view, click **Managed File Transfer**. **Managed File Transfer Central** is displayed in the Content view.
2. All of your coordination queue managers are displayed in the Navigator view. Expand the name of the coordination queue manager that the agent you want to use for the transfer is registered against. If you are currently connected to a coordination queue manager other than the one you want to use

for the transfer, right-click that coordination queue manager name in the Navigator view and click **Disconnect**. Then right-click the name of the coordination queue manager you want to use and click **Connect**.

3. Start the **Create New Managed File Transfer** wizard using either of the following methods:

   a. Right-click the name of any of the following nodes in the Navigator view: the relevant coordination queue manager, **Transfer Templates**, **Transfer Log**, or **Pending Transfers**. Then click **New Transfer** to start the wizard.

   b. Click **File** > **New** > **Other** > **Managed File Transfer Wizards** > **New Transfer Wizard**

4. Follow the instructions on the wizard panels. Ensure that you select the **Enable schedule transfer** check box and enter your schedule details on the **Schedule** tab. Scheduled file transfers start within a minute of the schedule start time, if there are no problems that might affect the transfer. For example, there might be issues with your network or agent that prevent the scheduled transfer starting. There is context-sensitive help provided for each panel. To access the context-sensitive help on Windows, press F1. On Linux, press `Ctrl+F1` or `Shift+F1`.

Results

For information about the messages involved in scheduled file transfers, see Message formats for scheduled transfers.

Working with pending transfers from the WebSphere MQ Explorer

You can view scheduled file transfers that are pending from the WebSphere MQ Explorer. The **Pending Transfers** window displays all of the pending transfers registered with the coordination queue manager that you are currently connected to.

About this task

To view the status of a scheduled file transfer that has not yet started, use the following steps:

Procedure

1. Expand **Managed File Transfer** in the Navigator view. **Managed File Transfer Central** is displayed in the Content view.

2. All of your coordination queue managers are displayed in the Navigator view. Expand the name of the coordination queue manager that you have used for the scheduled transfer. If you want to change which coordination queue manager you are connected to, right-click the name of the coordination queue manager you want to use in Navigator view and click **Connect**.

3. Click **Pending Transfers**. The Pending Transfers window is displayed in the Content view.

4. The Pending Transfers window displays the following details about your scheduled file transfers:

   a. **Name** The number of the scheduled file transfer. This number is automatically assigned.

   b. **Source** The name of the source agent.

   c. **Source File** The name of the file to be transferred on its host system.

   d. **Destination** The name of the destination agent.

   e. **Destination File** The name of the file after it is transferred to the destination system.

   f. **Scheduled Start (selected time zone)** The time and date that the file transfer is scheduled to start in the administrator's selected time zone. To change the time zone displayed, click **Window** > **Preferences** > **WebSphere MQ Explorer** > **Managed File Transfer** and select an alternative time zone from the **Time zone:** list. Click **OK**.

   g. **Repeat Every** If you have chosen to repeat the scheduled transfer, the specified interval that you want to repeat the transfer, expressed as a number.

h. **Repeat Type** If you have chosen to repeat the scheduled transfer, the type of repeat interval you have specified for the file transfer. The type can be one of the following values: `minutes`, `hours`, `days`, `weeks`, `months`, or `years`.

i. **Repeat Until** If you have chosen to repeat the scheduled transfer, the details of when you want the repeating file transfer to stop. For example, a specified date and time, or after a specified number of occurrences.

Results

To refresh what is displayed in the **Pending Transfers** window, click the Refresh button on the Content view toolbar.

To cancel a pending file transfer, right-click the particular transfer and click **Cancel**. Canceling a transfer completely discards the file transfer request.

Triggering a file transfer

You can set certain trigger conditions on a file transfer that must be true before that transfer can take place. If the triggering conditions are not true, the file transfer does not take place and a log message is optionally submitted to record the fact the transfer did not happen. The file transfer request is then discarded. For example, you can set up a file transfer that takes place only if a named file on the system where the source agent is located is over a specified size, or if a particular named file exists on the system where the source agent is located. You can set up a triggered file transfer from either the WebSphere MQ Explorer or from the command line.

About this task

You can monitor a resource continually for a trigger condition to be satisfied. For further information about resource monitoring see: "Resource monitoring" on page 215.

There are three different triggering conditions that you can set. The conditions are as follows:
- If a particular file exists on the same system as the source agent
- If a particular file does not exist on the same system as the source agent
- If a particular file is over a certain size on the system where the source agent is located (the size can be expressed in bytes, KB, MB, or GB). These units of measurement use the 2^{10} convention, for example 1 KB equals 1024 bytes and 1 MB equals 1024 KB.

The triggering types in the preceding list can be combined in two ways:
- For a single condition, you can specify more than one file on the system where the source agent is located. This triggers the transfer if any one of the specified files meets the condition (Boolean operator OR).
- You can specify multiple conditions. This triggers the transfer only if all of the conditions are met (Boolean operator AND).

You can also combine a triggered transfer with a scheduled transfer. See Creating a scheduled file transfer for more information. In this case the trigger conditions are evaluated at the time the schedule is due to start, or for a repeating schedule every time the schedule is due to start.

Triggered transfers are not supported on protocol bridge agents.

To create a triggered file transfer by using the command line, use the **-tr** parameter on the fteCreateTransfer command.

To create a scheduled file transfer by using the **Create New Managed File Transfer** wizard in WebSphere MQ Explorer, use the following steps:

Procedure

1. In the Navigator view, click **Managed File Transfer**. **Managed File Transfer Central** is displayed in the Content view.
2. All of your coordination queue managers are displayed in the Navigator view. Expand the name of the coordination queue manager that you have used for the scheduled transfer. If you want to change which coordination queue manager you are connected to, right-click the name of the coordination queue manager you want to use in Navigator view and click **Connect**.
3. Start the **Create New Managed File Transfer** wizard by using either of the following methods:
   a. Right-click the name of any of the following nodes in the Navigator view: the relevant coordination queue manager, **Transfer Templates**, **Transfer Log**, or **Pending Transfers**. Then click **New Transfer** to open the wizard.
   b. Click **File** > **New** > **Other** > **Managed File Transfer Wizards** > **New Transfer Wizard**
4. Follow the instructions on the wizard panels. Ensure that you select the **Enable triggered transfer** check box on the **Triggers** tab and complete the fields on that tab to set up triggering. There is context-sensitive help provided for each panel. To access the context-sensitive help on Windows, press F1. On Linux, press **Ctrl+F1** or Shift+F1.

Monitoring file transfers that are in progress from WebSphere MQ Explorer

You can monitor a file transfer that is in progress using the **Managed File Transfer - Current Transfer Progress** tab in WebSphere MQ Explorer. This file transfer can be one started from either WebSphere MQ Explorer or the command line. The tab also displays the progress of scheduled transfers at the point the scheduled transfers start.

About this task

If you want to use WebSphere MQ Explorer to monitor transfers associated with a coordination queue manager on a remote system, follow the instructions in the Configuring WebSphere MQ Explorer to monitor a remote coordination queue manager topic.

Previous file transfer information is not retained after you stop and restart WebSphere MQ Explorer. At restart, the information about past transfers is cleared from the **Current Transfer Progress** tab. You can clear completed transfers using **Remove completed transfers** at any point when WebSphere MQ Explorer is open.

Procedure

After you have started a new file transfer using WebSphere MQ Explorer or the command line, you can monitor the progress of your transfer in the **Current Transfer Progress** tab. The following information is displayed for each transfer in progress:

1. **Source**. The name of the agent used to transfer the file from the source system.
2. **Destination**. The name of the agent used to receive the file at the destination system.
3. **Current file**. The name of the file currently being transferred. The part of the individual file that has already been transferred is displayed in B, KiB, MiB. GiB, or TiB along with total size of the file in parentheses. The unit of measurement displayed depends on the size of the file. B is bytes per second. KiB/s is kibibytes per second, where 1 kibibyte equals 1024 bytes. MiB/s is mebibytes per second, where 1 mebibyte equals 1 048 576 bytes. GiB/s is gibibytes per second where 1 gibibyte equals 1 073 741 824 bytes. TiB/s is tebibytes per second where 1 tebibyte equals 1 099 511 627 776 bytes.

4. **File number**. If you are transferring more than one file, this number represents how far through the total group of files the transfer is.

5. **Progress**. The progress bar shows how complete the current file transfer is as a percentage.

6. **Rate**. The rate the file is being transferred in KiB/s (kibibytes per second, where 1 kibibyte equals 1024 bytes.)

7. **Started (selected time zone)**. The time that the file transfer started, presented in the selected time zone of the administrator. To change the time zone displayed, click **Window** > **Preferences** > **WebSphere MQ Explorer** > **Managed File Transfer** and select an alternative time zone from the **Time zone:** list. Click **OK**. If the transfer enters a recovery state while transferring the file, the started time updates to reflect the time that the file transfer resumed.

Results

This tab regularly refreshes its information automatically, but to force a refreshed view of what is displayed in the **Current Transfer Progress** tab, click **Refresh** on the Content view toolbar.

To delete file transfers from the **Current Transfer Progress** tab, click **Remove completed transfers** on the Content view toolbar. Clicking this button removes file transfer details from the tab only; it does not stop or cancel a current or scheduled transfer.

If you want to return to the **Current Transfer Progress** tab after closing it, you can display the tab by clicking **Window** > **Show View** > **Other** > **Other** > **Managed File Transfer - Current Transfer Progress**. Click **OK**.

Related tasks:

"Configuring WebSphere MQ Explorer to monitor a remote coordination queue manager"
Use WebSphere MQ Explorer to monitor file transfers associated with a coordination queue manager running on a remote system. In WebSphere MQ V7.5, you require a system that is capable of running the WebSphere MQ Explorer. The WebSphere MQ Explorer component needs to be installed to be able to connect to the remote coordination queue manager.

"Viewing the status of file transfers by using the Transfer Log" on page 213
You can view the details of file transfers by using the **Transfer Log** in WebSphere MQ Explorer. These can be transfers started from either the command line or the WebSphere MQ Explorer. You can also customize what is displayed in the **Transfer Log**.

Configuring WebSphere MQ Explorer to monitor a remote coordination queue manager

Use WebSphere MQ Explorer to monitor file transfers associated with a coordination queue manager running on a remote system. In WebSphere MQ V7.5, you require a system that is capable of running the WebSphere MQ Explorer. The WebSphere MQ Explorer component needs to be installed to be able to connect to the remote coordination queue manager.

About this task

Assumptions: Authority to connect to the remote coordination queue manager by configuring the queue manager to allow for remote connections.

For more information on how to configure this, see "Connecting to a IBM WebSphere MQ V7.1 or later queue manager in client mode with channel authentication" on page 110 and "Authorities for resources specific to WebSphere MQ Managed File Transfer" on page 404.

To monitor queue managers and file transfers between agents on a system that is not running Windows or Linux, configure the WebSphere MQ Explorer to connect to the remote system using the following steps:

Procedure

1. Start the local WebSphere MQ Explorer.
2. When WebSphere MQ Explorer is loaded, right-click on the **Managed File Transfer** folder and select **New configuration**.
3. Proceed through the wizard, selecting the Coordination and Commands queue manager, then define a name for the configuration.
4. Click **finish** to complete the definition.
5. When the definition is finished, right-click on the definition and select **Connect**.

Results

Now start WebSphere MQ Explorer and use it to monitor transfer activity for the WebSphere MQ Managed File Transfer network associated with the coordination queue manager.

Related tasks:

"Monitoring file transfers that are in progress from WebSphere MQ Explorer" on page 211
You can monitor a file transfer that is in progress using the **Managed File Transfer - Current Transfer Progress** tab in WebSphere MQ Explorer. This file transfer can be one started from either WebSphere MQ Explorer or the command line. The tab also displays the progress of scheduled transfers at the point the scheduled transfers start.

"Viewing the status of file transfers by using the Transfer Log"
You can view the details of file transfers by using the **Transfer Log** in WebSphere MQ Explorer. These can be transfers started from either the command line or the WebSphere MQ Explorer. You can also customize what is displayed in the **Transfer Log**.

Viewing the status of file transfers by using the Transfer Log

You can view the details of file transfers by using the **Transfer Log** in WebSphere MQ Explorer. These can be transfers started from either the command line or the WebSphere MQ Explorer. You can also customize what is displayed in the **Transfer Log**.

Procedure

1. Expand **Managed File Transfer** in the Navigator view and then expand the name of the coordination queue manager that you want to view the transfer log for.
2. Click **Transfer Log** in the Navigator view. The Transfer Log is displayed in the Content view.
3. The Transfer Log window displays the following details about your file transfers:
   a. **Source** The name of the agent on the system where the source file is located.
   b. **Destination** The name of the agent on the system you want to transfer the file to.
   c. **Completion State** The status of the file transfer. The state can be one of the following values: "Started", "In progress", "Successful", "Partially Successful", "Cancelled", or "Failed".
   d. **Owner** The user ID on the host that submitted the transfer request.
   e. **Started (selected time zone)** The time and date that the file transfer request was accepted by the WebSphere MQ Managed File Transfer agent, presented in the selected time zone of the administrator. To change the time zone displayed, click **Window** > **Preferences** > **WebSphere MQ Explorer** > **Managed File Transfer** and select an alternative time zone from the **Time zone:** list. Click **OK**.

f. **State Recorded (selected time zone)** (This column is not displayed by default. You can choose to display the column by using the **Configure Transfer Log Columns** ▦ window.) The time and date that the completion state was recorded, in the time zone selected by the administrator.

g. **Job Name** An identifier specified by the user by using the **-jn** parameter of `fteCreateTransfer` or in an Ant script

h. **Transfer ID** The unique identifier for the file transfer.

i. **Connect: Direct** Details about **Process Number**, **Process Name**, **Primary Node**, **Secondary Node**, **Source Type** and **Destination Type** are listed.

Results

To view further details about a completed transfer, expand the transfer that you are interested in using the plus sign (+). You can then see all of the source and destination file names included in that transfer. However, if the transfer is currently in progress and consists of many files, you can view only the files that have already been transferred so far.

To refresh what is displayed in the **Transfer Log**, click the **Refresh** button 🔄 on the Content view toolbar. The file transfer information in the Transfer Log remains in the log after you stop and restart the WebSphere MQ Explorer. If you want to delete all completed file transfers from the log, click **Remove Completed Transfers** 🧹 on the Content view toolbar.

To delete an individual completed file transfer from the log, right-click the transfer and click **Delete**. If you delete a transfer, it does not stop or cancel a transfer that is in progress or that has been scheduled; you are deleting only the stored historical data.

To copy the unique identifier of a transfer to the clipboard, right-click that transfer and click **Copy ID**.

The metadata and the complete audit XML for the transfer are available from the context menu, under the **Properties** action.

Related tasks:

"Monitoring file transfers that are in progress from WebSphere MQ Explorer" on page 211
You can monitor a file transfer that is in progress using the **Managed File Transfer - Current Transfer Progress** tab in WebSphere MQ Explorer. This file transfer can be one started from either WebSphere MQ Explorer or the command line. The tab also displays the progress of scheduled transfers at the point the scheduled transfers start.

"Configuring the Transfer Log"
You can configure what information is displayed and how information is displayed in the **Transfer Log** in the WebSphere MQ Explorer.

"Monitoring file transfers that are in progress from WebSphere MQ Explorer" on page 211
You can monitor a file transfer that is in progress using the **Managed File Transfer - Current Transfer Progress** tab in WebSphere MQ Explorer. This file transfer can be one started from either WebSphere MQ Explorer or the command line. The tab also displays the progress of scheduled transfers at the point the scheduled transfers start.

Configuring the Transfer Log

You can configure what information is displayed and how information is displayed in the **Transfer Log** in the WebSphere MQ Explorer.

About this task

To rearrange the order of the columns in the **Transfer Log**, click the title of the column you want to move and drag the column to its new position. The new column order is retained only until you next stop and restart the WebSphere MQ Explorer.

To filter entries in the **Transfer Log**, enter a string in the **Filter the displayed log entries** field. To restore all of the entries to the log, delete the string you entered from the field. You can use any valid Java regular expression in this field. For more information, see "Regular expressions used by WebSphere MQ Managed File Transfer" on page 685.

To customize which columns are displayed in the Transfer Log, use **Configure Transfer Log Columns** . Use the following steps to start and use the **Configure Transfer Log Columns** window.

Procedure

1. Ensure that you have the **Transfer Log** open in the Content view. Click **Configure Transfer Log Columns** on the Content view toolbar. The **Configure Transfer Log Columns** window opens.
2. To customize your view of the **Transfer Log**, select or clear individual check boxes for the columns you want to show or hide. You can click **Select All**, then **OK** to select all of the check boxes or **Deselect All**, then **OK** to clear all of the check boxes.

Related tasks:
"Monitoring file transfers that are in progress from WebSphere MQ Explorer" on page 211
You can monitor a file transfer that is in progress using the **Managed File Transfer - Current Transfer Progress** tab in WebSphere MQ Explorer. This file transfer can be one started from either WebSphere MQ Explorer or the command line. The tab also displays the progress of scheduled transfers at the point the scheduled transfers start.
"Viewing the status of file transfers by using the Transfer Log" on page 213
You can view the details of file transfers by using the **Transfer Log** in WebSphere MQ Explorer. These can be transfers started from either the command line or the WebSphere MQ Explorer. You can also customize what is displayed in the **Transfer Log**.

Resource monitoring

You can monitor WebSphere MQ Managed File Transfer resources; for example, a queue or a directory. When a condition on this resource is satisfied, the resource monitor starts a task, such as a file transfer. You can create a resource monitor by using the `fteCreateMonitor` command or the Monitors view in the WebSphere MQ Managed File Transfer plug-in for WebSphere MQ Explorer.

A common scenario is to monitor a directory for the presence of a trigger file. An external application might be processing multiple files and placing them in a known source directory. When the application has completed its processing, it indicates that the files are ready to be transferred, or otherwise acted upon, by placing a trigger file into a monitored location. The trigger file can be detected by a WebSphere MQ Managed File Transfer monitor and the transfer of those files from the source directory to another WebSphere MQ Managed File Transfer agent is initiated.

Two examples of monitoring a directory are as follows:
* Monitor for a trigger file (for example `trigger.file`) and then transfer a wildcard (for example, `*.zip`)
* Monitor for `*.zip` and then transfer `${FilePath}` (for example, the file that triggered the transfer). For more details about variable substitution, see "Customizing tasks with variable substitution" on page 225.

Do not create a monitor that monitors for `*.zip`, and then transfers `*.zip`. The monitor tries to start a transfer of `*.zip` for every .zip file on your system. That is, the monitor generates * number of transfers for `*.zip`.

To see an example of creating a resource monitor to monitor a directory, see "Monitoring a directory and using variable substitution" on page 223.

An example of monitoring a queue is as follows:

- An external application might be generating messages and placing them on a known queue with the same group ID. When the application has completed putting messages on the queue, it indicates that the group is complete. The complete group of messages can be detected by a WebSphere MQ Managed File Transfer monitor and the transfer of the group of messages from the source queue to a file is initiated.

To see an example of creating a resource monitor to monitor a queue, see "Example: Configuring a resource monitor to monitor a queue" on page 225.

WebSphere MQ Managed File Transfer resource monitoring uses the following terminology:

monitor
> A process that polls a resource (such as a directory or queue) at a predefined regular interval to see if the resource contents have changed. If they have, the contents are compared with the set of conditions for this monitor. If there is a match, the task for this monitor is started.

resource
> The system resource the monitor examines every poll interval to be compared with the trigger conditions. Queues, directories, or nested directory structures can be the monitored resource.

condition
> An expression that is evaluated (typically against the content of the monitored resource). If the expression evaluates to true, the condition contributes to the overall trigger condition.

trigger condition
> The overall condition, which is satisfied when all conditions are satisfied. When the trigger condition is satisfied the task can proceed.

task   The operation that is started when the trigger condition or set of conditions is satisfied. Supported tasks are file transfer and command call.

trigger file

> A file that is placed in a monitored directory to indicate that a task (typically a transfer) can begin. For example, it might indicate that all the files to be processed have arrived in a known location and can be transferred or otherwise acted upon. The name of the trigger file can be used to specify the files to be transferred by using variable substitution. For more information, see "Customizing tasks with variable substitution" on page 225.

> The trigger file is also known as ready file or go file. However, in this documentation it is always referred to as the trigger file.

Resource monitoring is not supported on protocol bridge agents, Connect:Direct bridge agents, or Web Gateway agents.

Related reference:

"Monitor request message formats" on page 820
Resource monitors are created when a suitable XML message arrives at an agent's command queue, typically as a result of a user issuing the fteCreateMonitor command or using the WebSphere MQ Explorer interface.

Resource monitoring concepts

An overview of the key concepts of the WebSphere MQ Managed File Transfer resource monitoring feature.

Monitors

The resource monitor is associated with a WebSphere MQ Managed File Transfer agent, and is only active when that agent is started. When the monitoring agent stops, so does the monitor. If the agent is already started when the monitor is created, the monitor starts immediately. The monitoring agent must also be the source agent of the task that is initiated by the monitor.

Monitor names must be unique within their agent. The monitor name must be a minimum of one character in length and must not contain asterisk (*), percent (%) or question mark (?) characters. The case of supplied monitor names is ignored and the monitor name is converted to uppercase. If you try to create a monitor with a name that is already present, the request is ignored and the attempt is logged to the monitor log topic.

There is no restriction on the number of monitors that can be created on an agent, and all run with the same priority. Consider the implications of overlapping monitored resources, conflicting trigger conditions and how frequently the resources are polled.

Monitors look at the contents of resources after every poll interval period. The contents of the resource are compared with the trigger conditions and if those conditions are satisfied, the task associated with the monitor is called.

The task is started asynchronously. If there is a condition match, and the task is started, the monitor continues to poll for further changes to the resource contents. So for example, if a match occurred because a file called reports.go arrived in a monitored directory, the task would be started once. At the next poll interval, even if the file still exists, the task is not started again. However, if the file is deleted and then placed in the directory again, or the file is updated (such that the last modified date attribute is changed), the next trigger condition check causes the task to be called again.

Resources

Monitors in WebSphere MQ Managed File Transfer can poll the contents of directories or nested directory structures. By default, the specified directory is monitored. To also examine subdirectories set the recursion level in the **fteCreateTransfer** command.

Monitors in WebSphere MQ Managed File Transfer can poll the contents of WebSphere MQ queues. You can specify only one monitor per queue. If you specify more than one monitor to poll a WebSphere MQ queue, unpredictable behavior occurs.

Monitoring data sets is not supported.

Trigger conditions

The condition is met when the resource contains a value that matches some other string or pattern. Conditions can be one of the following:
* Match on file name (pattern)
* No match on file name (pattern)
* File size
* Match if file size remains the same for a number of polls

File name matching can be expressed as:
* Exact string match
* Simple wildcard match as described in "Using wildcard characters" on page 683
* Regular expression match

File names can also be excluded from file name matching by using a wildcard or Java regular expression that identifies file names that are never matched.

When a matching file is detected, its last modified time stamp is retained. If subsequent polls detect that the file has been changed, the trigger condition is satisfied again, and the task is started. If the condition is to detect when a file does not exist, if no file in the monitored directory matches the file name pattern, the task is started. If a file is then added to the directory that does match the file name pattern, the task is only started if the file is then deleted.

Tasks

WebSphere MQ Managed File Transfer supports the following two types of task that you can configure to be started by resource monitors:

- File transfer
- Command

File transfer tasks are defined in the same way as any other file transfer. A useful way to generate the task XML required by a monitor is to run the fteCreateTransfer command with the **-gt** parameter. This command generates a task definition as an XML document, including the transfer specification. You then pass the name of the task XML document as the value for the **-mt** parameter on the fteCreateMonitor command. When the **fteCreateMonitor** is run, it reads the task XML document. After the **fteCreateMonitor** is run, any changes that are made to the task XML file are not used by the monitor.

Command tasks can run Ant scripts, call executable programs, or run JCL jobs. For more information, see Configuring monitor tasks to invoke commands and scripts.

When using a file transfer task, you can select how many trigger conditions are batched into a task. The default is for one trigger condition to start one task. You can run the fteCreateMonitor command with the **-bs** option to select the number of trigger conditions that are batched together into one task.

Related concepts:

"Resource monitoring" on page 215
You can monitor WebSphere MQ Managed File Transfer resources; for example, a queue or a directory. When a condition on this resource is satisfied, the resource monitor starts a task, such as a file transfer. You can create a resource monitor by using the **fteCreateMonitor** command or the Monitors view in the WebSphere MQ Managed File Transfer plug-in for WebSphere MQ Explorer.

Related tasks:

"Configuring monitor tasks to start commands and scripts" on page 219
Resource monitors are not limited to performing file transfers as their associated task. You can also configure the monitor to call other commands from the monitoring agent, including executable programs, Ant scripts or JCL jobs. To call commands, edit the monitor task definition XML to include one or more command elements with corresponding command call parameters, such as arguments and properties.

"Example: Configuring a resource monitor to monitor a queue" on page 225
You can specify a WebSphere MQ queue as the resource to be monitored by a resource monitor by using the **-mq** parameter with the **fteCreateMonitor** command.

"Monitoring a queue and using variable substitution" on page 228
You can monitor a queue and transfer messages from the monitored queue to a file by using the **fteCreateMonitor** command. The value of any WebSphere MQ message property in the first message to be read from the monitored queue can be substituted in the task XML definition and used to define the transfer behavior.

Related reference:

"**fteCreateMonitor** (create new resource monitor)" on page 444
The **fteCreateMonitor** command creates and starts a new resource monitor from the command line. You can monitor a resource (for example, the contents of a directory) using WebSphere MQ Managed File

Transfer so that when a trigger condition is satisfied, a specified task, such as a file transfer, is started.

"fteListMonitors (list WebSphere MQ Managed File Transfer resource monitors)" on page 497
Use the **fteListMonitors** command to list all of the existing resource monitors in a WebSphere MQ Managed File Transfer network using the command line.

"fteDeleteMonitor (delete a WebSphere MQ Managed File Transfer resource monitor)" on page 489
Use the **fteDeleteMonitor** command to stop and delete an existing WebSphere MQ Managed File Transfer resource monitor using the command line. Issue this command against the resource monitoring agent.

Related information:

"Customizing tasks with variable substitution" on page 225
When the trigger conditions of an active resource monitor are satisfied, the defined task is called. In addition to calling the transfer or command task with the same destination agent or the same destination file name every time, you can also modify the task definition at run time. You do this by inserting variable names into the task definition XML. When the monitor determines the trigger conditions are satisfied, and that the task definition contains variable names, it substitutes the variable names with the variable values, and then calls the task.

Configuring monitor tasks to start commands and scripts

Resource monitors are not limited to performing file transfers as their associated task. You can also configure the monitor to call other commands from the monitoring agent, including executable programs, Ant scripts or JCL jobs. To call commands, edit the monitor task definition XML to include one or more command elements with corresponding command call parameters, such as arguments and properties.

About this task

The file path to the executable program, Ant script, or JCL job that you want the monitoring agent to call must be included in the `commandPath` of the monitoring agent. For information about the command path property, see "The commandPath property" on page 414.

You can create the task definition XML document in one of the following ways:

- Create the task definition XML document manually according to the `FileTransfer.xsd` schema. For more information, see "Create the task definition XML manually according to the schema."
- Edit the XML document generated by the **fteCreateTransfer -gt** parameter as the basis for your task definition. For more information, see "Creating a task definition document by modifying a generated document" on page 222.

Whether you want a transfer task or a command task, the task definition must start with a <request> root element. The child element of <request> must be either <managedTransfer> or <managedCall>. You would typically choose <managedCall> when there is a single command or script to run, and <managedTransfer> if you want the task to include a file transfer and optionally up to four command calls.

Create the task definition XML manually according to the schema
About this task

You can manually create a task definition XML file according to the schema `FileTransfer.xsd`. This schema can be found in the *MQ_INSTALLATION_PATH*/mqft/samples/schema. For more information about this schema, see "File transfer request message format" on page 803.

Example

The following example shows an example task definition XML document saved as `cleanuptask.xml`, which uses the <managedCall> element to call an Ant script called `RunCleanup.xml`. The `RunCleanup.xml` Ant script must be located on the commandPath of the monitoring agent.

```
<?xml version="1.0" encoding="UTF-8"?>
<request version="4.00" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
  <managedCall>
    <originator>
      <hostName>hostName</hostName>
      <userID>userID</userID>
      <mqmdUserID>mqmdUserID</mqmdUserID>
    </originator>
    <agent QMgr="QM1" agent="AGENT1"/>
    <reply QMGR="QM1">reply</reply>
    <transferSet priority="1">
      <metaDataSet>
        <metaData key="name1">value1</metaData>
      </metaDataSet>
      <call>
       <command name="RunCleanup.xml" type="antscript" retryCount="2"
         retryWait="30" successRC="0">
       <target>check_exists</target>
       <target>copy_to_archive</target>
       <target>rename_temps</target>
         <target>delete_files</target>
       <property name="trigger.filename" value="${FileName}"/>
       <property name="trigger.path" value="${FilePath}"/>
       </command>
      </call>
    </transferSet>
    <job>
      <name>JOBCLEAN1</name>
    </job>
  </managedCall>
</request>
```

The `<agent>` element specifies the WebSphere MQ Managed File Transfer agent that is configured with the named Ant script on its commandPath.

The `<call><command>...` structure defines the executable or script you want to run. The command takes an optional `type` attribute that can have one of the following values:

antscript
> Run an Ant script in a separate JVM.

executable
> Invoke an executable program.

jcl   Invoke a JCL job.

If you omit the `type` attribute, the default value executable is used.

The `name` attribute specifies the name of the Ant script, executable, or JCL job you want to run, without any path information. The agent searches for the script or program in the locations specified by the commandPath property in the agent's `agent.properties` file.

The `retrycount` attribute specifies the number of times to try calling the program again if the program does not return a success return code. The value assigned to this attribute must not be negative. If you do not specify the `retrycount` attribute, a default value of zero is used.

The `retrywait` attribute specifies the time to wait, in seconds, before trying the program invocation again. The value assigned to this attribute must not be negative. If you do not specify the `retrywait` attribute, a default value of zero is used.

The `successrc` attribute is an expression used to determine when the program invocation successfully runs. The process return code for the command is evaluated using this expression. The value can be composed of one or more expressions combined with a vertical bar (|) character to signify Boolean OR, or an ampersand (&) character to signify Boolean AND. Each expression can be one of the following types of expression:

- A number to indicate an equality test between the process return code and the number.
- A number prefixed with a greater than character (>) to indicate a greater-than test between the number and the process return code.
- A number prefixed with a less than character (<) to indicate a less-than test between the number and the process return code.
- A number prefixed with an exclamation point character (!) to indicate a not-equal-to test between the number and the process return code. For example: >2&<7&!5|0|14 is interpreted as the following return codes being successful: 0, 3, 4, 6, 14. All other return codes are interpreted as being unsuccessful.

If you do not specify the `successrc` attribute, a default value of zero is used. This means that the command is judged to have successfully run if, and only if, it returns a code of zero.

For an Ant script, you would typically specify `<target>` and `<property>` elements. The `<target>` element values must match the target names in the Ant script.

For executable programs, you can specify `<argument>` elements. Nested argument elements specify arguments to pass to the program that is being called as part of the program invocation. The program arguments are built from the values specified by the argument elements in the order that the argument elements are encountered. You can specify zero or more argument elements as nested elements of a program invocation.

The administrator defines and starts the monitor as normal using the task definition XML document that includes the `<managedCall>` element. For example:

```
fteCreateMonitor -ma AGENT1 -mm QM1 -md /monitored -mn MONITOR01 -mt
   /tasks/cleanuptask.xml -pi 30 -pu seconds -tr match,*.go
```

The path to the transfer definition XML document must be on the local file system that you run the **fteCreateMonitor** command from (in this example `/tasks/cleanuptask.xml`). The `cleanuptask.xml` document is used to create the resource monitor only. Any tasks that the `cleanuptask.xml` document references (Ant scripts or JCL jobs) must be in the command path of the monitoring agent. When the monitor trigger condition is satisfied, any variables in the task definition XML are substituted with actual values from the monitor. So for example ${FilePath} is replaced in the request message sent to the agent with `/monitored/cleanup.go`. The request message is put on the agent command queue. The command processor detects that the request is for a program call and starts the specified program. If a command of type `antscript` is called, a new JVM is started and the Ant task runs under the new JVM. For more information about using variable substitution, see Customizing tasks with variable substitution.

Related reference:

"File transfer request message format" on page 803
File transfers are initiated by XML messages arriving at an agent command queue, typically as a result of a user issuing a file transfer command or by using the WebSphere MQ Explorer plug-in. The transfer request XML must conform to the `FileTransfer.xsd` schema and have the `<request>` element as the root element. The `FileTransfer.xsd` schema document is located in the *MQ_INSTALLATION_PATH*/mqft/samples/schema directory. The `FileTransfer.xsd` schema imports `fteutils.xsd`, which is in the same directory.

"The commandPath property" on page 414
Use the commandPath property to restrict the locations that WebSphere MQ Managed File Transfer can run commands from.

Related information:

"Customizing tasks with variable substitution" on page 225
When the trigger conditions of an active resource monitor are satisfied, the defined task is called. In addition to calling the transfer or command task with the same destination agent or the same destination file name every time, you can also modify the task definition at run time. You do this by inserting variable names into the task definition XML. When the monitor determines the trigger conditions are satisfied, and that the task definition contains variable names, it substitutes the variable names with the variable values, and then calls the task.

Creating a task definition document by modifying a generated document
About this task

You can create the monitor task definition document by modifying the XML document generated by the
-gt option of **fteCreateTransfer**. The generated document has a <request> followed by
<managedTransfer> element. To convert this task definition to a valid <managedCall> structure, follow
these steps:

Procedure

1. Replace the <managedTransfer> start and end tags with <managedCall> tags.

2. Remove any <schedule> element and child nodes.

3. Replace the <sourceAgent> start and end tags with <agent> to match the monitoring agent
   configuration details.

4. Remove <destinationAgent> and <trigger> elements.

5. Remove <item> elements.

6. Insert a new <call>...</call> structure within the <transferSet> element. This structure contains
   the command definition as shown in the following example:

```
<call>
     <command name="RunCleanup.xml" type="antscript" retryCount="2"
       retryWait="30" successRC="0">
     <target>check_exists</target>
     <target>copy_to_archive</target>
     <target>rename_temps</target>
       <target>delete_files</target>
     <property name="trigger.filename" value="${FileName}"/>
     <property name="trigger.path" value="${FilePath}"/>
     </command>
     </call>
```

Example

You can also retain the <managedTransfer> element including all the file transfer details, and insert up to
four command calls. In this case you insert any selection of the following call elements between the
<metaDataSet> and <item> elements:

preSourceCall
> Call a program on the source agent before starting the transfer.

postSourceCall
> Call a program on the source agent after completing the transfer.

preDestinationCall
> Call a program on the destination agent before starting the transfer.

postDestinationCall
> Call a program on the destination agent after completing the transfer.

Each of these elements takes the <command> element structure as described in the earlier example. The
FileTransfer.xsd schema defines the types used by the various call elements.

The following example shows preSourceCall, postSourceCall, preDestinationCall, and postDestinationCall
in a task definition document:

```
...
...
    <transferSet priority="1">
      <metaDataSet>
        <metaData key="key1">value1</metaData>
      </metaDataSet>
      <preSourceCall>
        <command name="send.exe" retryCount="0" retryWait="0" successRC="0"
          type="executable">
          <argument>report1.pdf</argument>
```

```
        <argument>true</argument>
      </command>
    </preSourceCall>
    <postSourceCall>
      <command name="//DO_IT.JCL" retryCount="0" retryWait="0" successRC="0"
        type="jcl">
        <argument>argument</argument>
      </command>
    </postSourceCall>
    <preDestinationCall>
      <command name="ant_script.xml" retryCount="0" retryWait="0" successRC="0"
        type="antscript">
        <target>step1</target>
        <property name="name" value="value"/>
      </command>
    </preDestinationCall>
    <postDestinationCall>
      <command name="runit.cmd" retryCount="0" retryWait="0" successRC="0" />
    </postDestinationCall>
    <item checksumMethod="none" mode="binary">
...
...
```

You can mix different types of command into the transfer. Argument, target, and property elements are optional.

Monitoring a directory and using variable substitution

You can monitor a directory using the **fteCreateMonitor** command. The value of a substitution variable can be substituted in the task XML definition and used to define the transfer behavior.

About this task

In this example, the source agent is called AGENT_HOP. The directory that AGENT_HOP monitors is called /test/monitored. The agent polls the directory every 5 minutes.

After a .zip file is written to the directory, the application that writes the file to the directory writes a trigger file to the same directory. The name of the trigger file is the same as the name of the .zip file, but has a different file extension. For example, after the file file1.zip is written to the directory, the file file1.go is written to the directory. The resource monitor monitors the directory for files that match the pattern *.go then uses variable substitution to request a transfer of the associated .zip file.

Procedure

1. Create the task XML that defines the task that the monitor performs when it is triggered.

```
<?xml version="1.0" encoding="UTF-8" ?>
<request version="4.00"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
  <managedTransfer>
    <originator>
      <hostName>blue.example.com</hostName>
      <userID>USER1</userID>
    </originator>
    <sourceAgent agent="AGENT_HOP" QMgr="QM_HOP" />
    <destinationAgent agent="AGENT_SKIP" QMgr="QM_SKIP" />
    <transferSet>
      <item mode="binary" checksumMethod="none">
        <source>
          <file>/test/monitored/${fileName{token=1}{separator=.}}.zip</file>
        </source>
        <destination type="file" exist="overwrite">
          <file>/out/${fileName{token=1}{separator=.}}.zip</file>
        </destination>
```

```
        </item>
      </transferSet>
    </managedTransfer>
  </request>
```

The variables that are replaced with the values associated with the trigger file are highlighted in **bold**. This task XML is saved to the file `/home/USER1/task.xml`

2. Create a resource monitor to monitor the directory `/test/monitored`. Submit the following command:

```
fteCreateMonitor -ma AGENT_HOP -mm QM_HOP -md /test/monitored
                 -mn myMonitor -mt /home/USER1/task.xml
                 -tr match,*.go -pi 5 -pu minutes
```

3. A user or program writes the file `jump.zip` to the directory `/test/monitored`, then writes the file `jump.go` to the directory.

4. The monitor is triggered by the existence of the file `jump.go`. The agent substitutes the information about the trigger file into the task XML. This results in the task XML being transformed to:

```
<?xml version="1.0" encoding="UTF-8" ?>
<request version="4.00"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
  <managedTransfer>
    <originator>
      <hostName>blue.example.com</hostName>
      <userID>USER1</userID>
    </originator>
    <sourceAgent agent="AGENT_HOP" QMgr="QM_HOP" />
    <destinationAgent agent="AGENT_SKIP" QMgr="QM_SKIP" />
    <transferSet>
      <item mode="binary" checksumMethod="none">
        <source>
          <file>/test/monitored/jump.zip</file>
        </source>
        <destination type="file" exist="overwrite">
          <file>/out/jump.zip</file>
        </destination>
      </item>
    </transferSet>
  </managedTransfer>
</request>
```

Results

The transfer defined by the task XML is performed. The `jump.zip` file is read from the `/test/monitored` directory by AGENT_HOP and is transferred to a file called `/out/jump.zip` located on the system where AGENT_SKIP is running.

Related concepts:

"Resource monitoring" on page 215
You can monitor WebSphere MQ Managed File Transfer resources; for example, a queue or a directory. When a condition on this resource is satisfied, the resource monitor starts a task, such as a file transfer. You can create a resource monitor by using the **fteCreateMonitor** command or the Monitors view in the WebSphere MQ Managed File Transfer plug-in for WebSphere MQ Explorer.

Related tasks:

"Configuring monitor tasks to start commands and scripts" on page 219
Resource monitors are not limited to performing file transfers as their associated task. You can also configure the monitor to call other commands from the monitoring agent, including executable programs, Ant scripts or JCL jobs. To call commands, edit the monitor task definition XML to include one or more command elements with corresponding command call parameters, such as arguments and properties.

Related reference:

"**fteCreateMonitor** (create new resource monitor)" on page 444
The **fteCreateMonitor** command creates and starts a new resource monitor from the command line. You can monitor a resource (for example, the contents of a directory) using WebSphere MQ Managed File Transfer so that when a trigger condition is satisfied, a specified task, such as a file transfer, is started.

Related information:

"Customizing tasks with variable substitution"
When the trigger conditions of an active resource monitor are satisfied, the defined task is called. In addition to calling the transfer or command task with the same destination agent or the same destination file name every time, you can also modify the task definition at run time. You do this by inserting variable names into the task definition XML. When the monitor determines the trigger conditions are satisfied, and that the task definition contains variable names, it substitutes the variable names with the variable values, and then calls the task.

Example: Configuring a resource monitor to monitor a queue

You can specify a WebSphere MQ queue as the resource to be monitored by a resource monitor by using the **-mq** parameter with the **fteCreateMonitor** command.

About this task

In this example, the resource to be monitored is the queue *MONITORED_QUEUE*. This queue must be on the monitoring agent's queue manager, *QM_NEPTUNE*. The condition that the queue is monitored for is the presence of a complete group of messages. The task to be performed if the condition is satisfied is defined in the file `task.xml`.

Procedure

Type the following command:

```
fteCreateMonitor -ma AGENT_NEPTUNE -mn myMonitor -mm QM_NEPTUNE -mq MONITORED_QUEUE
  -mt task.xml -tr completeGroups -pi 5 -pu minutes
```

The monitor checks the queue every five minutes to see if the condition `completeGroups` is true. If there are one or more complete groups on the queue, the monitor runs the task defined in the `task.xml` file once for each complete group.

What to do next

Do not create more than one resource monitor to monitor an individual queue. If you do then unpredictable behavior occurs.

Customizing tasks with variable substitution

When the trigger conditions of an active resource monitor are satisfied, the defined task is called. In addition to calling the transfer or command task with the same destination agent or the same destination file name every time, you can also modify the task definition at run time. You do this by inserting variable names into the task definition XML. When the monitor determines the trigger conditions are satisfied, and that the task definition contains variable names, it substitutes the variable names with the variable values, and then calls the task.

If the monitored resource is a queue

The value of any WebSphere MQ message property in the first message to be read from the monitored queue can be substituted in the task XML definition.

User-defined message properties are prefixed with `usr.`, but do not include this prefix in the variable name. Variable names must be preceded by a dollar sign ($) character and enclosed in braces, {}. For example, ${destFileName} is replaced with the value of the `usr.destFileName` message property of the

first message to be read from the source queue. For more information, see "WebSphere MQ message properties read from messages on source queues" on page 703 and, "Monitoring a queue and using variable substitution" on page 228

The following substitution variables are provided by default:

Variable	Description
AGENTNAME	The name of the resource monitor agent.
QUEUENAME	The name of the queue that is being monitored.
ENCODING	The character encoding of the first message on the queue or the first message in a group.
MESSAGEID	The WebSphere MQ message ID of the first message on the queue or the first message in the group.
GROUPID	The WebSphere MQ group ID of the group or the message ID if only a single message is found. This variable is only set if you are monitoring for complete groups.
CurrentTimeStamp	A time stamp based on the local time that the monitor triggered at. The time stamp value is unique for the agent.
CurrentTimeStampUTC	A time stamp based on the time, in the UTC time zone that the monitor triggered at. The time stamp value is unique for the agent.

For example, ${AGENTNAME} is replaced with the name of the resource monitor agent.

If the monitored resource is a file

The set of variable names that can be substituted in the task XML definition is as follows:

Variable	Description
FilePath	The complete path name of the trigger file.
FileName	The file name part of the trigger.
LastModifiedTime	The time that the trigger file was last modified. This time is expressed as the local time of the time zone the agent is running in and is formatted as an ISO 8601 time.
LastModifiedDate	The date that the trigger file was last modified. This date is expressed as the local date of the time zone the agent is running in and is formatted as an ISO 8601 date.
LastModifiedTimeUTC	The time that the trigger file was last modified. This time is expressed as the local time converted to the UTC time zone and is formatted as an ISO 8601 time.
LastModifiedDateUTC	The date that the trigger file was last modified. This date is expressed as the local date converted to the UTC time zone and is formatted as an ISO 8601 date.
AgentName	The name of the resource monitor agent.
CurrentTimeStamp	A time stamp that is based on the local time that the monitor triggered at. The time stamp value is unique for the agent.
CurrentTimeStampUTC	A time stamp that is based on the time, in the UTC time zone that the monitor triggered at. The time stamp value is unique for the agent.

Variable names must be preceded by a dollar sign ($) character and enclosed in braces, {}. For example, ${FilePath} is replaced with the fully qualified file path of the matching trigger file.

There are two special keywords that can be applied to variable names to provide further refinement. These are:
- token - token index to substitute (starting at 1 from the left and starting at -1 from the right)
- separator - single character to tokenize the variable value. The default is the forward slash character (/), but the separator can be any valid character that can appear in the variable value.

If the separator keyword is specified in a variable name, the variable value is split into tokens according to the separator character.

The value that is assigned to the token keyword is used as an index to select which token to use to replace the variable name. The token index is relative to the first character in the variable, and starts at 1. If the token keyword is not specified, the entire variable is inserted.

Variable names are not case-sensitive.

Any values that are substituted into an agent name in the message XML are treated in a not case-sensitive way. All WebSphere MQ Managed File Transfer agent names are uppercase. If the value "Paris" is substituted into an agent attribute in the message XML, this value is interpreted as a reference to the agent PARIS.

The following example explains the various behaviors:

Assuming the file path to the matching trigger file is c:\MONITOR\REPORTS\Paris\Report2009.doc, the variables are substituted as follows:

Variable specification	After variable substitution
${FilePath}	c:\MONITOR\REPORTS\Paris\Report2009.doc
${FilePath{token=1}{separator=.}}	c:\MONITOR\REPORTS\Paris\Report2009
${FilePath{token=2}{separator=.}}	doc
${FilePath{token=3}}	REPORTS

You can also specify a negative token index to select tokens relative to the last character of the variable. For example, by using the same variable value:

Variable specification	After variable substitution
${FilePath}	c:\MONITOR\REPORTS\Paris\Report2009.doc
${FilePath{token=-2}{separator=.}}	c:\MONITOR\REPORTS\Paris\Report2009
${FilePath{token=-2}{separator=\}}	Paris
${FilePath{token=-4}}	MONITOR

The variables that are used for substitution are only available for positive trigger conditions. Only match and fileSize trigger conditions cause variables to be substituted. If a noMatch condition is used, and there are substitution variable names in the task definition, the task is not called, and the monitor raises a return code of 103 and error message BFGDM0060E.

Example

The following example task definition XML uses the monitor agent name as the source agent for the transfer, uses the penultimate directory name in the file path as the destination agent name for the transfer, and renames the transferred file to be the root of the trigger file name with an extension of `.rpt`.

```
<?xml version="1.0" encoding="UTF-8" ?>
<request version="4.00" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
  <managedTransfer>
    <originator>
      <hostName>reportserver.com</hostName>
      <userID>USER1</userID>
    </originator>
    <sourceAgent agent="${AgentName}" QMgr="QM1" />
    <destinationAgent agent="${FilePath{token=-2}}" QMgr="QMD" />
    <transferSet>
      <item mode="binary" checksumMethod="MD5">
        <source recursive="false" disposition="leave">
          <file>c:/incoming/reports/summary/report.doc</file>
        </source>
        <destination type="file" exist="overwrite">
          <file>/reports/${FileName{token=1}{separator=.}}.rpt</file>
        </destination>
      </item>
    </transferSet>
  </managedTransfer>
</request>
```

This results in the task XML being transformed to:

```
<?xml version="1.0" encoding="UTF-8" ?>
<request version="4.00" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
  <managedTransfer>
    <originator>
      <hostName>reportserver.com</hostName>
      <userID>USER1</userID>
    </originator>
    <sourceAgent agent="AGENT1" QMgr="QM1" />
    <destinationAgent agent="Paris" QMgr="QMD" />
    <transferSet>
      <item mode="binary" checksumMethod="MD5">
        <source recursive="false" disposition="leave">
          <file>c:/incoming/reports/summary/report.doc</file>
        </source>
        <destination type="file" exist="overwrite">
          <file>/reports/Report2009.rpt</file>
        </destination>
      </item>
    </transferSet>
  </managedTransfer>
</request>
```

The variable `${FilePath{token=-2}}` in the `<destinationAgent>` element's `agent` attribute is replaced with the value "Paris". This value is treated in a not case-sensitive way and interpreted as a reference to the agent PARIS.

Monitoring a queue and using variable substitution

You can monitor a queue and transfer messages from the monitored queue to a file by using the **fteCreateMonitor** command. The value of any WebSphere MQ message property in the first message to be read from the monitored queue can be substituted in the task XML definition and used to define the transfer behavior.

About this task

In this example, the source agent is called AGENT_VENUS, which connects to QM_VENUS. The queue that AGENT_VENUS monitors is called START_QUEUE and is located on QM_VENUS. The agent polls the queue every 30 minutes.

When a complete group of messages is written to the queue the monitor task sends the group of messages to a file at one of a number of destination agents, all of which connect to the queue manager QM_MARS. The name of the file that the group of messages is transferred to is defined by the WebSphere MQ message property usr.fileName on the first message in the group. The name of the agent that the group of messages is sent to is defined by the WebSphere MQ message property usr.toAgent on the first message in the group. If the usr.toAgent header is not set, the default value to be used for the destination agent is AGENT_MAGENTA.

When you specify useGroups="true", if you do not also specify groupId="${GROUPID}", the transfer just takes the first message on the queue . For example, if you are using variable substitution to generate the fileName, it is therefore possible that the contents of a.txt will not be correct. This is because the fileName is generated by the monitor, but the transfer actually gets a message that is not the one that should generate the file called fileName.

Procedure

1. Create the task XML that defines the task that the monitor performs when it is triggered.

```
<?xml version="1.0" encoding="UTF-8" ?>
<request version="4.00"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
  <managedTransfer>
    <originator>
      <hostName>reportserver.com</hostName>
      <userID>USER1</userID>
    </originator>
    <sourceAgent agent="AGENT_VENUS" QMgr="QM_VENUS" />
    <destinationAgent agent="${toAgent}" QMgr="QM_MARS" />
    <transferSet>
      <item mode="binary" checksumMethod="none">
        <source>
          <queue useGroups="true" groupId="${GROUPID}">START_QUEUE</queue>
        </source>
        <destination type="file" exist="overwrite">
          <file>/reports/${fileName}.rpt</file>
        </destination>
      </item>
    </transferSet>
  </managedTransfer>
</request>
```

The variables that are replaced with the values of WebSphere MQ message headers are highlighted in **bold**. This task XML is saved to the file /home/USER1/task.xml

2. Create a resource monitor to monitor the queue START_QUEUE. Submit the following command:

```
fteCreateMonitor -ma AGENT_VENUS -mm QM_VENUS -mq START_QUEUE
                 -mn myMonitor -mt /home/USER1/task.xml
                 -tr completeGroups -pi 30 -pu minutes -dv toAgent=AGENT_MAGENTA
```

3. A user or program writes a group of messages to the queue START_QUEUE. The first message in this group has the following WebSphere MQ message properties set:

```
usr.fileName=larmer
usr.toAgent=AGENT_VIOLET
```

4. The monitor is triggered when the complete group is written. The agent substitutes the WebSphere MQ message properties into the task XML. This results in the task XML being transformed to:

```
<?xml version="1.0" encoding="UTF-8" ?>
<request version="4.00"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
  <managedTransfer>
    <originator>
      <hostName>reportserver.com</hostName>
      <userID>USER1</userID>
    </originator>
```

```
        <sourceAgent agent="AGENT_VENUS" QMgr="QM_VENUS" />
        <destinationAgent agent="AGENT_VIOLET" QMgr="QM_MARS" />
        <transferSet>
          <item mode="binary" checksumMethod="none">
            <source>
              <queue useGroups="true" groupId="${GROUPID}">START_QUEUE</queue>
            </source>
            <destination type="file" exist="overwrite">
              <file>/reports/larmer.rpt</file>
            </destination>
          </item>
        </transferSet>
      </managedTransfer>
  </request>
```

Results

The transfer that is defined by the task XML is performed. The complete group of messages that are read from the START_QUEUE by AGENT_VENUS is written to a file called /reports/larmer.rpt on the system where AGENT_VIOLET is running.

What to do next

Transferring each message to a separate file

If you want to monitor a queue and have every message transferred to a separate file, you can use a similar technique to the one described previously in this topic.

1. Create the monitor as described previously, specifying the **-tr completeGroups** parameter on the **fteCreateMonitor** command.
2. In the task XML specify the following:

   ```
   <queue useGroups="true" groupId="${GROUPID}">START_QUEUE</queue>
   ```

However, when you put the messages onto the source queue, do not put them in a IBM WebSphere MQ group. Add IBM WebSphere MQ message properties to each message. For example, specify the usr.filename property with a unique file name value for each message. This effectively causes the WebSphere MQ Managed File Transfer agent to treat each message on the source queue as a separate group.

Monitor retry behavior for message-to-file transfers

If a message-to-file transfer that is triggered by a resource monitor fails and leaves the message group that triggered the monitor on the queue, that transfer is resubmitted at subsequent poll intervals. The number of times that the transfer is resubmitted is limited by the **monitorGroupRetryLimit** property of the monitoring agent.

The number of times that the message-to-file transfer has been triggered is determined from the MQMD backout count of the first message in the group.

Each time a new message-to-file transfer is triggered a new transfer ID is generated for the transfer task.

If the agent is restarted the monitor triggers a transfer again even if the number of times the transfer has been triggered has exceeded the value of **monitorGroupRetryLimit**. If this transfer attempt causes the number of times that the transfer has been triggered to exceed the value of **monitorGroupRetryLimit**, the agent writes an error to its event log.

A single message is treated as if it was a single group, and the transfer is triggered again at each poll interval while the message remains on the queue and while the number of times the transfer has been triggered is less than the value of **monitorGroupRetryLimit**.

Setting the monitorGroupRetryLimit property

The value of the **monitorGroupRetryLimit** property is the maximum number of times that a monitor triggers a message-to-file transfer again if the message group still exists on the queue. The default value of this property is 10. The value of this property can be set to any positive integer value or -1. If the value -1 is specified for this property, the monitor triggers the transfer again an unlimited number of times, until the trigger condition is not satisfied.

To set the **monitorGroupRetryLimit** property on the monitoring agent, perform the following steps:

1. Stop the monitoring agent, using the **fteStopAgent** command.
2. Edit the monitoring agent `agent.properties` file to include the line `monitorGroupRetryLimit=`*number_of_retries*. The `agent.properties` file is located in the directory *MQ_DATA_PATH*`/mqft/config/`*coordination_qmgr_name*`/agents/`*monitoring_agent_name*.
3. Start the monitoring agent, using the **fteStartAgent** command.

Related tasks:

"Example: Configuring a resource monitor to monitor a queue" on page 225
You can specify a WebSphere MQ queue as the resource to be monitored by a resource monitor by using the **-mq** parameter with the **fteCreateMonitor** command.

Related reference:

"The agent.properties file" on page 548
Each agent has its own properties file, `agent.properties`, that must contain the information that an agent uses to connect to its queue manager. The `agent.properties` file can also contain properties that alter the behavior of the agent.

Working with transfer templates

You can use file transfer templates to store common file transfer settings for repeated or complex transfers. Either create a transfer template from the command line by using the **fteCreateTemplate** command or use the WebSphere MQ Explorer to create a transfer template by using the **Create New Template for Managed File Transfer** wizard, or save a template while you are creating a file transfer by selecting the **Save transfer settings as a template** check box. The **Transfer Templates** window displays all of the transfer templates that you have created in your WebSphere MQ Managed File Transfer network.

About this task

To create a transfer template from the command line, use the fteCreateTemplate command. Then when you want to submit a transfer template that you created on the command line, click **Submit** in WebSphere MQ Explorer.

To view transfer templates in the WebSphere MQ Explorer, use the following steps:

Procedure

1. Expand **Managed File Transfer** in the Navigator view. **Managed File Transfer Central** is displayed in the Content view.
2. All of your coordination queue managers are listed in the Navigator view. Expand the name of the coordination queue manager that you have used for the scheduled transfer. If you want to change which coordination queue manager you are connected to, right-click the name of the coordination queue manager you want to use in Navigator view and click **Connect**.
3. Click **Transfer Templates**. The Transfer Templates window is displayed in the Content view.
4. The Transfer Templates window lists the following details about your file transfers:
   a. **Name** The name of your file transfer template.
   b. **Source** The name of the agent used to transfer the file from the source system.

c. **Source File** The name of the file to be transferred on its host system. Expand the transfer template information to view this field.

d. **Destination** The name of the agent used to receive the file at the destination system.

e. **Destination File** The name of the file after it is transferred to the destination system. Expand t the transfer template information to view this field.

f. **Scheduled Start (selected time zone)** The time and date that the file transfer is scheduled to start in the time zone used by the administrator. To change the time zone displayed, click **Window** > **Preferences** > **WebSphere MQ Explorer** > **Managed File Transfer** and select an alternative time zone from the **Time zone:** list. Click **OK**.

g. **Trigger Events** The type of event that triggers the file transfer to start. The type can be one of the following values: `exists`, `does not exist`, or `exceeds`.

Results

To refresh what is displayed in the **Transfer Templates** window, click the Refresh button on the Content view toolbar.

To submit a transfer template and start the transfer defined in the template, right-click the template name and click **Submit**.

To change a transfer template, right-click the template name and click **Edit**. All files included in the original template are listed as part of a transfer group, even if they were not included as part of a group in the original template. If you want to remove a file from the template you must select the file specification from the group and click **Remove selected**. If you want to add new file specifications to the template use the fields in the template panel and click the **Add to group** button. When you have made your edits, you are prompted to give the edited template a new name.

To create a file transfer from a transfer template, right-click the template name and click **Edit as New Transfer**.

To create a duplicate copy of a transfer template, right-click the template name and click **Duplicate**. The duplicate transfer template is automatically saved with the same name as the original template, appended with "(copy)".

To delete a transfer template, right-click the template name and click **Delete**.

Related tasks:

"Creating a file transfer template using the WebSphere MQ Explorer" on page 233
You can create a file transfer template from the WebSphere MQ Explorer or from the command line. You can then use that template to create new file transfers using the template details or submit the template to start the file transfer.

Related reference:

"**fteCreateTemplate** (create new file transfer template)" on page 449
The **fteCreateTemplate** command creates a file transfer template that you can keep for future use. The only required parameter is the **-tn** (*template_name*) parameter. All other parameters are optional, although if you specify a source file specification, you must also provide a destination file. Similarly, if you specify a destination file, you must also specify a source file specification.

"fteListTemplates (list WebSphere MQ Managed File Transfer templates)" on page 500
Use the **fteListTemplates** command to list the available WebSphere MQ Managed File Transfer transfer templates on a coordination queue manager.

"fteDeleteTemplates (delete WebSphere MQ Managed File Transfer templates)" on page 491
Use the **fteDeleteTemplates** command to delete an existing WebSphere MQ Managed File Transfer template from a coordination queue manager.

Creating a file transfer template using the WebSphere MQ Explorer

You can create a file transfer template from the WebSphere MQ Explorer or from the command line. You can then use that template to create new file transfers using the template details or submit the template to start the file transfer.

About this task

To create a file transfer template from the command line, use the fteCreateTemplate command.

To create a file transfer template using the **Create New Template for Managed File Transfer** wizard in WebSphere MQ Explorer, use the following steps:

Procedure

1. In the Navigator view, click **Managed File Transfer**. **Managed File Transfer Central** is displayed in the Content view.
2. All of your coordination queue managers are displayed in the Navigator view. Expand the name of the coordination queue manager that you have used for the scheduled transfer. If you want to change which coordination queue manager you are connected to, right-click the name of the coordination queue manager you want to use in Navigator view and click **Connect**.
3. Start the **Create New Template for Managed File Transfer** wizard by right-clicking **Transfer Templates** and then clicking **New Template**.
4. Follow the instructions on the wizard panels. There is context-sensitive help provided for each panel. To access the context-sensitive help on Windows, press F1. On Linux, press `Ctrl+F1` or `Shift+F1`.

   If you have created a template that contains all the required transfer details, ensure that you select the **Save transfer settings as a template** check box on the **Transfer Summary** page if this check box is not already selected. Also enter a name for the template in the Name field. If you create a template that does not yet contain all of the required transfer details, the **Save transfer settings as a template** check box is automatically checked for you.

Related tasks:
"Working with transfer templates" on page 231
You can use file transfer templates to store common file transfer settings for repeated or complex transfers. Either create a transfer template from the command line by using the **fteCreateTemplate** command or use the WebSphere MQ Explorer to create a transfer template by using the **Create New Template for Managed File Transfer** wizard, or save a template while you are creating a file transfer by selecting the **Save transfer settings as a template** check box. The **Transfer Templates** window displays all of the transfer templates that you have created in your WebSphere MQ Managed File Transfer network.

Related reference:
"**fteCreateTemplate** (create new file transfer template)" on page 449
The **fteCreateTemplate** command creates a file transfer template that you can keep for future use. The only required parameter is the **-tn** (*template_name*) parameter. All other parameters are optional, although if you specify a source file specification, you must also provide a destination file. Similarly, if you specify a destination file, you must also specify a source file specification.

"fteListTemplates (list WebSphere MQ Managed File Transfer templates)" on page 500
Use the **fteListTemplates** command to list the available WebSphere MQ Managed File Transfer transfer templates on a coordination queue manager.

"fteDeleteTemplates (delete WebSphere MQ Managed File Transfer templates)" on page 491
Use the **fteDeleteTemplates** command to delete an existing WebSphere MQ Managed File Transfer template from a coordination queue manager.

Transfer data from files to messages

You can use the file-to-message feature of WebSphere MQ Managed File Transfer to transfer data from a file to a single message, or multiple messages, on a WebSphere MQ queue.

To perform file-to-message and message-to-file transfers both the source and destination agent of the transfer must either be at IBM WebSphere MQ Version 7.5 or at WebSphere MQ File Transfer Edition Version 7.0.3, or later. For information about message-to-file transfers, see "Transferring data from messages to files" on page 242.

The destination agent for a file-to-message transfer cannot be a protocol bridge agent or a Connect:Direct bridge agent.

You can transfer file data to WebSphere MQ message data. The WebSphere MQ messages can be read and used by applications. The following types of file-to-message transfer are supported:

- From a single file to a single message. The message does not have a WebSphere MQ group ID set.
- From a single file to multiple messages, by splitting the file into messages of a given length. The messages all have the same WebSphere MQ group ID.
- From a single file to multiple messages, by splitting a text file at a Java regular expression delimiter. The messages all have the same WebSphere MQ group ID.
- From a single file to multiple messages, by splitting a binary file at a hexadecimal delimiter. The messages all have the same WebSphere MQ group ID.

  If you want to split a binary file using a sequence of bytes as the delimiter, use the **-sqdb** parameter of the **fteCreateTransfer** command. For more information, see -sqdb parameter.

By default the messages created by a file-to-message transfer are persistent. The messages can be set to be non-persistent or to have the persistence value defined by the destination queue.

If you specify that a file is split into multiple messages, all messages created from the file have the same WebSphere MQ group ID. If you do not specify that a file is split into multiple messages, only one message is created from the file and this message does not have the WebSphere MQ group ID set.

If you are transferring files to large messages, or many small messages, you might need to change some WebSphere MQ or WebSphere MQ Managed File Transfer properties. For information about, see "Guidance for setting WebSphere MQ attributes and WebSphere MQ Managed File Transfer properties associated with message size" on page 363.

Note: If the destination queue is either a clustered queue, or an alias to a clustered queue, you will get an error message when transferring a file into a queue . For more information see "What to do if the destination queue is a clustered queue, or an alias to a clustered queue" on page 360

Configuring an agent to perform file-to-message transfers

By default agents cannot perform file-to-message or message-to-file transfers. To enable this function you must set the agent property enableQueueInputOutput to true.

About this task

If you attempt to perform a file-to-message transfer to a destination agent that does not have the enableQueueInputOutput property set to true, the transfer fails. The transfer log message that is published to the coordination queue manager contains the following message:

```
BFGIO0197E: An attempt to write to a queue was rejected by the destination agent. The
agent must have enableQueueInputOutput=true set in the agent.properties file to
support transferring to a queue.
```

To enable the agent to write to and read from queues perform the following steps:

Procedure

1. Stop the destination agent using the **fteStopAgent** command.

2. Edit the `agent.properties` file to include the line `enableQueueInputOutput=true`. The `agent.properties` file is located in the directory *MQ_DATA_PATH*/mqft/config/*coordination_queue_manager*/agents/*destination_agent_name*.

3. Start the destination agent using the **fteStartAgent** command.

Example: Transferring a single file to a single message

You can specify a queue as the destination of a file transfer by using the **-dq** parameter with the **fteCreateTransfer** command. The source file must be smaller than the maximum message length set on the destination queue. The destination queue does not have to be on the same queue manager as the queue manager that the destination agent connects to, but these two queue managers must be able to communicate.

About this task

The source file is called /tmp/`single_record.txt` and is located on the same system as the source agent, AGENT_NEPTUNE. The source agent, AGENT_NEPTUNE, uses the queue manager QM_NEPTUNE. The destination agent is AGENT_VENUS and this agent connects to the queue manager QM_VENUS. The destination queue, RECEIVING_QUEUE, is located on the queue manager QM_MERCURY. QM_MERCURY is in the same WebSphere MQ network as, and can be accessed by, the queue manager QM_VENUS.

Procedure

Type the following command:

```
fteCreateTransfer -sa AGENT_NEPTUNE -sm QM_NEPTUNE -da AGENT_VENUS -dm QM_VENUS
                -dq RECEIVING_QUEUE@QM_MERCURY /tmp/single_record.txt
```

If the destination queue is on a different queue manager to the queue manager used by the destination agent you must specify the value of the **-dq** parameter in the following format *queue_name@queue_manager_name*. If you do not specify *@queue_manager_name* in the value, the destination agent assumes that the destination queue is located on its own queue manager.
The source agent, AGENT_NEPTUNE, reads the data from the file /tmp/`single_record.txt` and transfers this data to the destination agent, AGENT_VENUS. The destination agent, AGENT_VENUS, sends the data to a persistent message on the queue RECEIVING_QUEUE@QM_MERCURY. The message does not have a WebSphere MQ group ID set.

Example: Splitting a single file into multiple messages by length

You can split a file into multiple WebSphere MQ messages by using the **-qs** parameter of the **fteCreateTransfer** command. The file is split into fixed-length sections, each of which is written to an individual message.

About this task

The source file is called /tmp/`source.file` and is 36 KB in size. The source file is located on the same system as the source agent AGENT_NEPTUNE. The source agent, AGENT_NEPTUNE, connects to the queue manager QM_NEPTUNE. The destination agent is AGENT_MERCURY, which connects to the queue manager QM_MERCURY. The destination queue, RECEIVING_QUEUE, is also located on the queue manager QM_MERCURY. The transfer splits the source file into sections that are 1 KB in size and writes each of these sections to a message on RECEIVING_QUEUE.

Procedure

Type the following command:

```
fteCreateTransfer -sa AGENT_NEPTUNE -sm QM_NEPTUNE -da AGENT_MERCURY -dm QM_MERCURY
                -dq RECEIVING_QUEUE -qs 1K /tmp/source.file
```

The source agent, AGENT_NEPTUNE, reads the data from the file /tmp/source.file and transfers this data to the destination agent, AGENT_MERCURY. The destination agent, AGENT_MERCURY, writes the data to thirty-six 1 KB persistent messages on the queue RECEIVING_QUEUE@QM_MERCURY. These messages all have the same WebSphere MQ group ID and the last message in the group has the WebSphere MQ LAST_MSG_IN_GROUP flag set.

Example: Splitting a text file into multiple messages using a regular expression delimiter

Transfer a single text file to multiple messages by splitting the file at each match of a given Java regular expression. To do this you use the **-dqdt** parameter of the **fteCreateTransfer** command.

About this task

The file is split into variable-length sections, each of which is written to an individual message. The text file is split at each point where the text in the file matches a given regular expression. The source file is called /tmp/names.text and has the following contents:

```
Jenny Jones,John Smith,Jane Brown
```

The regular expression that specifies where to split the file is the comma character (**,**).

The source file is located on the same system as the source agent *AGENT_NEPTUNE*, which connects to the queue manager QM_NEPTUNE. The destination queue, *RECEIVING_QUEUE*, is located on the queue manager *QM_MERCURY*. *QM_MERCURY* is also the queue manager used by the destination agent *AGENT_MERCURY*. The transfer splits the source file into sections and writes each of these sections to a message on *RECEIVING_QUEUE*.

Procedure

Type the following command:

```
fteCreateTransfer -sa AGENT_NEPTUNE -sm QM_NEPTUNE -da AGENT_MERCURY -dm QM_MERCURY
                  -dq RECEIVING_QUEUE -t text -dqdp postfix -dqdt "," /tmp/names.text
```

The source agent, AGENT_NEPTUNE, reads the data from the file /tmp/names.text and transfers this data to the destination agent, AGENT_MERCURY. The destination agent, AGENT_MERCURY, writes the data to three persistent messages on the queue *RECEIVING_QUEUE*. These messages all have the same WebSphere MQ group ID and the last message in the group has the WebSphere MQ LAST_MSG_IN_GROUP flag set.
The data in the messages is as follows.

- First message:

  ```
  Jenny Jones
  ```

- Second message:

  ```
  John Smith
  ```

- Third message:

  ```
  Jane Brown
  ```

Example: Splitting a text file with a regular expression delimiter and including the delimiter in the messages

Transfer a single text file to multiple messages by splitting the file at each match of a given Java regular expression and include the regular expression match in the resulting messages. To do this you use the **-dqdt** and **-qi** parameters of the **fteCreateTransfer** command.

About this task

Transfer a single text file to multiple messages on a queue. The file is split into variable-length sections, each of which is written to an individual message. The text file is split at each point where the text in the file matches a given regular expression. The source file is called /tmp/customers.text and has the following contents:

```
Customer name: John Smith
Customer contact details: john@example.net
Customer number: 314

Customer name: Jane Brown
Customer contact details: jane@example.com
Customer number: 42

Customer name: James Jones
Customer contact details: jjones@example.net
Customer number: 26
```

The regular expression that specifies where to split the file is Customer\snumber:\s\d+, which matches the text "Customer number: " followed by any number of digits. Regular expressions specified at the command line must be enclosed in double quotation marks to prevent the command shell evaluating the regular expression. The regular expression is evaluated as a Java regular expression. For more information, see "Regular expressions used by WebSphere MQ Managed File Transfer" on page 685.

By default the number of characters that a regular expression can match is set to five. The regular expression used in this example matches strings that are longer than five characters. To enable matches that are longer than five characters edit the agent properties file to include the property **maxDelimiterMatchLength**.

By default, the text that matches the regular expression is not included in the messages. To include the text that matches the regular expression in the messages, as in this example, use the **-qi** parameter. The source file is located on the same system as the source agent AGENT_NEPTUNE, which connects to the queue manager QM_NEPTUNE. The destination queue, RECEIVING_QUEUE, is located on the queue manager QM_MERCURY. QM_MERCURY is also the queue manager used by the destination agent AGENT_MERCURY. The transfer splits the source file into sections and writes each of these sections to a message on RECEIVING_QUEUE.

Procedure

1. Stop the destination agent using the following command:

   ```
   fteStopAgent AGENT_MERCURY
   ```
2. Add the following line to the agent properties file for AGENT_MERCURY:

   ```
   maxDelimiterMatchLength=25
   ```

   Note: Increasing the value of **maxDelimiterMatchLength** can decrease performance.
3. Start the destination agent using the following command:

   ```
   fteStartAgent AGENT_MERCURY
   ```
4. Type the following command:

```
fteCreateTransfer -sa AGENT_NEPTUNE -sm QM_NEPTUNE -da AGENT_MERCURY -dm QM_MERCURY
 -dq RECEIVING_QUEUE
     text -dqdt "Customer\snumber:\s\d+" -qi -dqdp postfix /tmp/customers.text
```

The source agent, AGENT_NEPTUNE, reads the data from the file /tmp/customers.text and transfers this data to the destination agent, AGENT_MERCURY. The destination agent, AGENT_MERCURY, writes the data to three persistent messages on the queue RECEIVING_QUEUE. These messages all have the same WebSphere MQ group ID and the last message in the group has the WebSphere MQ LAST_MSG_IN_GROUP flag set.

The data in the messages is as follows.

- First message:

  ```
  Customer name: John Smith
  Customer contact details: john@example.net
  Customer number: 314
  ```

- Second message:

  ```
  Customer name: Jane Brown
  Customer contact details: jane@example.com
  Customer number: 42
  ```

- Third message:

  ```
  Customer name: James Jones
  Customer contact details: jjones@example.net
  Customer number: 26
  ```

Example: Setting WebSphere MQ message properties on a file-to-message transfer

You can use the **-qmp** parameter on the **fteCreateTransfer** command to specify whether WebSphere MQ message properties are set on the first message written to the destination queue by the transfer. WebSphere MQ message properties allow an application to select messages to process, or to retrieve information about a message without accessing WebSphere MQ Message Descriptor (MQMD) or MQRFH2 headers.

About this task

Include the parameter -qmp true in the **fteCreateTransfer** command. In this example, the MQMD user ID of the user submitting the command is larmer.

Procedure

Type the following command:

```
fteCreateTransfer -sa AGENT_JUPITER -da AGENT_SATURN -dq MY_QUEUE@MyQM -qmp true
     -t text /tmp/source_file.txt
```

The WebSphere MQ message properties of the first message written by the destination agent, AGENT_SATURN, to the queue, MY_QUEUE, on queue manager, MyQM, are set to these values:

```
usr.WMQFTETransferId=414cbaedefa234889d999a8ed09782395ea213ebbc9377cd
usr.WMQFTETransferMode=text
usr.WMQFTESourceAgent=AGENT_JUPITER
usr.WMQFTEDestinationAgent=AGENT_SATURN
usr.WMQFTEFileName=source_file.txt
usr.WMQFTEFileSize=1024
usr.WMQFTEFileLastModified=1273740879040
usr.WMQFTEFileIndex=0
usr.WMQFTEMqmdUser=larmer
```

Example: Setting user-defined properties on a file-to-message transfer

User-defined metadata is set as a WebSphere MQ message property on the first message written to the destination queue by the transfer. WebSphere MQ message properties enable an application to select messages to process, or to retrieve information about a message without accessing WebSphere MQ Message Descriptor (MQMD) or MQRFH2 headers.

About this task

Include the parameters **-qmp true** and **-md account=123456** in the **fteCreateTransfer** command, to set the usr.account property to 123456 in the RFH2 header.

Procedure

Type the following command:

```
fteCreateTransfer -sa AGENT_JUPITER -da AGENT_SATURN -dq MY_QUEUE@MyQM
                  -qmp true -md account=123456 /tmp/source_file.txt
```

In addition to the standard set of WebSphere MQ message properties, the user-defined property is set in the message header of the first message written by the destination agent, AGENT_SATURN, to the queue, MY_QUEUE, on queue manager, MyQM. The header is set to the following value:

```
usr.account=123456
```

The prefix usr is added to the beginning of the name of the user-defined metadata.

Example: adding a user-defined message property for a file-to-message transfer

If you are using WebSphere MQ Managed File Transfer for message-to-file managed transfers, you can include a user-defined message property for the resulting message.

About this task

You can use any of the following methods to define a custom message property:

- Specify the **-md** parameter on the transfer request. For more information, see "Example: Setting user-defined properties on a file-to-message transfer."
- Use an Ant task; you can use either fte:filecopy or fte:filemove. The following example is a fte:filecopy task:

```
<project xmlns:fte="antlib:com.ibm.wmqfte.ant.taskdefs" default="complete">
<!-- Initialise the properties used in this script.-->

<target name="init" description="initialise task properties">
            <property name="src.file" value="/home/user/file1.bin"/>
            <property name="dst.queue" value="TEST.QUEUE@qm2"/>
            <fte:uuid property="job.name" length="8"
prefix="copyjob#"/>
</target>
<target name="step1" depends="init" description="transfer file">

<fte:filecopy cmdqm="qm0@localhost@1414@SYSTEM.DEF.SVRCONN"
            src="agent1@qm1" dst="agent2@qm2"
          rcproperty="copy.result">

<fte:metadata>
<fte:entry name="fileName" value="${FileName}"/>
</fte:metadata>

<fte:filespec srcfilespec="${src.file}" dstqueue="${dst.queue}"
dstmsgprops="true"/>

</fte:filecopy>
```

```
    </target>
  </project>
```

- Use a resource monitor and variable substitution. The following example shows some transfer task XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<monitor:monitor
xmlns:monitor="http://www.ibm.com/xmlns/wmqfte/7.0.1/MonitorDefinition"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="5.00"
xsi:schemaLocation="http://www.ibm.com/xmlns/wmqfte/7.0.1/MonitorDefinit ion ./Monitor.xsd">
 <name>METADATA</name>
 <pollInterval units="minutes">5</pollInterval>
 <batch maxSize="5"/>
 <agent>AGENT1</agent>
 <resources>
   <directory recursionLevel="0">e:\temp</directory>
 </resources>
 <triggerMatch>
   <conditions>
     <allOf>
       <condition>
         <fileMatch>
           <pattern>*.txt</pattern>
         </fileMatch>
       </condition>
     </allOf>
   </conditions>
 </triggerMatch>
 <tasks>
   <task>
     <name/>
     <transfer>
       <request version="5.00"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
         <managedTransfer>
           <originator>
             <hostName>mqjason.raleigh.ibm.com.</hostName>
             <userID>administrator</userID>
           </originator>
           <sourceAgent QMgr="AGENTQM" agent="AGENT1"/>
           <destinationAgent QMgr="AGENTQM" agent="AGENT2"/>
           <transferSet priority="0">
             <metaDataSet>
               <metaData key="FileName">${FileName}</metaData>
             </metaDataSet>
             <item checksumMethod="MD5" mode="text">
               <source disposition="delete" recursive="false">
                 <file>${FilePath}</file>
               </source>
               <destination type="queue">
                 <queue persistent="true"
setMqProps="true">TEST.QUEUE@AGENTQM</queue>
               </destination>
             </item>
           </transferSet>
           <job>
             <name>Metadata_example</name>
           </job>
         </managedTransfer>
       </request>
     </transfer>
   </task>
 </tasks>
 <originator>
```

240 IBM WebSphere MQ: Managed File Transfer

```
      <hostName>mqjason.raleigh.ibm.com.</hostName>
      <userID>administrator</userID>
    </originator>
  </monitor:monitor>
```

Related tasks:

"Example: Setting WebSphere MQ message properties on a file-to-message transfer" on page 238
You can use the **-qmp** parameter on the **fteCreateTransfer** command to specify whether WebSphere MQ message properties are set on the first message written to the destination queue by the transfer. WebSphere MQ message properties allow an application to select messages to process, or to retrieve information about a message without accessing WebSphere MQ Message Descriptor (MQMD) or MQRFH2 headers.

Related reference:

"fte:filecopy" on page 903
The **fte:filecopy** task copies files between WebSphere MQ Managed File Transfer agents. The file is not deleted from the source agent.

"fte:filemove" on page 906
The **fte:filemove** task moves files between WebSphere MQ Managed File Transfer agents. When a file has been successfully transferred from the source agent to the destination agent, the file is deleted from the source agent.

Failure of a file to message transfer

If a file-to- message transfer fails after the agent has started writing file data to the destination queue, the agent writes a message to the queue to indicate to an application consuming the messages that a failure has occurred.

The message written to the destination queue if a failure occurs:

• is blank
• has the same WebSphere MQ group ID as the previous message written to the destination queue by the agent
• has the WebSphere MQ LAST_MSG_IN_GROUP flag set
• contains additional WebSphere MQ message properties, if message properties are enabled. For more information, see the topic "Failure properties" on page 701.

Example

A transfer is requested by running the following command:

```
fteCreateTransfer -sa AGENT_JUPITER -da AGENT_SATURN -dq RECEIVING_QUEUE
                  -qmp true -qs 1K /tmp/source1.txt
```

The file `source1.txt` is 48 KB. The transfer splits this file into 1 KB messages and writes these messages to the destination queue RECEIVING_QUEUE.

While the transfer is in progress, after the agent has written 16 messages to RECEIVING_QUEUE, a failure occurs at the source agent.

The agent writes a blank message to RECEIVING_QUEUE. In addition to the standard set of message properties, the blank message has the following message properties set:

```
usr.WMQFTEResultCode = 40
usr.WMQFTESupplement = BFGTR0036I: The transfer failed to complete successfully.
```

Transferring data from messages to files

The message-to-file feature of WebSphere MQ Managed File Transfer enables you to transfer data from one or more messages on a WebSphere MQ queue to a file or a user file space. If you have an application that creates or processes WebSphere MQ messages, you can use the message-to-file capability of WebSphere MQ Managed File Transfer to transfer these messages to a file on any system in your WebSphere MQ Managed File Transfer network.

For information about file-to-message transfers, see "Transfer data from files to messages" on page 233.

The source agent for a message-to-file transfer cannot be a protocol bridge agent or a Connect:Direct bridge agent.

You can transfer WebSphere MQ message data to a file. The following types of message-to-file transfer are supported:
* From a single message to a single file
* From multiple messages to a single file
* From multiple messages with the same WebSphere MQ group ID to a single file.
* From multiple messages to a single file, including a text or binary delimiter between the data from each message written to the file.

If you are transferring files from large messages, or many small messages, you might need to change some WebSphere MQ or WebSphere MQ Managed File Transfer properties. For more information about, see "Guidance for setting WebSphere MQ attributes and WebSphere MQ Managed File Transfer properties associated with message size" on page 363.

Configuring an agent to perform message to file transfers

By default agents cannot perform message to file, or file to message, transfers. To enable this function you must set the agent property `enableQueueInputOutput` to true.

About this task

If you attempt to perform a message to file transfer from a source agent that does not have the `enableQueueInputOutput` property set to true, the transfer fails. The transfer log message that is published to the coordination queue manager contains the following message:

```
BFGIO0197E: An attempt to read from a queue was rejected by the source agent.
The agent must have enableQueueInputOutput=true set in the agent.properties file
to support transferring from a queue.
```

To enable the agent to write to and read from queues perform the following steps:

Procedure

1. Stop the source agent using the **fteStopAgent** command.
2. Edit the `agent.properties` file to include the line `enableQueueInputOutput=true`. The `agent.properties` file is located in the directory *MQ_DATA_PATH*/mqft/config/*coordination_queue_manager*/agents/*source_agent_name.*
3. Start the source agent using the **fteStartAgent** command.

Example: Transferring from a queue to a single file

You can specify a WebSphere MQ queue as the source of a file transfer by using the **-sq** parameter with the **fteCreateTransfer** command.

About this task

The source data is contained in three messages on the queue START_QUEUE. This queue must be on the source agent's queue manager, *QM_NEPTUNE.*

Procedure

Type the following command:

```
fteCreateTransfer -sa AGENT_NEPTUNE -sm QM_NEPTUNE
                  -da AGENT_VENUS -df /out/three_to_one.txt
                  -sq START_QUEUE
```

The data in the messages on the queue *START_QUEUE* is written to the file /out/three_to_one.txt on the system where AGENT_VENUS is running.

Example: Transferring a group of messages from a queue to a single file

You can specify a single complete group on a WebSphere MQ queue as the source of a file transfer by using the **-sq** and **-sqgi** parameters with the **fteCreateTransfer** command.

About this task

In this example, there are ten messages on the queue START_QUEUE. This queue must be on the source agent's queue manager, *QM_NEPTUNE.* The first three messages belong to a group with the WebSphere MQ group ID 41424b3ef3a22020202020202020202020202020201111; this group is not a complete group. The next five messages belong to a group with the WebSphere MQ group ID 41424b3ef3a22020202020202020202020202020202222; this group is complete. The remaining two messages belong to a group with the WebSphere MQ group ID 41424b3ef3a22020202020202020202020202020203333; this group is complete.

Procedure

Type the following command:

```
fteCreateTransfer -sa AGENT_NEPTUNE -sm QM_NEPTUNE -da AGENT_VENUS
                  -df /out/group.txt -sqgi -sq START_QUEUE
```

The data in the messages belonging to the first complete group on the queue *START_QUEUE*, the group with WebSphere MQ group ID 41424b3ef3a22020202020202020202020202020202222, is written to the file /out/group.txt on the system where AGENT_VENUS is running.

Example: Inserting a text delimiter before the data from each message

When you are transferring in text mode from a source queue to a file, you can specify that a text delimiter is inserted before the data from individual messages by using the **-sq**, **-sqdt** and **-sqdp** parameters with the **fteCreateTransfer** command.

About this task

In this example, there are four messages on the queue START_QUEUE. This queue is on the source agent's queue manager, *QM_NEPTUNE*. The text delimiter to be inserted before the data from each message can be expressed as a Java literal string, for example: \n\u002D\u002D\u002D\n.

Procedure

Type the following command:

```
fteCreateTransfer -sa AGENT_NEPTUNE -sm QM_NEPTUNE -da AGENT_VENUS -df /out/output.txt
                  -t text -sqdt "\n\u002D\u002D\u002D\n" -sqdp prefix -sq START_QUEUE
```

The text delimiter is added to the beginning of the data from each of the four messages on START_QUEUE by the source agent, AGENT_NEPTUNE. This data is written to the destination file, /out/output.txt.

Example: Inserting a binary delimiter after the data from each message

When transferring in binary mode from a source queue to a file, you can specify that a binary delimiter is inserted after the data from individual messages by using the **-sq**, **-sqdb**, and **-sqdp** parameters with the **fteCreateTransfer** command.

About this task

In this example, there are three messages on the queue START_QUEUE. This queue is on the source agent's queue manager, *QM_NEPTUNE*. The binary delimiter to be inserted after the data from each message must be expressed as a comma-separated list of hexadecimal bytes, for example: x34**,**xE7**,**xAE.

Procedure

Type the following command:

```
fteCreateTransfer -sa AGENT_NEPTUNE -sm QM_NEPTUNE -da AGENT_VENUS -df /out/binary.file
                  -sqdp postfix -sqdb x34,xE7,xAE -sq START_QUEUE
```

The binary delimiter is appended to the data from each of the three messages on START_QUEUE by the source agent, AGENT_NEPTUNE. This data is written to the destination file, /out/binary.file.

Monitoring a queue and using variable substitution

You can monitor a queue and transfer messages from the monitored queue to a file by using the **fteCreateMonitor** command. The value of any WebSphere MQ message property in the first message to be read from the monitored queue can be substituted in the task XML definition and used to define the transfer behavior.

About this task

In this example, the source agent is called AGENT_VENUS, which connects to QM_VENUS. The queue that AGENT_VENUS monitors is called START_QUEUE and is located on QM_VENUS. The agent polls the queue every 30 minutes.

When a complete group of messages is written to the queue the monitor task sends the group of messages to a file at one of a number of destination agents, all of which connect to the queue manager QM_MARS. The name of the file that the group of messages is transferred to is defined by the WebSphere MQ message property `usr.fileName` on the first message in the group. The name of the agent that the group of messages is sent to is defined by the WebSphere MQ message property `usr.toAgent` on the first message in the group. If the `usr.toAgent` header is not set, the default value to be used for the destination agent is AGENT_MAGENTA.

When you specify useGroups="true", if you do not also specify groupId="${GROUPID}", the transfer just takes the first message on the queue . For example, if you are using variable substitution to generate the fileName, it is therefore possible that the contents of `a.txt` will not be correct. This is because the fileName is generated by the monitor, but the transfer actually gets a message that is not the one that should generate the file called fileName.

Procedure

1. Create the task XML that defines the task that the monitor performs when it is triggered.

```
<?xml version="1.0" encoding="UTF-8" ?>
<request version="4.00"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
  <managedTransfer>
    <originator>
      <hostName>reportserver.com</hostName>
      <userID>USER1</userID>
    </originator>
    <sourceAgent agent="AGENT_VENUS" QMgr="QM_VENUS" />
    <destinationAgent agent="${toAgent}" QMgr="QM_MARS" />
    <transferSet>
      <item mode="binary" checksumMethod="none">
        <source>
          <queue useGroups="true" groupId="${GROUPID}">START_QUEUE</queue>
        </source>
        <destination type="file" exist="overwrite">
          <file>/reports/${fileName}.rpt</file>
        </destination>
      </item>
    </transferSet>
  </managedTransfer>
</request>
```

   The variables that are replaced with the values of WebSphere MQ message headers are highlighted in **bold**. This task XML is saved to the file `/home/USER1/task.xml`

2. Create a resource monitor to monitor the queue START_QUEUE. Submit the following command:

```
fteCreateMonitor -ma AGENT_VENUS -mm QM_VENUS -mq START_QUEUE
                 -mn myMonitor -mt /home/USER1/task.xml
                 -tr completeGroups -pi 30 -pu minutes -dv toAgent=AGENT_MAGENTA
```

3. A user or program writes a group of messages to the queue START_QUEUE. The first message in this group has the following WebSphere MQ message properties set:

```
usr.fileName=larmer
usr.toAgent=AGENT_VIOLET
```

4. The monitor is triggered when the complete group is written. The agent substitutes the WebSphere MQ message properties into the task XML. This results in the task XML being transformed to:

```
<?xml version="1.0" encoding="UTF-8" ?>
<request version="4.00"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
  <managedTransfer>
    <originator>
      <hostName>reportserver.com</hostName>
      <userID>USER1</userID>
    </originator>
```

```
      <sourceAgent agent="AGENT_VENUS" QMgr="QM_VENUS" />
      <destinationAgent agent="AGENT_VIOLET" QMgr="QM_MARS" />
      <transferSet>
        <item mode="binary" checksumMethod="none">
          <source>
            <queue useGroups="true" groupId="${GROUPID}">START_QUEUE</queue>
          </source>
          <destination type="file" exist="overwrite">
            <file>/reports/larmer.rpt</file>
          </destination>
        </item>
      </transferSet>
    </managedTransfer>
  </request>
```

Results

The transfer that is defined by the task XML is performed. The complete group of messages that are read from the START_QUEUE by AGENT_VENUS is written to a file called `/reports/larmer.rpt` on the system where AGENT_VIOLET is running.

What to do next

Transferring each message to a separate file

If you want to monitor a queue and have every message transferred to a separate file, you can use a similar technique to the one described previously in this topic.

1. Create the monitor as described previously, specifying the **-tr completeGroups** parameter on the **fteCreateMonitor** command.
2. In the task XML specify the following:
   ```
   <queue useGroups="true" groupId="${GROUPID}">START_QUEUE</queue>
   ```

However, when you put the messages onto the source queue, do not put them in a IBM WebSphere MQ group. Add IBM WebSphere MQ message properties to each message. For example, specify the `usr.filename` property with a unique file name value for each message. This effectively causes the WebSphere MQ Managed File Transfer agent to treat each message on the source queue as a separate group.

Example: Failing a message to file transfer using WebSphere MQ message properties

You can cause a message to file transfer to fail by setting the `usr.UserReturnCode` WebSphere MQ message property to a non-zero value. You can also specify supplementary information about the reason for the failure by setting the `usr.UserSupplement` WebSphere MQ message property.

About this task

In this example, a transfer is in progress between the queue INPUT_QUEUE and the file `/home/user/output.file`.

A user is creating messages and placing them on the queue INPUT_QUEUE. The source agent is consuming messages from the queue INPUT_QUEUE and is sending the transfer data to the destination agent. The destination agent is writing this data to the file `/home/user/output.file`.

The user writing messages to the queue INPUT_QUEUE wants to stop the transfer that is in progress and delete any data that has already been written to the destination file.

Procedure

1. The user writes a message to the queue INPUT_QUEUE that has the following WebSphere MQ message properties set:

   ```
   usr.UserReturnCode=1
   usr.UserSupplement="Cancelling transfer - sent wrong data."
   ```

2. The source agent reads the WebSphere MQ message properties and stops processing messages from the queue. The destination agent deletes any file data that has been written to the destination directory.

3. The source agent sends a transfer log message to the coordination queue manager reporting the transfer failure. The message contains the following information:

   ```
   <?xml version="1.0" encoding="UTF-8"?>
   <transaction version="1.00"
               ID="414d5120514d312020202020202020202020207e970d4920008702" agentRole="sourceAgent"
               xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xsi:noNamespaceSchemaLocation="TransferLog.xsd"
               xmlns="">
     <action time="2008-11-02T21:28:09.593Z">progress</action>
     <sourceAgent agent="FTEAGENT" QMgr="QM1">
       <systemInfo architecture="x86" name="Windows 7"
    version="6.1 build 7601 Service Pack 1" />
     </sourceAgent>
     <destinationAgent agent="FTEAGENT" QMgr="QM1">
       <systemInfo architecture="x86" name="Windows 7"
    version="6.1 build 7601 Service Pack 1" />
     </destinationAgent>
     <originator>
       <hostName>reportserver.com</hostName>
       <userID>USER1</userID>
       <mqmdUserID>USER1    </mqmdUserID>
     </originator>
     <transferSet index="0" size="1"
               startTime="2008-11-02T21:28:09.281Z"
               total="1">
       <item mode="binary">
         <source>
           <queue>INPUT_QUEUE@QM1</queue>
         </source>
         <destination exist="error">
           <file>/home/user/output.file</file>
         </destination>
         <status resultCode="1">
           <supplement>Cancelling transfer - sent wrong data.</supplement>
         </status>
       </item>
     </transferSet>
   </transaction>
   ```

Listing WebSphere MQ Managed File Transfer agents

You can list the agents registered with a particular queue manager using the command line or the WebSphere MQ Explorer.

About this task

To list agents using the command line, see fteListAgents command.

To list agents using the WebSphere MQ Explorer, in the Navigator view click **Agents** under the coordination queue manager name.

If an agent is not listed by the `fteListAgents` command or is not displayed in the WebSphere MQ Explorer, use the diagnosis flowchart in the following topic to locate and fix the problem: If your agent is

not listed by the `fteListAgents` command.

Stopping a WebSphere MQ Managed File Transfer agent

You can stop an agent from the command line. When you stop an agent, you are quiescing the agent and allowing the agent to complete its current file transfer before stopping. You can also specify the **-i** parameter at the command line to stop an agent immediately. When the agent has stopped, you cannot use that agent to transfer files until you restart it.

Before you begin

If you want to check the names of the agents associated with a queue manager, you can list agents by using the WebSphere MQ Explorer or the command line, see fteListAgents command.

About this task

To stop an agent from the command line, see fteStopAgent.

If you have configured your agent to run as a Windows service, running the **fteStopAgent** command also stops the Windows service. Alternatively, you can stop the agent by stopping the service by using the Windows Services tool. For more information, see the topic "Starting an agent as a Windows service" on page 201.

Related tasks:

"Starting a WebSphere MQ Managed File Transfer agent" on page 201
Before you can use a WebSphere MQ Managed File Transfer agent for a file transfer, you must first start the agent.

The protocol bridge

The protocol bridge enables your WebSphere MQ Managed File Transfer (MQMFT) network to access files stored on a file server outside your MQMFT network, either in your local domain or a remote location. This file server can use the FTP, FTPS, or SFTP network protocols. Each file server needs at least one dedicated agent. The dedicated agent is known as the protocol bridge agent. A bridge agent can interact with multiple file servers.

The protocol bridge is available as part of the Service component of WebSphere MQ Managed File Transfer. You can have multiple dedicated agents on a single system running MQMFT that connect to different file servers.

You can use a protocol bridge agent to transfer files to multiple endpoints simultaneously. MQMFT provides a file called `ProtocolBridgeProperties.xml` that you can edit to define the different protocol file servers that you want to transfer files to. The **fteCreateBridgeAgent** command adds the details of the default protocol file server to `ProtocolBridgeProperties.xml` for you. This file is described in "Protocol bridge properties file format" on page 571.

You can use the protocol bridge agent to perform the following actions:
- Upload files from the MQMFT network to a remote server using FTP, FTPS, or SFTP.
- Download files from a remote server, using FTP, FTPS, or SFTP, to the MQMFT network

Note: The protocol bridge agent can support only FTP, FTPS, or SFTP servers that allow files to be accessed by their absolute file path. If a relative file path is specified in a transfer request, the protocol bridge agent will attempt to convert the relative path into an absolute file path based on the home directory used to login to the protocol server. Those protocol servers that allow access to files based only on the current directory are not supported by the protocol bridge agent.
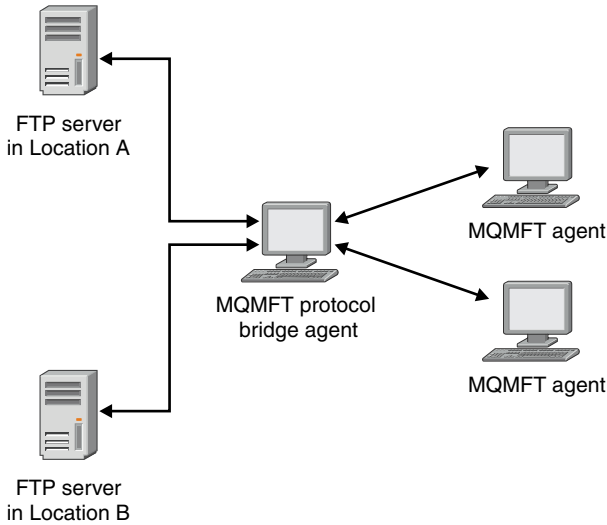
The diagram shows two FTP servers, at different locations. The FTP servers are being used to exchange files with the WebSphere MQ Managed File Transfer agents. The protocol bridge agent is between the FTP servers and the rest of the MQMFT network, and is configured to communicate with both FTP servers.

Ensure that you have another agent in your MQMFT network in addition to the protocol bridge agent. The protocol bridge agent is a bridge to the FTP, FTPS, or SFTP server only and does not write transferred files to the local disk. If you want to transfer files to or from the FTP, FTPS, or SFTP server you must use the protocol bridge agent as the destination or source for the file transfer (representing the FTP, FTPS, or SFTP server) and another standard agent as the corresponding source or destination.

When you transfer files using the protocol bridge, the bridge must have permission to read the source or destination directory containing the files you want to transfer. For example, if you want to transfer files from the directory /home/fte/bridge that has execute permissions (d--x--x--x) only, any transfers you attempt from this directory fail with the following error message:

```
BFGBR0032E: Attempt to read filename from the protocol file server
  has failed with server error 550. Failed to open file.
```

Configuring a protocol bridge agent

A protocol bridge agent is like a standard MQMFT agent. Create a protocol bridge agent by using the **fteCreateBridgeAgent** command. You can configure a protocol bridge agent using the ProtocolBridgeProperties.xml file, which is described in "Protocol bridge properties file format" on page 571. If you are using an earlier version, configure the agent using the specific protocol bridge properties described in Advanced agent properties. For all versions, you can also configure a credential mapping as described in "Mapping credentials for a file server" on page 257. After you have configured a protocol bridge agent for a particular protocol file server, you can then use that agent for that purpose only.

Protocol bridge recovery

If the protocol bridge agent is unable to connect to the file server because the file server is unavailable, all file transfer requests are queued until the file server becomes available. If the protocol bridge agent is unable to connect to the file server because the agent is using the wrong credentials, the transfer fails and the transfer log message reflects this error. If the protocol bridge agent is ended for any reason, all requested file transfers are retained and continue when the protocol bridge is restarted.

During file transfer, files are typically written as temporary files at the destination and are then renamed when the transfer is complete. However, if the transfer destination is a protocol file server that is configured as limited write (users can upload files to the protocol file server but cannot change those uploaded files in any way; effectively users can write once only), transferred files are written to the destination directly. This means that if a problem occurs during the transfer, the partially written files remain on the destination protocol file server and WebSphere MQ Managed File Transfer cannot delete or edit these files. In this situation, the transfer fails.

Upgrading a protocol bridge agent to use the V7.0.4.1 function

You can upgrade an existing protocol bridge agent that was created in V7.0.4 or earlier to use the new V7.0.4.1 function. This upgrade allows the agent to support multiple endpoints.

About this task

To upgrade a protocol bridge agent, ensure you have enabled the new function for V7.0.4.1 and complete the following steps:

Procedure

1. Generate a `ProtocolBridgeProperties.xml` file from the `ProtocolBridgeProperties.xsd` file in the `MQ_INSTALLATION_PATH`/mqft/samples/schema directory.

   For more information about the content of the `ProtocolBridgeProperties.xml` file, see "Defining properties for protocol file servers using the `ProtocolBridgeProperties.xml` file" on page 252.

2. Put the generated `ProtocolBridgeProperties.xml` file into the *configuration_directory*/*coordination_queue_manager*/agents/*agent_name* directory.

3. Copy all protocol bridge properties for the agent you want to migrate from the `agent.properties` file to the `ProtocolBridgeProperties.xml` file. The properties that relate to the protocol bridge in the `agent.properties` file start with `protocol`.

   You are recommended to delete the protocol bridge properties from the `agent.properties` file after you have successfully upgraded the protocol bridge agent. However, if you have a default protocol file server, consider keeping the protocol bridge properties in the `agent.properties` file in step with the values in the `ProtocolBridgeProperties.xml` file. This means that if you choose to back out the V7.0.4.1 function, the values in the `agent.properties` file are not outdated.

4. Update the `ProtocolBridgeCredentials.xml` file so that it contains the required <server> elements for your protocol file servers. For more information about this file, see "Protocol bridge credentials file format" on page 567.

   Ensure each protocol file server name meets the naming restrictions, that is, the name must be at least 2 characters in length, is not case sensitive, and is limited to alphanumeric characters and the following characters:
   - period (.)
   - underscore (_)
   - forward slash (/)
   - percent sign (%)

"The protocol bridge" on page 248
The protocol bridge enables your WebSphere MQ Managed File Transfer (MQMFT) network to access
files stored on a file server outside your MQMFT network, either in your local domain or a remote
location. This file server can use the FTP, FTPS, or SFTP network protocols. Each file server needs at least
one dedicated agent. The dedicated agent is known as the protocol bridge agent. A bridge agent can
interact with multiple file servers.

Related tasks:

"Backing out the V7.0.4.1 function on a protocol bridge agent"
If you have upgraded a protocol bridge agent from V7.0.4 or earlier to the new V7.0.4.1 function and
want to back out the upgrade, you can return the agent to its original level of function. This means the
agent returns to being able to transfer to a single endpoint only.

"Defining properties for protocol file servers using the `ProtocolBridgeProperties.xml` file" on page 252
Define the properties of one or more protocol file servers that you want to transfer files to and from
using the `ProtocolBridgeProperties.xml` file, which is provided by WebSphere MQ Managed File
Transfer in the agent configuration directory.

Backing out the V7.0.4.1 function on a protocol bridge agent

If you have upgraded a protocol bridge agent from V7.0.4 or earlier to the new V7.0.4.1 function and
want to back out the upgrade, you can return the agent to its original level of function. This means the
agent returns to being able to transfer to a single endpoint only.

About this task

To back out the V7.0.4.1 upgrade, complete the following steps:

Procedure

1. Move the protocol bridge properties from the `ProtocolBridgeProperties.xml` file for the agent you
   want to migrate to the `agent.properties` file. The agent properties file is located at
   configuration_directory/coordination_queue_manager/agents/*bridge_agent*/`agent.properties`. The
   protocol bridge properties are described in Protocol bridge properties.

2. Delete the `ProtocolBridgeProperties.xml` file from the *configuration_directory/coordination_qmgr*/
   agents/*agent_name* directory.

   The properties in the `ProtocolBridgeProperties.xml` file always take precedence over the properties
   in the `agent.properties` file, so deleting `ProtocolBridgeProperties.xml` ensures this file is not used if
   you keep the V7.0.4.1 function enabled.

3. Update the `ProtocolBridgeCredentials.xml` file so that it has the required <serverHost> elements for
   your protocol file servers. For more information about this file, see "Protocol bridge credentials file
   format" on page 567.

"The protocol bridge" on page 248
The protocol bridge enables your WebSphere MQ Managed File Transfer (MQMFT) network to access
files stored on a file server outside your MQMFT network, either in your local domain or a remote
location. This file server can use the FTP, FTPS, or SFTP network protocols. Each file server needs at least
one dedicated agent. The dedicated agent is known as the protocol bridge agent. A bridge agent can
interact with multiple file servers.

Related tasks:

"Upgrading a protocol bridge agent to use the V7.0.4.1 function" on page 250
You can upgrade an existing protocol bridge agent that was created in V7.0.4 or earlier to use the new
V7.0.4.1 function. This upgrade allows the agent to support multiple endpoints.

"Defining properties for protocol file servers using the `ProtocolBridgeProperties.xml` file" on page 252
Define the properties of one or more protocol file servers that you want to transfer files to and from
using the `ProtocolBridgeProperties.xml` file, which is provided by WebSphere MQ Managed File

Transfer in the agent configuration directory.

Defining properties for protocol file servers using the `ProtocolBridgeProperties.xml` file

Define the properties of one or more protocol file servers that you want to transfer files to and from using the `ProtocolBridgeProperties.xml` file, which is provided by WebSphere MQ Managed File Transfer in the agent configuration directory.

About this task

The **fteCreateBridgeAgent** command creates the `ProtocolBridgeProperties.xml` file in the agent configuration directory *MQ_DATA_PATH*/mqft/config/*coordination_queue_manager*/agents/ *bridge_agent_name*. The command also creates an entry in the file for the default protocol file server, if a default was specified when the command was run.

If you want to add further non-default protocol servers, edit this file to define their properties. This example adds an additional FTP server.

Procedure

1. Define a protocol file server by inserting the following lines into the file as a child element of `<tns:serverProperties>`:

```
<tns:ftpServer name="myserver" host="myhost.hursley.ibm.com" port="1234"
    platform="windows"
                timeZone="Europe/London" locale="en-GB" fileEncoding="UTF-8"
              listFormat="unix" limitedWrite="false" >
<tns:limits maxListFileNames="10" maxListDirectoryLevels="500"/>
```

2. Then change the value of the attributes:
   - `name` is the name of your protocol file server
   - `host` is the host name or IP address of the protocol file server
   - `port` is the port number of the protocol file server
   - `platform` is the platform that the protocol file server runs on
   - `timeZone` is the time zone that the protocol file server runs in
   - `locale` is the language used on the protocol file server
   - `fileEncoding` is the character encoding of the protocol file server
   - `listFormat` is the file listing format returned from the protocol file server
   - `limitedWrite` determines whether to follow the default mode when writing to a file server, which is to create a temporary file and then rename that file when the transfer has completed. For a file server that is configured as write only, the file is created directly with its final name. The value of this property can be true or false. The `limitedWrite` attribute and the `doNotUseTempOutputFile` agent property are used together in the case of protocol bridge agents. If you want to use temporary files, then you must not set the value of `doNotUseTempOutputFile`, and you must set the value of `limitedWrite` to false. Another other combination of settings means that temporary files will not be used.
   - `maxListFileNames` is the maximum number of names collected when scanning a directory on the protocol file server for file names.
   - `maxListDirectoryLevels` is the maximum number of directory levels to recurse when scanning a directory on the protocol file server for file names.

   For more details about these attributes, including whether they are required or optional and their default values, see "Protocol bridge properties file format" on page 571.

Related reference:

"Protocol bridge properties file format" on page 571
The `ProtocolBridgeProperties.xml` file in the agent configuration directory defines properties for protocol file servers.

"Regular expressions used by WebSphere MQ Managed File Transfer" on page 685
WebSphere MQ Managed File Transfer uses regular expressions in a number of scenarios. For example, regular expressions are used to match user IDs for Connect:Direct security credentials, or to split a file into multiple messages by creating a new message each time a regular expression is matched. The regular expression syntax used by WebSphere MQ Managed File Transfer is the syntax supported by the `java.util.regex` API. This regular expression syntax is similar to, but not the same as, the regular expression syntax used by the Perl language.

Looking up protocol file server properties by using exit classes (ProtocolBridgePropertiesExit2)

If you have a large number of protocol file servers, you can implement the `com.ibm.wmqfte.exitroutine.api.ProtocolBridgePropertiesExit2` interface to look up protocol file server properties that are referenced in transfers. You can implement this interface in preference to maintaining a `ProtocolBridgeProperties.xml` file. WebSphere MQ Managed File Transfer provides a sample user exit that looks up protocol file server properties.

Configuring user exits that look up protocol bridge properties
About this task

Any user exit that looks up protocol bridge properties must implement the interface `com.ibm.wmqfte.exitroutine.api.ProtocolBridgePropertiesExit2`. For more information, see "ProtocolBridgePropertiesExit2.java interface" on page 961.

You can chain multiple protocol server properties exits together in a similar manner to other user exits. The exits are called in the order that they are specified in using the `protocolBridgePropertiesExitClasses` property in the agent properties file. The initialize methods all return separately and if one or more returns a value of false, the agent does not start. The error is reported in the agent event log.

Only one overall result is returned for the getProtocolServerProperties methods of all of the exits. If the method returns a properties object as the result code, this value is the returned result and the getProtocolServerProperties methods of the subsequent exits are not called. If the method returns a value of null as the result code, the getProtocolServerProperties method of the next exit is called. If there is no subsequent exit, the null result is returned. An overall result code of null is considered as a lookup failure by the protocol bridge agent.

You are recommended to use the ProtocolBridgePropertiesExit2.java interface, but for information about the ProtocolBridgePropertiesExit.java interface, see "Looking up protocol file server properties by using exit classes (ProtocolBridgePropertiesExit.java)" on page 255.

To run your exit, complete the following steps:

Procedure
1. Compile the protocol server properties user exit.
2. Create a Java archive (JAR) file containing the compiled exit and its package structure.
3. Put the JAR file containing the exit class in the `exits` directory of the protocol bridge agent . This directory is found in the *MQ_DATA_PATH*/mqft/config/*coordination_queue_manager*/agents/*bridge_agent_name* directory.
4. Edit the property file of the protocol bridge agent to include the property `protocolBridgePropertiesExitClasses`. For the value of this property, specify a comma-separated list

of classes that implement a protocol bridge server properties user exit. The exit classes are called in the order that they are specified in this list. For more information, see "The agent.properties file" on page 548.

5. You can optionally specify the `protocolBridgePropertiesConfiguration` property. The value you specify for this property is passed in as a String to the initialize() method of the exit classes specified by `protocolBridgePropertiesExitClasses`. For more information, see "The agent.properties file" on page 548.

Using the sample user exit
About this task

A sample user exit that looks up protocol bridge properties is provided in the *MQ_INSTALLATION_PATH*/`mqft/samples/protocolBridge` directory and in the topic "Sample protocol bridge properties user exit" on page 339.

The `SamplePropertiesExit2.java` exit reads a properties file that contains properties for protocol servers. The format of each entry in the properties file is as follows:

serverName=type`://host:port`

The location of the properties file is taken from the protocol bridge agent property `protocolBridgePropertiesConfiguration`.

To run the sample user exit, complete the following steps:

Procedure
1. Compile the `SamplePropertiesExit2.java` file.
2. Create a JAR file containing the compiled exit and its package structure.
3. Put the JAR file in the *MQ_DATA_PATH*/`mqft/config/`*coordination_queue_manager*/`agents/`*bridge_agent*/`exits` directory.
4. Edit the *MQ_DATA_PATH*/`mqft/config/`*coordination_queue_manager*/`agents/`*bridge_agent_name*/`agent.properties` file to contain the line:

   `protocolBridgePropertiesExitClasses=SamplePropertiesExit2`
5. Create a protocol bridge properties file, for example `protocol_bridge_properties.properties`, in the directory *MQ_DATA_PATH*/`mqft/config/`*coordination_queue_manager*/`agents/`*bridge_agent*. Edit this file to include entries in the format:

   serverName=type`://host:port`
6. Edit the *MQ_DATA_PATH*/`mqft/config/`*coordination_queue_manager*/`agents/`*bridge_agent*/`agent.properties` file to contain the line:

   `protocolBridgePropertiesConfiguration=`*MQ_DATA_PATH*/`mqft/config/`*coordination_queue_manager*/`agents/`*bridge_agent*/`protocol_bridge_properties.properties`

   You must use the absolute path to the `protocol_bridge_properties.properties` file.
7. Start the protocol bridge agent by using the **fteStartAgent** command.

Related concepts:

"The protocol bridge" on page 248
The protocol bridge enables your WebSphere MQ Managed File Transfer (MQMFT) network to access files stored on a file server outside your MQMFT network, either in your local domain or a remote location. This file server can use the FTP, FTPS, or SFTP network protocols. Each file server needs at least one dedicated agent. The dedicated agent is known as the protocol bridge agent. A bridge agent can interact with multiple file servers.

Related reference:

"ProtocolBridgePropertiesExit2.java interface" on page 961

"Sample protocol bridge properties user exit" on page 339

"The agent.properties file" on page 548
Each agent has its own properties file, `agent.properties`, that must contain the information that an agent uses to connect to its queue manager. The `agent.properties` file can also contain properties that alter the behavior of the agent.

"fteCreateBridgeAgent (create and configure WebSphere MQ Managed File Transfer protocol bridge agent)" on page 432
The **fteCreateBridgeAgent** command creates a protocol bridge agent and its associated configuration. Create a protocol bridge agent for each file server that you want to send files to and receive files from.

Looking up protocol file server properties by using exit classes (ProtocolBridgePropertiesExit.java)

If you have a large number of protocol file servers, you can implement the `com.ibm.wmqfte.exitroutine.api.ProtocolBridgePropertiesExit` interface to look up protocol file server properties that are referenced in transfers. You can implement this interface in preference to maintaining a `ProtocolBridgeProperties.xml` file. You are recommended to use the ProtocolBridgePropertiesExit2.java interface but the ProtocolBridgePropertiesExit.java interface is also supported. If you have an existing implementation of the ProtocolBridgePropertiesExit.java interface from WebSphere MQ File Transfer Edition, you can use it in WebSphere MQ V7.5. The new getCredentialLocation method in ProtocolBridgePropertiesExit2.java uses the default location of the `ProtocolBridgeCredentials.xml` file, which is your home directory.

Configuring user exits that look up protocol bridge properties

Any user exit that looks up protocol bridge properties must implement the interface `com.ibm.wmqfte.exitroutine.api.ProtocolBridgePropertiesExit`. For more information, see "ProtocolBridgePropertiesExit.java interface" on page 256.

You can chain multiple protocol server properties exits together in a similar manner to other user exits. The exits are called in the order that they are specified in using the `protocolBridgePropertiesExitClasses` property in the agent properties file. The initialize methods all return separately and if one or more returns a value of false, the agent does not start. The error is reported in the agent event log.

Only one overall result is returned for the getProtocolServerProperties methods of all of the exits. If the method returns a properties object as the result code, this value is the returned result and the getProtocolServerProperties methods of the subsequent exits are not called. If the method returns a value of null as the result code, the getProtocolServerProperties method of the next exit is called. If there is no subsequent exit, the null result is returned. An overall result code of null is considered as a lookup failure by the protocol bridge agent.

To run your exit, complete the following steps:

1. Compile the protocol server properties user exit.

2. Create a Java archive (JAR) file containing the compiled exit and its package structure.
3. Put the JAR file containing the exit class in the `exits` directory of the protocol bridge agent . This directory is found in the *MQ_DATA_PATH*/mqft/config/*coordination_queue_manager*/agents/ *bridge_agent_name* directory.
4. Edit the property file of the protocol bridge agent to include the property `protocolBridgePropertiesExitClasses`. For the value of this property, specify a comma-separated list of classes that implement a protocol bridge server properties user exit. The exit classes are called in the order that they are specified in this list. For more information, see "The agent.properties file" on page 548.
5. You can optionally specify the `protocolBridgePropertiesConfiguration` property. The value you specify for this property is passed in as a String to the initialize() method of the exit classes specified by `protocolBridgePropertiesExitClasses`. For more information, see "The agent.properties file" on page 548.

ProtocolBridgePropertiesExit.java interface

```
package com.ibm.wmqfte.exitroutine.api;

import java.util.Map;
import java.util.Properties;

/**
 * An interface that is implemented by classes that are to be invoked as part of
 * user exit routine processing. This interface defines methods that will be
 * invoked by a protocol bridge agent to look up properties for protocol servers
 * that are referenced in transfers.
 * <p>
 * There will be one instance of each implementation class for each protocol
 * bridge agent. The methods can be called from different threads so the methods
 * must be synchronised.
 */
public interface ProtocolBridgePropertiesExit {

  /**
   * Invoked once when a protocol bridge agent is started. It is intended to
   * initialize any resources that are required by the exit.
   *
   * @param bridgeProperties
   *            The values of properties defined for the protocol bridge.
   *            These values can only be read, they cannot be updated by the
   *            implementation.
   * @return {@code true} if the initialization is successful and {@code
   *         false} if unsuccessful. If {@code false} is returned from an exit
   *         the protocol bridge agent will not start.
   */
  public boolean initialize(final Map<String, String> bridgeProperties);

  /**
   * Obtains a set of properties for the specified protocol server name.
   * <p>
   * The returned {@link Properties} must contain entries with key names
   * corresponding to the constants defined in
   * {@link ProtocolServerPropertyConstants} and in particular must include an
   * entry for all appropriate constants described as required.
   *
   * @param protocolServerName
   *            The name of the protocol server whose properties are to be
   *            returned. If a null or a blank value is specified, properties
   *            for the default protocol server are to be returned.
   * @return The {@link Properties} for the specified protocol server, or null
   *         if the server cannot be found.
   */
  public Properties getProtocolServerProperties(
    final String protocolServerName);
```

```
/**
 * Invoked once when a protocol bridge agent is shut down. It is intended to
 * release any resources that were allocated by the exit.
 *
 * @param bridgeProperties
 *             The values of properties defined for the protocol bridge.
 *             These values can only be read, they cannot be updated by the
 *             implementation.
 */
public void shutdown(final Map<String, String> bridgeProperties);

}
```

Mapping credentials for a file server

Map user credentials in WebSphere MQ Managed File Transfer to user credentials on the file server by using the default credential mapping function of the protocol bridge agent or by writing your own user exit. WebSphere MQ Managed File Transfer provides a sample user exit that performs user credential mapping.

Mapping credentials for a file server using the `ProtocolBridgeCredentials.xml` file

Map user credentials in WebSphere MQ Managed File Transfer to user credentials on the file server by using the default credential mapping function of the protocol bridge agent. WebSphere MQ Managed File Transfer provides an XML file that you can edit to include your credential information.

About this task

The `ProtocolBridgeCredentials.xml` file must be manually created by the user. By default, the location of this file is the home directory of the user who started the protocol bridge agent, but this can be stored anywhere on the file system accessible by the agent. To specify a different location, add the <credentialsFile> element to the `ProtocolBridgeProperties.xml` file. For example,

```
<tns:credentialsFile path="/example/path/to/ProtocolBridgeCredentials.xml"/>
```

Before you can use a protocol bridge agent, set up credential mapping by editing this file to include host, user, and credential information. For more information and samples, see "Protocol bridge credentials file format" on page 567.

Procedure

1.
   - Edit the line `<tns:server name="`*server name*`">` to change the value of the name attribute to the server name in the `ProtocolBridgeProperties.xml` file.

     Protocol bridge agents that are created for Version 7.0.4 and earlier do not have a `ProtocolBridgeProperties.xml` file (or related user exits), so for Version 7.0.4.1 and later the server name is automatically assigned the server's host name. Therefore, if you use an updated `ProtocolBridgeCredentials.xml` file with <server> entries, a name corresponding to the server's host name will match.

     You can use the pattern attribute to specify that you used a server name that contains wildcards or regular expressions. For example,

     ```
     <tns:server name="serverA*" pattern="wildcard">
     ```

2. Insert user ID and credential information into the file as child elements of `<tns:server>`. You can insert one or many of the following elements into the file:
   - If the protocol file server is an FTP, FTPS, or SFTP server, you can use passwords to authenticate the user requesting the transfer. Insert the following lines into the file:

```
<tns:user name="FTE User ID"
  serverUserId="Server User ID"
  serverPassword="Server Password">
</tns:user>
```

Then change the value of the attributes.

– `name` is a Java regular expression to match the MQMD user ID associated with the MQMFT transfer request

– `serverUserId` is the value that is passed to the protocol file server as the login user ID. If the `serverUserId` attribute is not specified, the MQMD user ID associated with the MQMFT transfer request is used instead

– `serverPassword` is the password that is associated with the `serverUserId`.

The `name` attribute can contain a Java regular expression. The credential mapper attempts to match the MQMD user ID of the MQMFT transfer request to this regular expression. The protocol bridge agent attempts to match the MQMD user ID to the regular expression in the name attribute of the <tns:user> elements in the order that the elements exist in the file. When a match is found the protocol bridge agent does not look for more matches. If a match is found, the corresponding `serverUserId` and `serverPassword` values are passed to the protocol file server as the login user ID and password. The MQMD user ID matches are case-sensitive.

• If the protocol file server is an SFTP server, you can use public and private keys to authenticate the user requesting the transfer. Insert the following lines into the file and change the value of the attributes. The <tns:user> element can contain one or many <tns:privateKey> elements.

```
<tns:user name="FTE User ID"
    serverUserId="Server User ID"
    hostKey="Host Key">
    <tns:privateKey associationName="association"
      keyPassword="Private key password">
  Private key file text     </tns:privateKey>
</tns:user>
```

– `name` is a Java regular expression to match the MQMD user ID associated with the MQMFT transfer request

– `serverUserId` is the value that is passed to the protocol file server as the login user ID. If the `serverUserId` attribute is not specified, the MQMD user ID associated with the MQMFT transfer request is used instead

– `hostKey` is the expected key that is returned from the server when logging on

– `key` is the private key of the `serverUserId`

– `keyPassword` is the password for the key to generate public keys

– `associationName` is a value that is used to identify for trace and logging purposes

The `name` attribute can contain a Java regular expression. The credential mapper attempts to match the MQMD user ID of the MQMFT transfer request to this regular expression. The protocol bridge agent attempts to match the MQMD user ID to the regular expression in the name attribute of the <tns:user> elements in the order that the elements exist in the file. When a match is found the protocol bridge agent does not look for more matches. If a match is found, the corresponding `serverUserId` and `key` values are used to authenticate the MQMFT user with the protocol file server. The MQMD user ID matches are case-sensitive.

For more information about using private keys with a protocol bridge agent, see "Example: How to configure a protocol bridge agent to use private key credentials with a UNIX SFTP server" on page 260.

Mapping credentials for a file server using exit classes

If you do not want to use the default credential mapping function of the protocol bridge agent, you can map user credentials in WebSphere MQ Managed File Transfer to user credentials on the file server by writing your own user exit. WebSphere MQ Managed File Transfer provides a sample user exit that performs user credential mapping. If you configure credential mapping user exits, they take the place of the default credential mapping function.

Configuring protocol bridge credential user exits:
About this task

A user exit for mapping protocol bridge credentials must implement one of the following interfaces:
- `com.ibm.wmqfte.exitroutine.api.ProtocolBridgeCredentialExit`, which allows a protocol bridge agent to transfer files to and from one default protocol file server
- `com.ibm.wmqfte.exitroutine.api.ProtocolBridgeCredentialExit2`, which allows you to transfer files to and from multiple endpoints.

The `com.ibm.wmqfte.exitroutine.api.ProtocolBridgeCredentialExit2` interface contains the same function as `com.ibm.wmqfte.exitroutine.api.ProtocolBridgeCredentialExit` and also includes extended function. For more information, see "ProtocolBridgeCredentialExit.java interface" on page 959 and "ProtocolBridgeCredentialExit2.java interface" on page 960.

The credential exits can be chained together in a similar manner to other user exits. The exits are called in the order that they are specified in using the `protocolBridgeCredentialConfiguration` property in the agent properties file. The initialize methods all return separately and if one or more returns a value of false, the agent does not start. The error is reported in the agent event log.

Only one overall result is returned for the mapMQUserId methods of all of the exits as follows:
- If the method returns a value of USER_SUCCESSFULLY_MAPPED or USER_DENIED_ACCESS as the result code, this value is the returned result and the mapMQUserId methods of the subsequent exits are not called.
- If the method returns a value of NO_MAPPING_FOUND as the result code, the mqMQUserId method of the next exit is called.
- If there is no subsequent exit, the NO_MAPPING_FOUND result is returned.
- An overall result code of USER_DENIED_ACCESS or NO_MAPPING_FOUND is considered as a transfer failure by the bridge agent.

To run your exit, complete the following steps:

Procedure
1. Compile the protocol bridge credential user exit.
2. Create a Java archive (JAR) file containing the compiled exit and its package structure.
3. Place the JAR file containing the exit class in the `exits` directory of the bridge agent. This directory is in the *MQ_DATA_PATH*/mqft/config/*coordination_queue_manager*/agents/*bridge_agent_name* directory.
4. Edit the property file of the protocol bridge agent to include the property `protocolBridgeCredentialExitClasses`. For the value of this property, specify a comma-separated list of classes that implement a protocol bridge credential exit routine. The exit classes are called in the order that they are specified in this list. For more information, see "The agent.properties file" on page 548.
5. You can optionally specify the `protocolBridgeCredentialConfiguration` property. The value you specify for this property is passed in as a String object to the initialize() method of the exit classes specified by `protocolBridgeCredentialExitClasses`. For more information, see "The agent.properties file" on page 548.
6. Start the protocol bridge agent with the **fteStartAgent** command.

Using the sample user exit:
About this task

A sample protocol bridge credential exit is provided in the *MQ_INSTALLATION_PATH*/mqft/samples/
protocolBridge directory and in the topic "Sample protocol bridge credential user exit" on page 337. This
sample is based on the com.ibm.wmqfte.exitroutine.api.ProtocolBridgeCredentialExit interface.

The SampleCredentialExit.java exit reads a properties file that maps the MQMD user IDs associated
with transfer requests to server user IDs and server passwords. The location of the properties file is taken
from the protocol bridge agent property protocolBridgeCredentialConfiguration.

To run the sample user exit, complete the following steps:

Procedure
1. Compile the SampleCredentialExit.java file.
2. Create a JAR file containing the compiled exit and its package structure.
3. Place the JAR file in the *MQ_DATA_PATH*/mqft/config/*coordination_queue_manager*/agents/
   bridge_agent_name/exits directory.
4. Edit the *MQ_DATA_PATH*/mqft/config/*coordination_queue_manager*/agents/*bridge_agent_name*/
   agent.properties file to contain the line:

   protocolBridgeCredentialExitClasses=SampleCredentialExit
5. Create a credential properties file (credentials.properties) in the directory *MQ_DATA_PATH*/mqft/
   config/*coordination_queue_manager*/agents/*bridge_agent* and edit it to include entries in the format:

   mqUserId=*serverUserId*,*serverPassword*
6. Edit the *MQ_DATA_PATH*/mqft/config/*coordination_queue_manager*/agents/*bridge_agent_name*/
   agent.properties file to contain the line:

   protocolBridgeCredentialConfiguration=*MQ_DATA_PATH*/mqft/
   config/*coordination_queue_manager*/agents/*bridge_agent_name*/credentials.properties

   You must use the absolute path to the credentials.properties file.
7. Start the protocol bridge agent by using the **fteStartAgent** command.

Example: How to configure a protocol bridge agent to use private key credentials with a UNIX SFTP server

This example demonstrates how you can generate and configure the ProtocolBridgeCredentials.xml file.
This example is a typical example and the details might vary according to your platform, but the
principles remain the same.

About this task

Procedure
1. On the SFTP client, log on with the login ID to be passed to the SFTP server by the protocol bridge
   agent and run the **ssh-keygen** command to create a public/private key sequence. Supply a pass
   phrase when asked for one. The **ssh-keygen** command generates the following two files: id_rsa and
   id_rsa.pub. If you need DSA format, use **-t dsa** when you run the **ssh-keygen** command
2. Copy the contents of the id_rsa.pub file into the ~/.ssh/authorized_keys file of the SFTP user on the
   SFTP server. Ensure that the SFTP file server process has read access to this file.
3. Run the following command to obtain the host ssh fingerprint of the SFTP server: ssh-keygen -l -f
   /etc/ssh/ssh_host_rsa_key.pub
4. On the protocol bridge agent system, edit the ProtocolBridgeCredentials.xml file. Substitute the
   values shown in italics in the following example with your own values:

```
<tns:credentials xmlns:tns="http://wmqfte.ibm.com/ProtocolBridgeCredentials"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://wmqfte.ibm.com/ProtocolBridgeCredentials
 ProtocolBridgeCredentials.xsd ">

<tns:agent name="Agent_name">

<tns:server name="SFTP_name">

<tns:user name="mq_User_ID" serverUserId="SFTP_user_ID"
  hostKey="ssh_host_finger">
<tns:privateKey associationName="name" keyPassword="pass_phrase">
 Complete contents of the id_rsa file including the entries
 -----BEGIN RSA PRIVATE KEY-----

 -----END RSA PRIVATE KEY-----
</tns:privateKey>
</tns:user>

</tns:server>
</tns:agent>
</tns:credentials>
```

where:
- *Agent_name* is the name of the protocol bridge agent.
- *SFTP_host_name* is the name of the SFTP server as shown in the `ProtocolBridgeProperties.xml` file.
- *mq_User_ID* is the MQMD user ID associated with the transfer request.
- *SFTP_user_ID* is the SFTP user ID as used in step 2. It is the value passed to the SFTP serve as the login user ID.
- *ssh_host_finger* is the fingerprint collected in step 3.
- *name* is a name that you can specify to be used for trace and logging purposes.
- *pass_phrase* is the pass phrase you provided in the ssh-keygen in step 1.
- *Complete contents of the id_rsa file* is the complete contents of the generated `id_rsa` file from step 1. To prevent a connection error, ensure that you include both of the following entries:

      -----BEGIN RSA PRIVATE KEY-----

      -----END RSA PRIVATE KEY-----

  You can add additional keys by duplicating the <tns:privatekey> element.
5. Start the protocol bridge agent if the agent is not already started. Alternatively, the protocol bridge agent periodically polls the `ProtocolBridgeCredentials.xml` file and pick up the changes.

Configuring a protocol bridge for an FTPS server

Configure an FTPS server in a similar way as you configure an FTP server: create a bridge agent for the server, define the server properties, and map user credentials.

About this task

To configure an FTPS server, complete the following steps:

Procedure

1. Create a protocol bridge agent for the FTPS server by using the **fteCreateBridgeAgent** command. The parameters that are applicable to FTP are also applicable to FTPS but there are also three required parameters specific to FTPS:

   a. The **-bt** parameter. Specify FTPS as the value of this parameter.

b. The **-bts** parameter for the truststore file. The command assumes that only server authentication is required and you must specify the location of the truststore file.

   The explicit form of the FTPS protocol is configured by the **fteCreateBridgeAgent** command by default but you can configure the implicit form by changing the protocol bridge properties file. The protocol bridge always connects to FTPS servers in passive mode.

   For more information about the **fteCreateBridgeAgent** command, see "fteCreateBridgeAgent (create and configure WebSphere MQ Managed File Transfer protocol bridge agent)" on page 432.

   If you need instructions about how to create truststore files, see the developerWorks article, Configuring Secure Sockets Layer connectivity in WebSphere MQ File Transfer Edition, or see the information about the keytool at the Oracle keytool documentation.

2. Define the FTPS server properties within an <ftpsServer> element in the protocol bridge properties file: ProtocolBridgeProperties.xml. For more information, see "Defining properties for protocol file servers using the ProtocolBridgeProperties.xml file" on page 252. You can also enable client authentication by editing the protocol bridge properties file. For details of all the configuration options, see "Protocol bridge properties file format" on page 571.

3. Map user credentials in WebSphere MQ Managed File Transfer to user credentials on the FTPS server either by using the default credential mapping function of the protocol bridge agent or by writing your own user exit. For more information, see "Mapping credentials for a file server" on page 257.

4. By default, the truststore file is configured as having the JKS format; if you want to change the format, edit the protocol bridge properties file.

5. If you need FIPS support, set the agent property agentSslFipsRequired property to true in the `agent.properties` file. For more information, see SSL properties.

Example

An example entry for an FTPS server in the protocol bridge properties file is shown as follows:

```
<tns:serverProperties xmlns:tns="http://wmqfte.ibm.com/ProtocolBridgeProperties"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://wmqfte.ibm.com/ProtocolBridgeProperties
      ProtocolBridgeProperties.xsd">
   <tns:defaultServer name="ftpsserver.mycompany.com" />

   <tns:ftpsServer name="ftpsserver.mycompany.com" host="ftpsserver.mycompany.com" port="990" platform="windows"
      timeZone="Europe/London" locale="en_US" fileEncoding="UTF8"
      listFormat="unix" limitedWrite="false"
      trustStore="c:\mydirec\truststore.jks" />

   <!-- Define servers here -->
</tns:serverProperties>
```

What to do next

For information about the parts of the FTPS protocol that are supported and, which are not supported, see "FTPS server support by the protocol bridge" on page 690.

The Connect:Direct bridge

You can transfer files to and from an existing IBM Sterling Connect:Direct network. Use the Connect:Direct bridge, which is a component of WebSphere MQ Managed File Transfer, to transfer files between MQMFT and IBM Sterling Connect:Direct.

**B2B IT Department
(Connect:Direct)**   **Application Integration
IT department (MQMFT)**

Connect:Direct
node

**Connect:Direct
node**   **MQMFT
Connect:Direct
bridge agent**   MQMFT agent

Connect:Direct
node   The Connect:Direct bridge   MQMFT agent
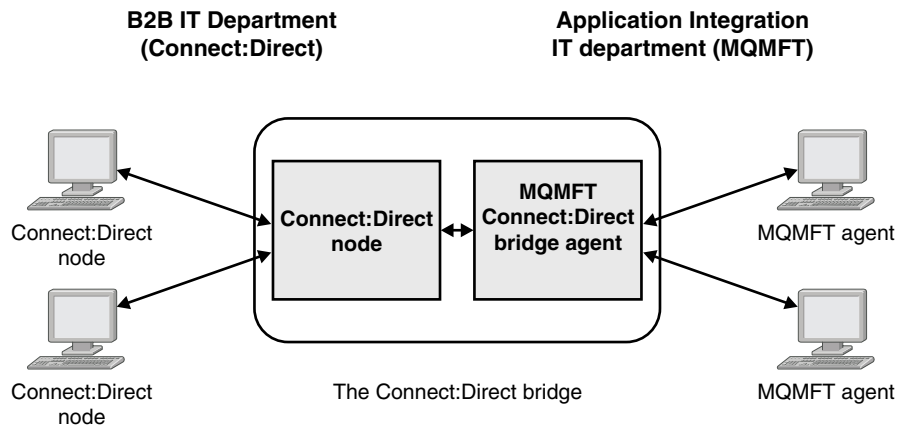
The diagram shows an MQMFT Connect:Direct bridge between two departments, the B2B IT department and the Application Integration IT department. The B2B IT department uses Connect:Direct to transfer files to and from the company's business partners. The Application Integration IT department uses WebSphere MQ as its messaging infrastructure and so has recently chosen WebSphere MQ Managed File Transfer as its file transfer solution.

Using the MQMFT Connect:Direct bridge, the two departments can transfer files between the Connect:Direct network in the B2B IT department and the MQMFT network in the Application Integration IT department. The Connect:Direct bridge is a component of WebSphere MQ Managed File Transfer, which includes an MQMFT agent that communicates with a Connect:Direct node. The MQMFT agent is dedicated to transfers with the Connect:Direct node, and is known as the Connect:Direct bridge agent.

The Connect:Direct bridge is available as part of the Service and Agent components of WebSphere MQ Managed File Transfer, and can be used for the following tasks:

1. Use Managed File Transfer commands to initiate a transfer of a file, or multiple files, from an MQMFT agent to a Connect:Direct node.
2. Use Managed File Transfer commands to initiate a transfer of a file, or multiple files, from a Connect:Direct node to an MQMFT agent.
3. Use Managed File Transfer commands to initiate a file transfer that starts a user-defined Connect:Direct process.
4. Use Connect:Direct process to submit an MQMFT file transfer request.

A Connect:Direct bridge can transfer files to or from only Connect:Direct nodes. The Connect:Direct bridge can transfer files to or from its local file system only as part of a transfer submitted by a Connect:Direct process.

Supported platforms

The Connect:Direct bridge is made up of an MQMFT Connect:Direct bridge agent and a Connect:Direct node. The agent is supported on Windows and Linux for System x. The node is supported on the platforms that are supported for IBM Sterling Connect:Direct for Windows and IBM Sterling

Connect:Direct for UNIX. For instructions on creating the Connect:Direct bridge agent and configuring a Connect:Direct node for the agent to communicate with, see "Configuring the Connect:Direct bridge" on page 190.

The Connect:Direct bridge can transfer files to and from Connect:Direct nodes that are running as part of a Connect:Direct for Windows or Connect:Direct for UNIX Service installation. For details of the versions of Connect:Direct that are supported, see the web page WebSphere MQ System Requirements.

The agent and node that make up the Connect:Direct bridge must be on the same system, or have access to the same file system, for example through a shared NFS mount. This file system is used to temporarily store files during file transfers that involve the Connect:Direct bridge, in a directory defined by the **cdTmpDir** parameter. The Connect:Direct bridge agent and the Connect:Direct bridge node must be able to address this directory using the same path name. For example, if the agent and node are on separate Windows systems, the systems must use the same drive letter to mount the shared file system. The following configurations allow the agent and the node to use the same path name:

- The agent and node are on the same system, which is either running Windows or Linux for System x
- The agent is on Linux for System x, and the node is on UNIX
- The agent is on one Windows system, and the node is on another Windows system

The following configurations do not allow the agent and the node to use the same path name:

- The agent is on Linux for System x, and the node is on Windows
- The agent is on Windows, and the node is on UNIX

Consider this restriction when planning your installation of the Connect:Direct bridge.

Transferring a file to a Connect:Direct node

You can transfer a file from a WebSphere MQ Managed File Transfer agent to a Connect:Direct node using the Connect:Direct bridge. Specify a Connect:Direct node as the destination of the transfer by specifying the Connect:Direct bridge agent as the destination agent and specifying the destination file in the form *connect_direct_node_name:file_path*.

Before you begin

Before transferring a file, you must configure the Connect:Direct bridge, which is a component of WebSphere MQ Managed File Transfer. For more information, see "Configuring the Connect:Direct bridge" on page 190.

About this task

In this example, the Connect:Direct bridge agent is called CD_BRIDGE. The source agent is called FTE_AGENT and can be any version of WMQFTE. The destination Connect:Direct node is called CD_NODE1. The file to be transferred is located at the file path `/home/helen/file.log` on the system where FTE_AGENT is located. The file is transferred to the file path `/files/data.log` on the system where CD_NODE1 is running.

Procedure

1. Use the fteCreateTransfer command with the value for the **-df** (destination file) parameter in the form *connect_direct_node_name:file_path* and the value of the **-da** (destination agent) parameter specified as the name of the Connect:Direct bridge agent.

   Note: The Connect:Direct node specified by *connect_direct_node_name* is the node that you want the file to be transferred to, not the Connect:Direct node that operates as part of the Connect:Direct bridge.

   ```
   fteCreateTransfer -sa FTE_AGENT -da CD_BRIDGE
     -df CD_NODE1:/files/data.log /home/helen/file.log
   ```

For more information, see "**fteCreateTransfer** (create new file transfer)" on page 461.

2. The source agent FTE_AGENT transfers the file to the Connect:Direct bridge agent CD_BRIDGE. The file is temporarily stored on the system where the Connect:Direct bridge agent is running, in the location defined by the cdTmpDir agent property. The Connect:Direct bridge agent transfers the file to the Connect:Direct node CD_NODE1.

Related concepts:

"The Connect:Direct bridge" on page 263
You can transfer files to and from an existing IBM Sterling Connect:Direct network. Use the Connect:Direct bridge, which is a component of WebSphere MQ Managed File Transfer, to transfer files between MQMFT and IBM Sterling Connect:Direct.

Related tasks:

"Transferring a file from a Connect:Direct node"
You can transfer a file from a Connect:Direct node to a WebSphere MQ Managed File Transfer agent by using the Connect:Direct bridge. You can specify a Connect:Direct node as the source of the transfer by specifying the Connect:Direct bridge agent as the source agent and specifying the source specification in the form *connect_direct_node_name:file_path*.

Related reference:

"The agent.properties file" on page 548
Each agent has its own properties file, `agent.properties`, that must contain the information that an agent uses to connect to its queue manager. The `agent.properties` file can also contain properties that alter the behavior of the agent.

Transferring a file from a Connect:Direct node

You can transfer a file from a Connect:Direct node to a WebSphere MQ Managed File Transfer agent by using the Connect:Direct bridge. You can specify a Connect:Direct node as the source of the transfer by specifying the Connect:Direct bridge agent as the source agent and specifying the source specification in the form *connect_direct_node_name:file_path*.

Before you begin

Before transferring a file, you must configure the Connect:Direct bridge, which is a component of WebSphere MQ Managed File Transfer. See "Configuring the Connect:Direct bridge" on page 190.

About this task

In this example, the Connect:Direct bridge agent is called CD_BRIDGE. The destination agent is called FTE_AGENT and can be any version of WMQFTE. The source Connect:Direct node is called CD_NODE1. The file to be transferred is located at the file path `/home/brian/in.file` on the system where CD_NODE1 is located. The file is transferred to the file path `/files/out.file` on the system where FTE_AGENT is running.

Procedure

Use the **fteCreateTransfer** command with the value for the source specification in the form *connect_direct_node_name:file_path* and the value of the **-sa** parameter specified as the name of the Connect:Direct bridge agent.

Note: The Connect:Direct node specified by *connect_direct_node_name* is the node that you want the file to be transferred from, not the Connect:Direct node that operates as part of the Connect:Direct bridge. For example:

```
fteCreateTransfer -sa CD_BRIDGE -da FTE_AGENT
                  -df /files/out.file CD_NODE1:/home/brian/in.file
```

For more information, see "**fteCreateTransfer** (create new file transfer)" on page 461.

Results

The Connect:Direct bridge agent CD_BRIDGE requests the file from the Connect:Direct node CD_NODE1. The Connect:Direct node sends the file to the Connect:Direct bridge. While the file is being transferred from the Connect:Direct node, the Connect:Direct bridge stores the file temporarily in the location defined by the `cdTmpDir` agent property. When the file has finished transferring from the Connect:Direct node to the Connect:Direct bridge, the Connect:Direct bridge then sends the file to the destination agent FTE_AGENT and deletes the file from the temporary location.

Related concepts:

"The Connect:Direct bridge" on page 263
You can transfer files to and from an existing IBM Sterling Connect:Direct network. Use the Connect:Direct bridge, which is a component of WebSphere MQ Managed File Transfer, to transfer files between MQMFT and IBM Sterling Connect:Direct.

Related reference:

"The agent.properties file" on page 548
Each agent has its own properties file, `agent.properties`, that must contain the information that an agent uses to connect to its queue manager. The `agent.properties` file can also contain properties that alter the behavior of the agent.

Transferring multiple files to a Connect:Direct node

You can transfer multiple files from a WebSphere MQ Managed File Transfer agent to a Connect:Direct node by using the Connect:Direct bridge. To use a Connect:Direct node as the destination of the multiple file transfer, specify the Connect:Direct bridge agent as the destination agent and specify the destination directory in the form *connect_direct_node_name:directory_path*.

Before you begin

Before transferring files, you must configure the Connect:Direct bridge, which is a component of WebSphere MQ Managed File Transfer. See "Configuring the Connect:Direct bridge" on page 190.

About this task

In this example, the source agent is called FTE_AGENT. The Connect:Direct bridge agent is called CD_BRIDGE. The destination Connect:Direct node is called CD_NODE1. The files to be transferred are `/home/jack/data.log`, `/logs/log1.txt`, and `/results/latest` on the system where FTE_AGENT is located. The files are transferred to the directory `/in/files` on the system where CD_NODE1 is running.

Procedure

Use the fteCreateTransfer command with the value for the **-dd** (destination directory) parameter in the form *connect_direct_node_name:directory_path*. Specify the value of the **-da** (destination agent) parameter as the name of the Connect:Direct bridge agent.

Note: The Connect:Direct node specified by *connect_direct_node_name* is the node that you want the files to be transferred to, not the Connect:Direct node that operates as part of the Connect:Direct bridge.

```
fteCreateTransfer -sa FTE_AGENT -da CD_BRIDGE
                  -dd CD_NODE1:/in/files /home/jack/data.log
                  /logs/log1.txt /results/latest
```

For more information, see "**fteCreateTransfer** (create new file transfer)" on page 461.

Results

The source agent FTE_AGENT transfers the first file to the Connect:Direct bridge agent CD_BRIDGE. The Connect:Direct bridge agent temporarily stores the file in the location defined by the cdTmpDir property.

When the file has been completely transferred from the source agent to the Connect:Direct bridge, the Connect:Direct bridge agent sends the file to the Connect:Direct node that is defined by the cdNode agent property. This node sends the file to the destination Connect:Direct node CD_NODE1. The Connect:Direct bridge agent deletes the file from the temporary location when the transfer between the two Connect:Direct nodes is complete. This process is repeated for each specified source file.

Related concepts:

"The Connect:Direct bridge" on page 263
You can transfer files to and from an existing IBM Sterling Connect:Direct network. Use the Connect:Direct bridge, which is a component of WebSphere MQ Managed File Transfer, to transfer files between MQMFT and IBM Sterling Connect:Direct.

Related tasks:

"Transferring a file to a Connect:Direct node" on page 264
You can transfer a file from a WebSphere MQ Managed File Transfer agent to a Connect:Direct node using the Connect:Direct bridge. Specify a Connect:Direct node as the destination of the transfer by specifying the Connect:Direct bridge agent as the destination agent and specifying the destination file in the form `connect_direct_node_name:file_path`.

"Transferring multiple files to Connect:Direct by using wildcards"
To transfer multiple files from a WebSphere MQ Managed File Transfer agent to a Connect:Direct node, use the Connect:Direct bridge. You can use wildcard characters in the source specification that you provide to the **fteCreateTransfer** command. As with all WebSphere MQ Managed File Transfer transfers involving wildcards, only the last part of the file path can contain a wildcard character. For example, /abc/def* is a valid file path and /abc*/def is not valid.

"Transferring a file from a Connect:Direct node" on page 265
You can transfer a file from a Connect:Direct node to a WebSphere MQ Managed File Transfer agent by using the Connect:Direct bridge. You can specify a Connect:Direct node as the source of the transfer by specifying the Connect:Direct bridge agent as the source agent and specifying the source specification in the form `connect_direct_node_name:file_path`.

Related reference:

"The agent.properties file" on page 548
Each agent has its own properties file, `agent.properties`, that must contain the information that an agent uses to connect to its queue manager. The `agent.properties` file can also contain properties that alter the behavior of the agent.

Transferring multiple files to Connect:Direct by using wildcards

To transfer multiple files from a WebSphere MQ Managed File Transfer agent to a Connect:Direct node, use the Connect:Direct bridge. You can use wildcard characters in the source specification that you provide to the **fteCreateTransfer** command. As with all WebSphere MQ Managed File Transfer transfers involving wildcards, only the last part of the file path can contain a wildcard character. For example, /abc/def* is a valid file path and /abc*/def is not valid.

Before you begin

Before transferring a file, you must configure the Connect:Direct bridge, which is a component of WebSphere MQ Managed File Transfer. For more information, see "Configuring the Connect:Direct bridge" on page 190.

About this task

In this example, the source agent is called FTE_AGENT and the Connect:Direct bridge agent is called CD_BRIDGE. The destination Connect:Direct node is called CD_NODE1. The files to be transferred are located in the directory /reports on the system where FTE_AGENT is located. Only files with names that start with report, followed by two characters and the suffix .log, are transferred. For example, the file /reports/report01.log is transferred, but the file /reports/report1.log is not transferred. The files are

transferred to the directory /home/fred on the system where CD_NODE1 is running.

Procedure

1. Use the fteCreateTransfer command with the value for the **-dd** (destination directory) parameter in the form *connect_direct_node_name:directory_path*. For the **-da** (destination agent) parameter, specify the Connect:Direct bridge agent.

   Note: The Connect:Direct node specified by *connect_direct_node_name* is the node that you want the files to be transferred to, not the Connect:Direct node that operates as part of the Connect:Direct bridge.

   ```
   fteCreateTransfer -sa FTE_AGENT -da CD_BRIDGE
                     -dd CD_NODE1:/home/fred "/reports/report??.log"
   ```

   For more information, see "**fteCreateTransfer** (create new file transfer)" on page 461.

2. The source agent FTE_AGENT transfers the first file that matches the pattern /reports/report??.log to the Connect:Direct bridge agent CD_BRIDGE. The Connect:Direct bridge agent temporarily stores the file in the location defined by the cdTmpDir property. When the file has been completely transferred from the source agent to the Connect:Direct bridge, the Connect:Direct bridge agent sends the file to the Connect:Direct node that is defined by the cdNode agent property. This node sends the file to the destination Connect:Direct node CD_NODE1. The Connect:Direct bridge agent deletes the file from the temporary location when the transfer between the two Connect:Direct nodes is complete.This process is repeated for each source file that matches the wildcard pattern /reports/report??.log.

   Note: The list of files that match the pattern /reports/report??.log varies depending on the operating system of the system where the source agent FTE_AGENT is located.

   - If the source agent is located on a system with a Windows operating system, the pattern matching is not case sensitive. The pattern matches all files in the /reports directory with a file name of the form report followed by two characters and a suffix of .log, regardless of the case that the letters are in. For example, Report99.Log is a match.

   - If the source agent is located on a system with a Linux or UNIX operating system, the pattern matching is case sensitive. The pattern matches only those files in the /reports directory with a file name of the form report followed by two characters and a suffix of .log. For example, reportAB.log is a match, but reportAB.LOG and Report99.Log are not matches.

Recovery and restart for transfers to and from Connect:Direct nodes

WebSphere MQ Managed File Transfer might be unable to connect to your IBM Sterling Connect:Direct node during a transfer; for example, if the node becomes unavailable. Either WebSphere MQ Managed File Transfer attempts to recover the transfer, or the transfer fails and an error message is produced.

If the Connect:Direct node becomes unavailable

If the Connect:Direct node becomes unavailable; for example, due to a network or power outage, WebSphere MQ Managed File Transfer recovers a file transfer in the following ways:

- If WebSphere MQ Managed File Transfer has not previously successfully connected to the Connect:Direct node as part of this transfer request, the transfer is tried again for a length of time determined by the values of the **cdMaxConnectionRetries** and **recoverableTransferRetryInterval properties**. These properties are specified in the agent.properties file for the Connect:Direct bridge agent. The transfer fails, and an error message is produced, after the number of failed attempts reaches the value of the **cdMaxConnectionRetries property**. By default, the transfer is attempted indefinitely, with 60 seconds between attempts.

- If WebSphere MQ Managed File Transfer has previously successfully connected to the Connect:Direct node as part of this transfer request, the transfer is tried again for a length of time determined by the values of the **cdMaxPartialWorkConnectionRetries** and **recoverableTransferRetryInterval** properties.

The transfer fails, and an error message is produced, after the number of failed attempts reaches the value of the **cdMaxPartialWorkConnectionRetries** property. By default, the transfer is attempted indefinitely, with 60 seconds between attempts.

- For certain types of Connect:Direct node failure, for example the node being forcibly stopped, Connect:Direct processes go into `Held Due to Error (HE)` status when the node recovers. After the node recovers, WebSphere MQ Managed File Transfer automatically resumes any Connect:Direct processes that are related to the file transfer and have a status of HE.
- If the transfer fails, any temporary files relating to the transfer are deleted from the system that hosts the Connect:Direct bridge. The location of these temporary files is defined by the **cdTmpDir** property.
- If the transfer is from WebSphere MQ Managed File Transfer to Connect:Direct, and a source disposition of delete is specified, then the source files are not deleted if the transfer fails.

If the Connect:Direct node user credentials are invalid

If WebSphere MQ Managed File Transfer fails to connect to the Connect:Direct node because the credentials of the user are rejected by the node, the transfer fails and an error message is produced. In this situation, check that you have provided the correct user credentials for the Connect:Direct node. For more information, see "Mapping credentials for Connect:Direct" on page 193.

If the Connect:Direct bridge agent becomes unavailable

If the Connect:Direct bridge agent becomes unavailable, any ongoing file transfers recover in the same way as standard WebSphere MQ Managed File Transfer transfers. For more information, see "Recovery and restart for WebSphere MQ Managed File Transfer" on page 276.

Related concepts:

"The Connect:Direct bridge" on page 263
You can transfer files to and from an existing IBM Sterling Connect:Direct network. Use the Connect:Direct bridge, which is a component of WebSphere MQ Managed File Transfer, to transfer files between MQMFT and IBM Sterling Connect:Direct.

"Recovery and restart for WebSphere MQ Managed File Transfer" on page 276
If your agent or queue manager are unavailable for any reason, for example because of a power or network failure, WebSphere MQ Managed File Transfer recovers as follows in these scenarios:

Related tasks:

"Configuring the Connect:Direct bridge" on page 190
Configure the Connect:Direct bridge to transfer files between a WebSphere MQ Managed File Transfer network and a Connect:Direct network. The components of the Connect:Direct bridge are a Connect:Direct node and a WebSphere MQ Managed File Transfer agent that is dedicated to communicating with that node. This agent is referred to as the Connect:Direct bridge agent.

Related reference:

"The agent.properties file" on page 548
Each agent has its own properties file, `agent.properties`, that must contain the information that an agent uses to connect to its queue manager. The `agent.properties` file can also contain properties that alter the behavior of the agent.

Submitting a user-defined Connect:Direct process from a file transfer request

You can submit a transfer request for a transfer that goes through the Connect:Direct bridge agent that calls a user-defined Connect:Direct process as part of the file transfer.

By default, when you submit a file transfer request for a transfer that goes through the Connect:Direct bridge, the Connect:Direct bridge agent generates the Connect:Direct process that is used to transfer the file to or from the remote Connect:Direct node.

However, you can configure the Connect:Direct bridge agent to instead call a user-defined Connect:Direct process by using the `ConnectDirectProcessDefinition.xml` file.

The `ConnectDirectProcessDefinition.xml` file

The **fteCreateCDAgent** command creates the file `ConnectDirectProcessDefinitions.xml` in the agent configuration directory *MQ_DATA_PATH*/mqft/config/*coordination_queue_manager*/agents/ *cd_bridge_agent_name*. Before you can call user-defined Connect:Direct processes from the Connect:Direct bridge agent, you must set up process definitions by editing this file.

The file defines one or more process sets that includes the location of one or more Connect:Direct processes that are called as part of a transfer. Each process set includes a number of conditions. If the transfer satisfies all of the conditions of the process set, the process set is used to specify which Connect:Direct processes are called by the transfer. For more information, see "Specifying the Connect:Direct process to start by using the ConnectDirectProcessDefinition.xml file" on page 198.

Intrinsic symbolic variables

You can use the intrinsic symbolic variables that are defined by WebSphere MQ Managed File Transfer to substitute values into user-defined Connect:Direct processes. To follow the Connect:Direct naming convention, all intrinsic symbolic variables used by WebSphere MQ Managed File Transfer have the format %FTE followed by five uppercase alphanumeric characters.

When creating a process to transfer files from a Connect:Direct node to the Connect:Direct bridge system, you must use the intrinsic variable %FTETFILE as the value of TO FILE in the Connect:Direct process. When creating a process to transfer files to a Connect:Direct node from the Connect:Direct bridge system, you must use the intrinsic variable %FTEFFILE as the value of FROM FILE in the Connect:Direct process. These variables contain the temporary file paths that the Connect:Direct bridge agent uses for transfers into and out of the WebSphere MQ Managed File Transfer network.

For more information about intrinsic symbolic variables, see the Connect:Direct product documentation.

Sample Connect:Direct processes

WebSphere MQ File Transfer Edition provides sample Connect:Direct processes. These samples are located in the following directory: *MQ_INSTALLATION_PATH*/mqft/samples/ConnectDirectProcessTemplates.

Specifying the Connect:Direct process to start by using the ConnectDirectProcessDefinition.xml file

Specify which Connect:Direct process to start as part of a WebSphere MQ Managed File Transfer transfer. WebSphere MQ Managed File Transfer provides an XML file that you can edit to specify process definitions.

About this task

The **fteCreateCDAgent** command creates the file `ConnectDirectProcessDefinitions.xml` in the agent configuration directory *MQ_DATA_PATH*/mqft/config/*coordination_queue_manager*/agents/ *cd_bridge_agent_name*. Before you can call user-defined Connect:Direct processes from the Connect:Direct bridge agent, you must set up process definitions by editing this file.

For each process that you want to specify to call as part of a transfer through the Connect:Direct bridge, perform the following steps:

Procedure

1. Define the Connect:Direct process that you want the Connect:Direct bridge agent to call as part of the transfer and save the process template in a file.

2. Open the *MQ_DATA_PATH*/mqft/config/*coordination_queue_manager*/agents/*cd_bridge_agent_name*/ `ConnectDirectProcessDefinitions.xml` file in a text editor.
3. Create a `<processSet>` element.
4. Inside the `<processSet>` element, create a `<condition>` element.
5. Inside the `<condition>` element, create one or more elements that define a condition that the transfer request must match to call the Connect:Direct process you defined in Step 1. These elements can be either `<match>` elements or `<defined>` elements.
   - Use a `<match>` element to specify that the value of a variable must match a pattern. Create the `<match>` element with the following attributes:
     – `variable` - the name of the variable whose value is compared. The variable is an intrinsic symbol. For more information, see "Substitution variables for use with user-defined Connect:Direct processes" on page 686.
     – `value` - the pattern to compare to the value of the specified variable.
     – Optional: `pattern` - the type of pattern used by the value of the `value` attribute. This pattern type can be `wildcard` or `regex`. This attribute is optional and the default is `wildcard`.
   - Use a `<defined>` element to specify that a variable must have a value defined. Create the `<defined>` element with the following attribute:
     – `variable` - the name of the variable that must have a value defined. The variable is an intrinsic symbol. For more information, see "Substitution variables for use with user-defined Connect:Direct processes" on page 686.

   The conditions specified within the `<condition>` element are combined with a logical AND. All conditions must be met for the Connect:Direct bridge agent to call the process specified by this `<processSet>` element. If you do not specify a `<condition>` element, the process set matches all transfers.
6. Inside the `<processSet>` element, create a `<process>` element.
7. Inside the `<process>` element, create a `<transfer>` element. The transfer element specifies the Connect:Direct process that the Connect:Direct bridge agent calls as part of the transfer. Create the `<transfer>` element with the following attribute:
   - `process- -` the location of the Connect:Direct process that you defined in step 1. The location of this file is specified with an absolute path or relative to the *MQ_DATA_PATH*/mqft/config/ *coordination_queue_manager*/agents/*cd_bridge_agent_name* directory.

Results

When searching for a condition match, the Connect:Direct bridge agent searches from the start of the file to the end of the file. The first match that is found is the one that is used.

Related tasks:

"Configuring the Connect:Direct bridge" on page 190
Configure the Connect:Direct bridge to transfer files between a WebSphere MQ Managed File Transfer network and a Connect:Direct network. The components of the Connect:Direct bridge are a Connect:Direct node and a WebSphere MQ Managed File Transfer agent that is dedicated to communicating with that node. This agent is referred to as the Connect:Direct bridge agent.

Related reference:

"Connect:Direct process definitions file format" on page 584
The `ConnectDirectProcessDefinitions.xml` file in the Connect:Direct bridge agent configuration directory specifies the user-defined Connect:Direct process to start as part of the file transfer.

"fteCreateCDAgent (create a Connect:Direct bridge agent)" on page 437
The fteCreateCDAgent command creates a WebSphere MQ Managed File Transfer agent and its associated configuration for use with the Connect:Direct bridge.

Using intrinsic symbolic variables in Connect:Direct processes that are called by WebSphere MQ Managed File Transfer

You can call a user-defined Connect:Direct process from a WebSphere MQ Managed File Transfer transfer and pass in information from the transfer to the Connect:Direct process by using intrinsic symbolic variables in the process definition.

About this task

This example uses intrinsic symbolic variables to pass information from a WebSphere MQ Managed File Transfer transfer in to a user-defined Connect:Direct process. For more information about intrinsic symbolic variables used by WebSphere MQ Managed File Transfer, see "Substitution variables for use with user-defined Connect:Direct processes" on page 686.

In this example, the file is transferred from a WebSphere MQ Managed File Transfer agent to a Connect:Direct bridge node. The first part of the transfer is performed by WebSphere MQ Managed File Transfer. The second part of the transfer is performed by a user-defined Connect:Direct process.

Procedure

1. Create a Connect:Direct process that uses intrinsic symbolic variables.

```
%FTEPNAME PROCESS
 SNODE=%FTESNODE
 PNODEID=(%FTEPUSER,%FTEPPASS)
 SNODEID=(%FTESUSER,%FTESPASS)

COPY001 COPY
 FROM (
  FILE=%FTEFFILE
  DISP=%FTEFDISP
 )
 TO (
  FILE=%FTETFILE
  DISP=%FTETDISP
 )
PEND
```

2. Save this process to a text file at the following location: *MQ_DATA_PATH*/mqft/config/ *coordination_queue_manager*/agents/*cd_bridge_agent*/Example.cdp

3. Edit the `ConnectDirectProcessDefinition.xml` file to include a rule that calls the Connect:Direct process that you created in Step 1.

```
<?xml version="1.0" encoding="UTF-8"?>
<tns:cdprocess xmlns:tns="http://wmqfte.ibm.com/ConnectDirectProcessDefinitions"
               xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xsi:schemaLocation="http://wmqfte.ibm.com/
                ConnectDirectProcessDefinitions ConnectDirectProcessDefinitions.xsd">


    <tns:processSet>
        <tns:condition>
            <tns:match variable="%FTESNODE" value="TOBERMORY" pattern="wildcard" />
        </tns:condition>
        <tns:process>
            <tns:transfer process="Example.cdp" />
        </tns:process>
    </tns:processSet>

</tns:cdprocess>
```

In this example, if a transfer request is submitted to the Connect:Direct bridge agent that has TOBERMORY as its source or destination Connect:Direct node, the `Example.cdp` Connect:Direct process is called.

4. Submit a file transfer request that satisfies the conditions that you defined in the `ConnectDirectProcessDefinition.xml` file in Step 3. For example,

```
fteCreateTransfer -sa ORINOCO -da CD_BRIDGE
                  -sm QM_WIMBLEDON -dm QM_COMMON
                  -de overwrite -df TOBERMORY:/home/bulgaria/destination.txt
                  -sd leave c:\bungo\source.txt
```
In this example, the destination Connect:Direct node is TOBERMORY. This node is the secondary node in the transfer and the value of %FTESNODE is set to TOBERMORY. This command matches the condition that is set in the `ConnectDirectProcessDefinition.xml` file.

5. WebSphere MQ Managed File Transfer transfers the source file to a temporary location on the same system as the Connect:Direct bridge agent.

6. The Connect:Direct bridge agent sets the values of the intrinsic symbolic variables from the information in the transfer request and configuration information. The intrinsic symbolic variables are set to the following values:
   - %FTEPNAME=*process_name* - This value is an 8 character process name generated by the Connect:Direct bridge agent.
   - %FTESNODE=TOBERMORY - This value is set from the **-df** parameter of the **fteCreateTransfer** command.
   - %FTEPUSER,=*primary_node_user* - This information is taken from the `ConnectDirectCredentials.xml` file.
   - %FTEPPASS=*primary_node_user_password* - This information is taken from the `ConnectDirectCredentials.xml` file.
   - %FTESUSER,=*secondary_node_user* - This information is taken from the `ConnectDirectCredentials.xml` file.
   - %FTESPASS=*secondary_node_user_password* - This information is taken from the `ConnectDirectCredentials.xml` file.
   - %FTEFFILE =*temporary_location* - This value is the temporary location of the file on the same system as the Connect:Direct bridge agent.
   - %FTEFDISP=leave - This value is set from the **-sd** parameter of the **fteCreateTransfer** command.
   - %FTETFILE=/home/bulgaria/destination.txt - This value is set from the **-df** parameter of the **fteCreateTransfer** command.
   - %FTETDISP=overwrite - This value is set from the **-de** parameter of the **fteCreateTransfer** command.

7. The Connect:Direct process is started on the Connect:Direct bridge node. Connect:Direct transfers the file from the temporary location on the Connect:Direct bridge system to the destination `/home/bulgaria/destination.txt` on the system where the Connect:Direct node TOBERMORY is running.

Related concepts:

"Submitting a user-defined Connect:Direct process from a file transfer request" on page 269
You can submit a transfer request for a transfer that goes through the Connect:Direct bridge agent that calls a user-defined Connect:Direct process as part of the file transfer.

Related reference:

"Substitution variables for use with user-defined Connect:Direct processes" on page 686
You can define values to substitute in to user-defined Connect:Direct processes by using intrinsic symbolic variables that are specific to WebSphere MQ Managed File Transfer.

Using Connect:Direct processes to submit WebSphere MQ Managed File Transfer transfer requests

You can submit a transfer request to the Connect:Direct bridge agent from a Connect:Direct process. WebSphere MQ Managed File Transfer provides commands that can be called from a **RUN TASK** statement in a Connect:Direct process.

WebSphere MQ Managed File Transfer provides the following commands for use with Connect:Direct processes:

ftetag Specify this command in a step that precedes the **ftebxfer** or **ftecxfer** command to create the required audit information for the transfer. This command takes the source specification of the transfer as a parameter. For information about the format of source specification, see "**fteCreateTransfer** (create new file transfer)" on page 461.

ftebxfer

Specify this command to create a file transfer request when the queue manager that the transfer request is submitted to is located on the same system as the Connect:Direct node that submits the command. This command takes the same parameters as the **fteCreateTransfer** command. For information about these parameters, see "**fteCreateTransfer** (create new file transfer)" on page 461. This command also has an additional parameter:

-qmgrname
Required. The name of the queue manager to submit the command to.

ftecxfer

Specify this command to create a file transfer request when the queue manager that the transfer request is submitted to is located on a different system to the Connect:Direct node that submits the command. This command takes the same parameters as the **fteCreateTransfer** command. For information about the parameters, see "**fteCreateTransfer** (create new file transfer)" on page 461. This command also has three additional parameters:

-qmgrname
Required. The name of the queue manager to submit the command to.

-connname
Required. The host and port of the queue manager to submit the command to, specified in WebSphere MQ CONNAME format. For example, `host.example.com(1337)`.

-channelname
Optional. The name of the channel to use to connect to the queue manager to submit the command to. If this is not specified, a default of SYSTEM.DEF.SVRCONN is used.

Related tasks:

"Creating and submitting a Connect:Direct process that calls WebSphere MQ Managed File Transfer by using the Connect:Direct Requester"
The Connect:Direct Requester is a graphical user interface that you can use to create and submit a Connect:Direct process that calls WebSphere MQ Managed File Transfer.

Related reference:

"Example of a Connect:Direct process file that calls the ftecxfer command" on page 688
An example Connect:Direct process file that calls the WebSphere MQ Managed File Transfer **ftetag** command and the **ftecxfer** command.

Creating and submitting a Connect:Direct process that calls WebSphere MQ Managed File Transfer by using the Connect:Direct Requester

The Connect:Direct Requester is a graphical user interface that you can use to create and submit a Connect:Direct process that calls WebSphere MQ Managed File Transfer.

About this task

This task describes how to create a Connect:Direct process that calls the WebSphere MQ Managed File Transfer **ftecxfer** command or the **ftebxfer** command. Use the **ftecxfer** command when the queue manager that the transfer request is submitted to is located on a different system to the Connect:Direct node that submits the command. Use the **ftebxfer** command when the queue manager that the transfer request is submitted to is located on the same system as the Connect:Direct node that submits the command. The **ftecxfer** command makes a client connection to the agent queue manager of the source agent of the transfer. Before calling the **ftecxfer** command, you must call the **ftetag** command and pass

it the source specification information. This allows the process to be logged and audited in the same way as transfers initiated from MQMFT.

Procedure

1. Start the Connect:Direct Requester.
2. In the **Nodes** tab of the panel, select the Connect:Direct node that is used as the primary node of the process.
3. Select **File** > **New** > **Process**. The Process Properties window opens.
4. In the **Name:** field, type the name of the process.
5. Select the secondary node from the **Snode** > **Name:** list.
6. Select the operating system of the secondary node from the **Snode** > **Operating System:** list.
7. Optional: Complete any further information in this window that you require.
8. Click **OK**. The Process Properties window closes.
9. Create a statement that runs the MQMFT **ftetag** command.
   a. Right-click in the Process window on the **End** statement.
   b. Select **Insert** > **Run Task**. The Run Task Statement window opens.
   c. In the **Label:** field, type Tag.
   d. In the **Optional Parameters or Commands** field, type pgm(*MQ_INSTALLATION_PATH*/bin/ftetag) args(*source_specification*). For more information about the format of *source_specification*, see "**fteCreateTransfer** (create new file transfer)" on page 461.
   e. Click **OK**. The Run Task Statement window closes.
10. Create a statement that runs the MQMFT **ftecxfer** or **ftebxfer** command.
    a. Right-click in the Process window on the **End** statement.
    b. Select **Insert** > **Run Task**. The Run Task Statement window opens.
    c. In the **Label:** field, type Transfer.
    d. In the **Optional Parameters or Commands** field, type pgm(*MQ_INSTALLATION_PATH*/bin/ftecxfer) args(*parameters*) or pgm(*MQ_INSTALLATION_PATH*/bin/ftebxfer) args(*parameters*) depending on which command you choose. The parameters used by the **ftecxfer** and **ftebxfer** commands are the same as the parameters used by the **fteCreateTransfer** command, plus some additional parameters specific to **ftecxfer** and **ftebxfer**. For more information, see "**fteCreateTransfer** (create new file transfer)" on page 461 and "Using Connect:Direct processes to submit WebSphere MQ Managed File Transfer transfer requests" on page 273.
    e. Click **OK**. The Run Task Statement window closes.
11. Optional: Create any additional statements that you require.
12. Submit the process.
    a. Right-click in the Process window.
    b. Select **Submit**. The Connect:Direct Attach window opens.
    c. Enter the user name and password to use to run the process.
    d. Click **OK**.

Related concepts:

"Using Connect:Direct processes to submit WebSphere MQ Managed File Transfer transfer requests" on page 273

You can submit a transfer request to the Connect:Direct bridge agent from a Connect:Direct process. WebSphere MQ Managed File Transfer provides commands that can be called from a **RUN TASK** statement in a Connect:Direct process.

Working with IBM Integration Bus

You can work with WebSphere MQ Managed File Transfer from IBM Integration Bus using the FTEOutput and FTEInput nodes.

- Use the FTEInput node to transfer a file across the network using WebSphere MQ Managed File Transfer and then process that file as part of an Integration Bus flow.
- Use the FTEOutput node to transfer a file that has been output by an Integration Bus flow to another location in the network.

The agents that transfer files to or from the broker agent can be at any level of WebSphere MQ Managed File Transfer.

For more information, refer to the IBM Integration Bus product documentation.

Recovery and restart for WebSphere MQ Managed File Transfer

If your agent or queue manager are unavailable for any reason, for example because of a power or network failure, WebSphere MQ Managed File Transfer recovers as follows in these scenarios:

- Typically, if there is a problem while a file is being transferred, WebSphere MQ Managed File Transfer recovers and restarts that file transfer after the problem is repaired.
- If a file that was in the process of being transferred is deleted or changed while the agent or queue manager are unavailable, the transfer fails and you get a message in the transfer log that provides details about the failure.
- If an agent process fails during a file transfer, the transfer continues when you restart the agent.
- If an agent loses the connection to its agent queue manager, the agent waits while trying to reconnect to the queue manager. When the agent successfully reconnects to its queue manager, the current transfer continues.
- If the agent is stopped for any reason, any resource monitors associated with an agent stop polling. When the agent recovers, the monitors are also restarted, and resource polling resumes.
- For a file transfer with a source disposition of `delete`, if a recovery occurs after all the data is sent from a source agent to a destination agent, the source file is unlocked before deletion. This unlocking means that the source file could possibly be modified before the file is deleted. Therefore, it is considered to be unsafe to delete the source file and the following warning is displayed:

  `BFGTR0075W: The source file has not been deleted because it is possible that the source file was modified after the sourc`

  In this case, verify that the content of the source file is unmodified and then manually delete the source file.

You can check the status of your transfers in the WebSphere MQ Explorer. If any transfers appear as `Stalled`, you might need to take corrective action because the stalled status denotes an issue either with the agent or between the two agents involved in the transfer.

Developing applications

Specifying programs to run

You can run programs on a system where a IBM WebSphere MQ Managed File Transfer agent is running. As part of a file transfer request, you can specify a program to run either before a transfer starts, or after it finishes. Additionally, you can start a program that is not part of a file transfer request by submitting a managed call request.

There are five scenarios in which you can specify a program to run:
- As part of a transfer request, at the source agent, before the transfer starts
- As part of a transfer request, at the destination agent, before the transfer starts
- As part of a transfer request, at the source agent, after the transfer completes
- As part of a transfer request, at the destination agent, after the transfer completes
- Not as part of a transfer request. You can submit a request to an agent to run a program. This scenario is sometimes referred to as a managed call.

There are several ways to specify a program that you want to run. These options are as follows:

Use an Apache Ant task

> Use one of the `fte:filecopy`, `fte:filemove`, and `fte:call` Ant tasks to start a program. Using an Ant task, you can specify a program in any of the five scenarios, using the `fte:presrc`, `fte:predst`, `fte:postdst`, `fte:postsrc`, and `fte:command` nested elements. For more information, see "Program invocation nested elements" on page 917.

Edit the file transfer request message

> You can edit the XML that is generated by a transfer request. Using this method, you can run a program in any of the five scenarios, by adding **preSourceCall**, **postSourceCall**, **preDestinationCall**, **postDestinationCall**, and **managedCall** elements to the XML file. Then, use this modified XML file as the transfer definition for a new file transfer request, for example with the **fteCreateTransfer -td** parameter. For more information, see "Call request message examples" on page 819.

Use the `fteCreateTransfer` command

> You can use the **fteCreateTransfer** command to specify programs to start. You can use the command to specify programs to run in the first four scenarios, as part of a transfer request, but you cannot start a managed call. For information about the parameters to use, see "**fteCreateTransfer** (create new file transfer)" on page 461. For examples of using this command, see "Examples of using fteCreateTransfer to start programs" on page 864.

Use the Web Gateway

> If you have configured a Web Gateway, you can run programs at the destination agent after the transfer has completed. You cannot use this method to submit a managed call request, or to run programs at the source agent, or at the destination agent before the transfer starts. Specify the `x-fte-postdest` header or use the `postdest` form field in the HTTP request. For more information, see "HTTP headers and HTML form fields for using the Web Gateway" on page 867.

The WebSphere MQ Managed File Transfer Web Gateway

The Web Gateway provides a RESTful API, which you can use to interact with your WebSphere MQ Managed File Transfer network.

This section explains the concepts of the Web Gateway, and how the Web Gateway fits into your existing Managed File Transfer network. For more information, see "Scenarios for the Web Gateway" and "How the Web Gateway fits into your WebSphere MQ Managed File Transfer topology" on page 280. For examples of HTTP requests that you can send to the Web Gateway, see "Using the WebSphere MQ Managed File Transfer Service Web Gateway" on page 285.

For information about configuring and securing the Web Gateway in an application server, see "Configuring the Web Gateway" on page 165 and "Securing the Web Gateway" on page 111. To check your Web Gateway setup, see "Verifying your Web Gateway installation" on page 187.

For reference information about the Web Gateway RESTful API, see "Web Gateway API reference" on page 866.

To solve problems related to the Web Gateway, see "Troubleshooting the Web Gateway" on page 380.

Scenarios for the Web Gateway

Use the WebSphere MQ Managed File Transfer Web Gateway to transfer files to WebSphere MQ Managed File Transfer agents and retrieve the status of transfers using an HTTP client.

The Web Gateway is useful if you have files on a system where you do not want to run an agent but where you can use an HTTP client. For example, you can use the Web Gateway for the following tasks:

- Sending files to a WebSphere MQ Managed File Transfer agent from a web page
- Monitoring the status of transfers from a web page
- Sending files from a portable device that is not capable of running the WebSphere MQ Managed File Transfer infrastructure but has HTTP capabilities
- Sending files from an operating system that the WebSphere MQ Managed File Transfer agent is not supported on

Uploading a file using the Web Gateway
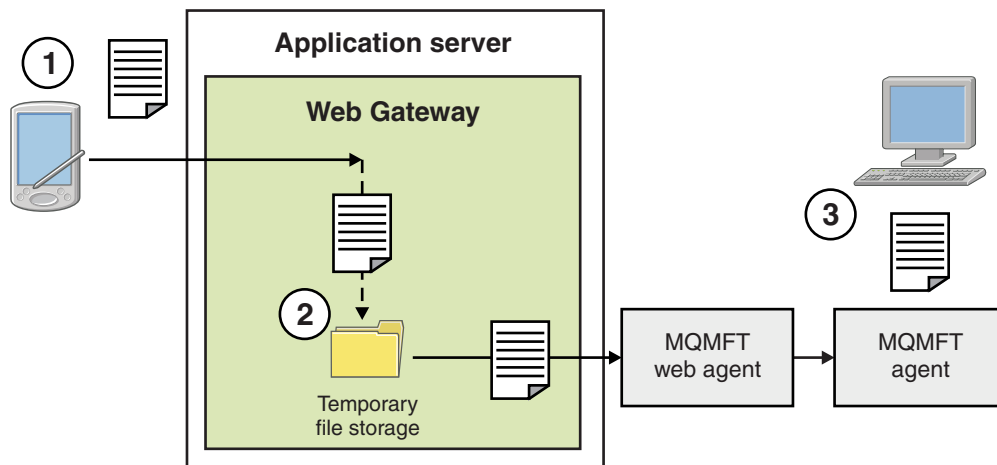
Figure 1. Uploading a file to your WebSphere MQ Managed File Transfer network using the Web Gateway

You can upload a file to the Web Gateway using an HTTP client. The application server that is hosting the Web Gateway application receives the HTTP request and the file is temporarily stored until the web agent starts to transfer it. The web agent transfers the file to the agent that was named as the destination agent in the original transfer request. As shown in Figure 1, there is no need for the HTTP client that submitted the transfer request to have an agent installed. The destination system must have an agent

installed, and the system hosting the Web Gateway application must have a web agent installed.

Downloading a file from a file space
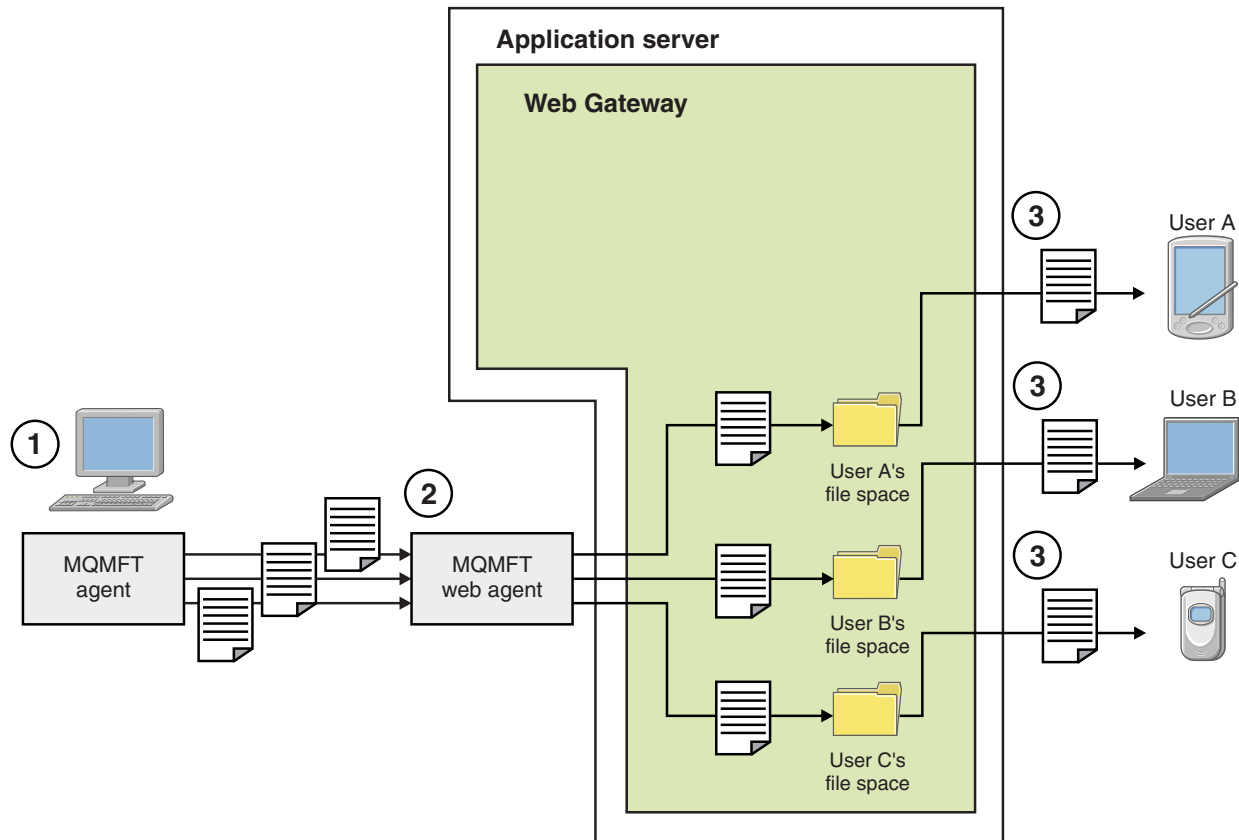
Figure 2. Downloading a file from a file space using the Web Gateway

You can use the Web Gateway to make files available to users in file spaces. A file space is a reserved area of file storage that is associated with a Web Gateway user. Use an agent to transfer a file to the Web Gateway. A web agent on the same system as the Web Gateway application transfers the file to the file space that you specified in the transfer request. A user who owns a file space can download files at their own convenience, and they do not need an agent or other WebSphere MQ Managed File Transfer infrastructure to download the file.

How to use the Web Gateway

WebSphere MQ Managed File Transfer provides an administrative console. You can use the administrative console to create file spaces, modify the set of users who can access a file space, and map users to WebSphere MQ Message Descriptor (MQMD) user IDs. For more information about using the administrative console, see "Administering the WebSphere MQ Managed File Transfer Service Web Gateway" on page 302.

If you prefer, you can program directly to the application programming interface (API) that is provided with the Web Gateway to build a customized application. For more information, see "Web Gateway API reference" on page 866 and "Web Gateway administration API reference" on page 885. There are three principal ways of building an application to work with this API. These are:

Web application

You can write a set of web pages or a web application, which uses Web Gateway API functions to perform the file-related part of its function. A sample application is shipped with the Web Gateway, which demonstrates one way of doing this. For more information, see "Sample web page" on page 327.

Client application

You can write a program using a language such as Perl, Ruby, or Python that runs on client systems and communicates with WebSphere MQ Managed File Transfer by using Web Gateway API functions. Nearly all programming languages have HTTP facilities available. The benefit of this approach is that you can interact with WebSphere MQ Managed File Transfer from platforms where the WebSphere MQ Managed File Transfer agent cannot be deployed.

System integration

This approach uses the same technology as the client application option, but integrates different systems in the datacenter. HTTP provides a common denominator for communication between disparate tools and systems.

How the Web Gateway fits into your WebSphere MQ Managed File Transfer topology

Use the WebSphere MQ Managed File Transfer Service Web Gateway to transfer files to WebSphere MQ Managed File Transfer (MQMFT) agents and retrieve the status of transfers using an HTTP client.

Use the Web Gateway to extend an existing WebSphere MQ Managed File Transfer network to support clients that use the HTTP protocol. The Web Gateway provides a link from clients that are using the HTTP protocol into a WebSphere MQ Managed File Transfer network that already exists. Transfers that use the Web Gateway are logged throughout the transfer. For more information about the purpose of the Web Gateway, see "Scenarios for the Web Gateway" on page 278.
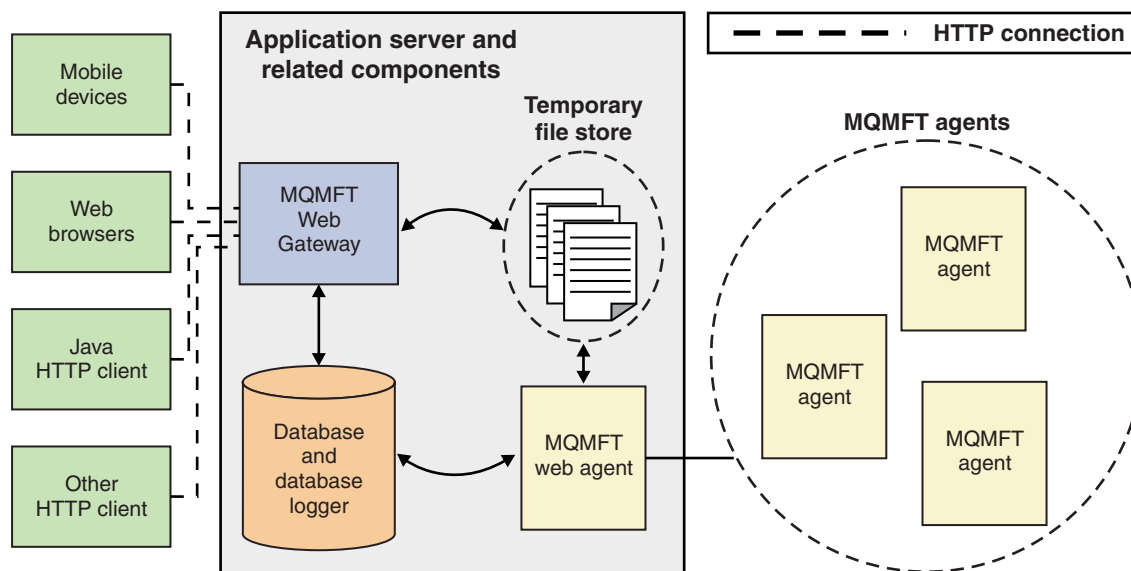
Figure 3. Overview of Web Gateway architecture

The Web Gateway application requires the following component, which is not provided with WebSphere MQ Managed File Transfer:

A Java Platform, Enterprise Edition 5-compliant application server
This application server hosts the Web Gateway application. HTTP requests from clients are directed to the application server, which passes the contents of the requests to the application.

A Web Gateway consists of several parts:

The MQMFT Web Gateway application
The Web Gateway application handles both file uploads and transfer status requests.

When a file is uploaded, the Web Gateway application writes the file data to a temporary store on the file system of the system that the application is running on. The Web Gateway application then submits a file transfer request to the MQMFT agent, which is running on the same system. For more information on this request, see "File transfer request message format" on page 803.

When a request for status information is received, the Web Gateway application connects to the MQMFT database logger database (using the data access facilities provided by the application server) to retrieve the required information. The application then generates the response, which is passed to the client.

An MQMFT web agent
The Web Gateway requires an MQMFT agent installed on the same system as the application. This web agent can be created using the **fteCreateWebAgent** command; see "fteCreateWebAgent (create a WebSphere MQ Managed File Transfer web agent)" on page 480. This agent receives the file transfer request message described in the previous section. The request message refers to the file or files in the temporary store. The agent transfers the files to an existing agent in the MQMFT network, reading the files from the file system store. The source disposition behavior is set to `delete` so that the files are removed after the transfer successfully completes, see fteCreateTransfer at: http://www-01.ibm.com/support/knowledgecenter/SSFKSJ_8.0.0/ com.ibm.wmqfte.doc/start_new_transfer_cmd.htm for more information.

You do not need to specially configure this agent, because the file transfer request is an ordinary message and not specific to the Web Gateway.

The MQMFT database logger and a supported database
To provide status information about transfers, started using the web or by other means, the Web Gateway application must be able to query a database that contains audit information for MQMFT activity. This database is populated by the database logger component provided with the product. Database access is provided by the data access facilities included in each application server. The database does not need to be located on the same system as the other components.

Components needed for Web Gateway scenarios

The following diagrams show the WebSphere MQ Managed File Transfer components, and other objects, that are involved in file transfer requests. All the Java Platform, Enterprise Edition (JEE) resources used in each scenario must be defined in your application server, regardless of which scenario you are using. For details of how to configure the JEE resources, see "Configuring the Web Gateway" on page 165.
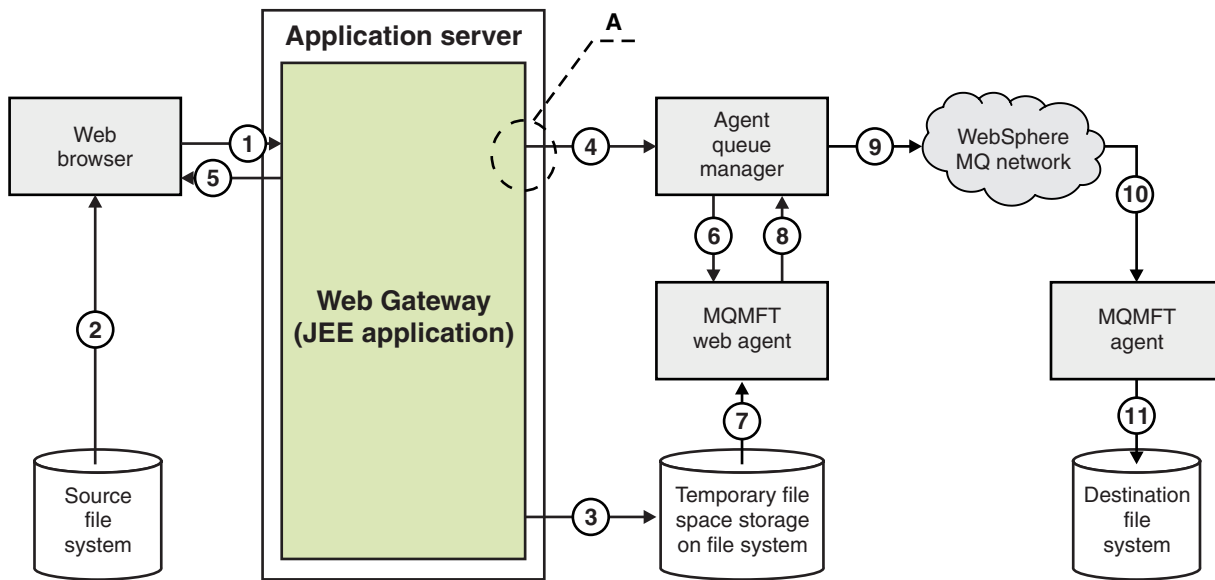
Figure 4. WebSphere MQ Managed File Transfer components involved in a file upload through the Web Gateway

1. A JavaScript application running in the user's web browser uses functions defined by the RESTful API provided by the Web Gateway to upload a file.
2. The file data is read from the file storage located on the same system as the web browser and sent using the HTTP protocol to the application server that hosts the Web Gateway application.
3. The Web Gateway Java Platform, Enterprise Edition (JEE) application receives the file data as the body of an HTTP request and writes it to file storage that is accessible from both the application server and the web agent. If the Web Gateway application and web agent are on the same system, this can be a directory on the system's file system.
4. The Web Gateway application sends a message to the agent queue manager to which the web agent is connected. This message contains instructions that identify both the file to move and the WebSphere MQ Managed File Transfer agent that the file data is sent to. This information is taken from the HTTP request in step 1.
5. The Web Gateway JEE application sends an HTTP response to the web browser.
6. The web agent receives the message that requests the transfer of the file data.
7. The web agent reads the file data, which corresponds to the uploaded file from step 1.
8. The web agent transfers the file data, as a sequence of messages, to the agent queue manager.
9. The agent queue manager transfers the messages, which correspond to the uploaded file from step 1, across the WebSphere MQ network. This might involve exchanging the file data between further queue managers until the data arrives at the queue manager to which the agent running on the destination system is connected.
10. The agent on the destination system receives the messages containing the file data and converts the data back into a file.
11. The file data is written to the file storage at the destination system.

JEE resources used in this scenario:
> A - JMS Queue Connection Factory called `WMQFTEWebAgentConnectionFactory` with a JNDI name of `jms/WMQFTEWebAgentConnectionFactory`
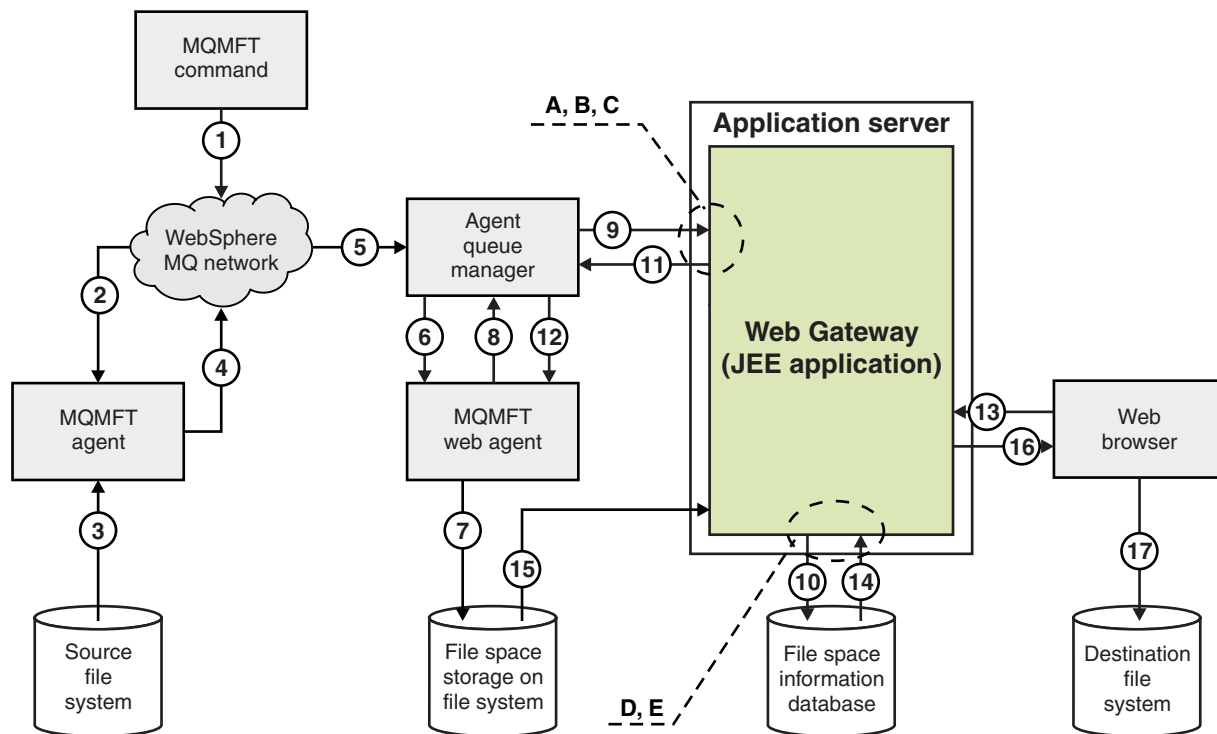
Figure 5. WebSphere MQ Managed File Transfer components involved in a file upload to a file space, and subsequent download from the file space

1. The user, or a process, sends a file transfer request (in the form of a WebSphere MQ message) into the WebSphere MQ network. This request can be sent from the command line or through another MQMFT interface. The message is addressed to the queue manager to which the agent on the source system is connected.
2. The agent on the source system receives the message, which instructs it to perform a file transfer to the web agent.
3. The agent reads the file from the source file system and converts it to a sequence of WebSphere MQ messages.
4. The agent sends the sequence of messages to a queue manager in the WebSphere MQ network.
5. The WebSphere MQ network routes the messages, which contain the file data, to the agent queue manager.
6. The web agent receives the messages, which contain the file data, from the agent queue manager.
7. The web agent writes the file data, as a file, to the file space storage on a file system that is accessible to the Web Gateway JEE application.
8. The web agent sends a message to the agent queue manager, to inform the Web Gateway JEE application that a file has arrived.
9. The Web Gateway JEE application receives the notification message sent from the web agent via the agent queue manager.
10. The Web Gateway JEE application updates a database that contains information about the files that are stored in file spaces.
11. The Web Gateway JEE application sends a response, which is destined for the web agent, to the agent queue manager.
12. The web agent receives the response message and completes the file transfer operation.

13. At some later time, a user or process makes a RESTful HTTP request to the Web Gateway JEE application, to retrieve a file from the user's file space. In this diagram the request is made by a web browser. The request can be made by any HTTP client.

14. The Web Gateway JEE application receives the HTTP request, decodes it, and uses the file space information database to locate the file data.

15. The Web Gateway JEE application reads the file data from the file space storage, which is located on a file system that is accessible from the Web Gateway JEE application.

16. The Web Gateway JEE application sends the file data back to the web browser that requested it.

17. The web browser writes the file data to the file system on the destination system.

JEE resources used in this scenario:

A - JMS Queue called `WMQFTEWebAgentRequestQueue` with a JNDI name of `jms/WMQFTEWebAgentRequestQueue`

B - JMS Queue Connection Factory called `WMQFTEWebAgentConnectionFactory` with a JNDI name of `jms/WMQFTEWebAgentConnectionFactory`

C - Activation Spec called `WMQFTEActivationSpec` with a JNDI name of `jms/WMQFTEActivationSpec`, which is configured with the connection details for the web agent's queue manager

D - Data source called `wmqfte-filespace` with a JNDI name of `jdbc/wmqfte-filespace`

E - JDBC Provider referenced by the data source `jdbc/wmqfte-filespace`
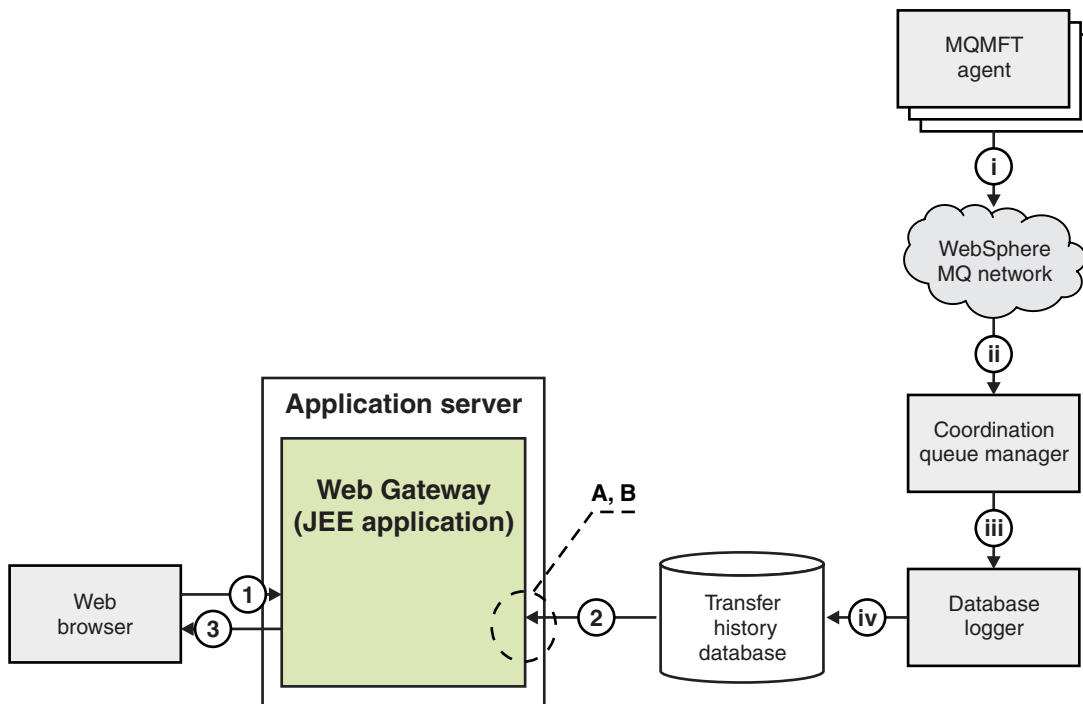
Figure 6. Requesting the status of file transfers through the Web Gateway

1. A JavaScript application running in the user's web browser sends a RESTful HTTP request to the Web Gateway application, requesting information about a transfer.
2. The Web Gateway application queries a database containing information about file transfers that have taken place in a network of WebSphere MQ Managed File Transfer agents.
3. The Web Gateway application returns the result of the query to the JavaScript application.

Activities that occur during the previous steps:

i - WebSphere MQ Managed File Transfer agents produce messages containing information about file transfers that are taking place.

ii - The queue managers route these messages to a designated queue manager that is performing the coordination queue manager role.

iii - The coordination queue manager is connected to the database logger component. The database logger receives a copy of each message that relates to a transfer being performed by an agent.

iv - The database logger records the information about transfers in a transfer history database, so that it can be queried by other applications including the Web Gateway.

JEE resources used in this scenario:

A - Data source called `wmqfte-filespace` with a JNDI name of `jdbc/wmqfte-database`

B - JDBC provider referenced by the data source `wmqfte-database`

Using the WebSphere MQ Managed File Transfer Service Web Gateway

You can upload files, query the files in a file space, view the status of file transfers, and delete files from a file space by creating HTTP requests that you submit to the Web Gateway.

Before configuring or using the Web Gateway, refer to "Scenarios for the Web Gateway" on page 278 and "How the Web Gateway fits into your WebSphere MQ Managed File Transfer topology" on page 280. These topics explain the purpose and components of the Web Gateway.

You can customize your HTTP requests by using HTTP headers or HTML form fields to supply extended information with your request. For more information about the options available, see "HTTP headers and HTML form fields for using the Web Gateway" on page 867.

The following topics explain how to create HTTP requests to submit to the Web Gateway. For more information about the format of these requests and the Web Gateway API, see "Web Gateway API reference" on page 866.

You do not need administrative rights to use these examples. If you want to administer the Web Gateway, for example by creating or deleting file spaces for users, see the topic "Administering the WebSphere MQ Managed File Transfer Service Web Gateway" on page 302.

Example: Transferring a file to a file space

Transfer a single file to a WebSphere MQ Managed File Transfer file space. You can specify a file space as the destination of a file transfer by using the **-du** parameter with the **fteCreateTransfer** command.

About this task

When transferring a file to a file space, the WebSphere MQ Managed File Transfer Web Gateway checks whether the transfer would cause the file space quota to be exceeded. If the quota would be exceeded, an error is produced and the file transfer fails. The Web Gateway administrator can increase the size of the file space quota by submitting an HTTP request. For an example request, see the topic "Example: Modifying file space configuration" on page 306.

The file space quota is checked before the transfer begins. If you are using more than one agent to transfer files to the same file space, or if the Web Gateway administrator reduces the file space quota while a file is being transferred to that file space, one or more transfers might succeed even though they cause the file space quota to be exceeded.

In this example, the source file is called `/tmp/Accounts.csv` and is located on the same system as the source agent, *AGENT_1*. The destination file space john, which belongs to the user *john*, is located on the

same system as the agent *FS_AGENT*. The user requesting the transfer has write access to the file space john. The agent *FS_AGENT* uses the queue manager *FS_QM*.

Procedure

Type the following command:

```
fteCreateTransfer -sa AGENT_1 -da FS_AGENT -dm FS_QM -du john /tmp/Accounts.csv
```

The file /tmp/Accounts.csv is transferred to the file space john. The user *john* can download this file from the file space when it is required.

Related concepts:

"File spaces" on page 313
A file space is a reserved area of file storage that is associated with a Web Gateway user. A file space has an allocated quota of storage. Access to the file space is restricted to users with authorization to read from it or write to it.

Related tasks:

"Starting a new file transfer" on page 205
You can start a new file transfer from the WebSphere MQ Explorer or from the command line and you can choose to transfer either a single file or multiple files in a group.

Example HTTP flows

You can construct HTTP requests and submit them to the WebSphere MQ Managed File Transfer Web Gateway. These examples show you sample requests and the corresponding HTTP responses from the Web Gateway.

You can use various technologies to submit requests to, and interpret responses from, the Web Gateway. For example, you can write a web application. For information about the example web application which is included with the Web Gateway, see "Sample web page" on page 327.

If you want to communicate with the Web Gateway by using a web application, you can use either HTML forms or the Javascript XMLHttpRequest function. To upload a file, you must use an HTML form, because browsers prevent Javascript from accessing files from the local system, for security reasons. The form can be controlled and submitted by Javascript if you prefer. To request the status of a transfer, XMLHttpRequest is most likely to be appropriate, although other techniques are possible; loading content into an invisible iFrame element, for example.

You can also write a client application in a language such as Ruby or Perl to communicate with the Web Gateway API.

Example: Sending a file using an HTTP request:

You can send a single file to a destination agent file system by submitting a request through the WebSphere MQ Managed File Transfer Web Gateway.

About this task

File contents can be uploaded to any standard WebSphere MQ Managed File Transfer agent as POST data using the multipart/form-data Content-Type. This should be submitted to a location containing the target agent and file destination in the following format: /fte/file/agent/*agent_name@queue_manager/filepath*. You can modify the file transfer request parameters using the custom HTTP headers described in "HTTP headers and HTML form fields for using the Web Gateway" on page 867.

When you submit a file transfer request using the Web Gateway, your user ID in the application server environment is checked to see if it is mapped to a WebSphere MQ Message Descriptor (MQMD) user ID. The mappings between application server user ID (web user ID) and MQMD user ID are created by your

Web Gateway administrator. For more details, see the topic "Example: Mapping web user IDs to MQMD user IDs" on page 312. If there is no MQMD user ID defined for your web user ID, the value of the **defaultMQMDUserID** servlet initialization parameter is used. This parameter is defined during deployment of the Web Gateway application.

Use the following example to transfer a text file to the destination file path *destination-root-path*/temp and destination file name myfile.txt on the destination agent ACCOUNTS. Use an MD5 checksum to check the integrity of the transferred file. The content of the file is:

```
Account No, Balance
123456, 100.00
234567, 1022.00
345678, 2801.00
456789, 16.75
```

The server hosting the WebSphere MQ Managed File Transfer Web Gateway is example.com.

Procedure

1. Create an HTTP request with this format:

```
POST HTTP/1.1 /fte/file/agent/ACCOUNTS@QM/temp
Host: example.com
Content-Type: multi-part/form-data; boundary=Aa6b74
x-fte-checksum: MD5


--Aa6b74
Content-Disposition: form-data; name="files"; filename="myfile.txt"
Content-Type: text/plain

Account No, Balance
123456, 100.00
234567, 1022.00
345678, 2801.00
456789, 16.75
--Aa6b74
```

2. Submit the request to the Web Gateway. The Web Gateway returns an HTTP response with this format.

```
HTTP/1.1 200 OK
Server: WAS/6.0
Content-Length: 0
x-fte-id: 4d63c28ae6e72eb9c51cd812736acd4362ef5

<transfers>
 <submission id="4d63c28ae6e72eb9c51cd812736acd4362ef5">
 </submission>
</transfers>
```

The value of x-fte-id is the transfer ID. You can use this transfer ID in an HTTP request for information about the status of the transfer. For an example request, see the topic "Example: Viewing the status of a file transfer using an HTTP request" on page 288.

Related reference:

"Uniform Resource Identifier syntax for using the Web Gateway" on page 869
A WebSphere MQ Managed File Transfer Uniform Resource Identifier (URI) is distinguished from other
WebSphere MQ URIs by the context root specified at deploy time. The recommended context root is
/wmqfte.

"HTTP headers and HTML form fields for using the Web Gateway" on page 867
You can customize a request to create or retrieve a resource by using HTTP headers or HTML form fields.
Each parameter maps to a property or function of WebSphere MQ Managed File Transfer.

Example: Viewing the status of a file transfer using an HTTP request:

You can view the status of your file transfer by submitting a request through the . The Web Gateway
returns information in XML format that describes the current status of the specified transfer. To view the
status of file transfers by using the Web Gateway, you must have a database logger in your WebSphere
MQ Managed File Transfer network.

About this task

A successful request returns an HTTP status code of 200 and an XML payload that describes the current
status of the transfer. You can use this XML to view details of the transfer including the status of the
transfer, the transfer ID, source and destination agent details, and information about the transfer's source
and destination files.

You can view the status of a file transfer if you initiated the upload or if you own the file space to which
the file is transferred. If your user ID is associated with either of the WebSphere MQ Managed File
Transfer security roles wmqfte-audit or wmqfte-admin, you can view the status of all file transfers in your
WebSphere MQ Managed File Transfer network.

The following steps describe how to submit a request. In this example, the server hosting the WebSphere
MQ Managed File Transfer Web Gateway is example.com and the HTTP request is submitted using a web
browser which identifies itself as mozilla.

Procedure

1. Create an HTTP request with the following format:
   ```
   GET HTTP/1.1 /transfer/414d51205245444841542e434f4f5244ed60b44b03310020
   Host: example.com
   User-Agent: mozilla
   ```

   The final part of the URL is the valid 48-character hexadecimal WebSphere MQ Managed File Transfer
   transfer ID of the transfer you want to view.

2. Submit the request to the Web Gateway. The Web Gateway returns an HTTP response with the
   following format:
   ```
   HTTP/1.1 200 OK
   Server: WAS/6.0
   Content-Length: 1664
   Content-type: application/xml

   <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
   <transfers>
    <transfer start-time="2010-04-01T13:10:04.209+01:00" status="Complete"
     id="414d51205245444841542e434f4f5244ed60b44b03310020">
     <source>
      <agent qmgr="REDHAT.SOURCE.QM" name="REDHAT.SOURCE.AGENT" />
      <metadata>
       <key value="REDHAT.SOURCE.AGENT" name="com.ibm.wmqfte.SourceAgent" />
       <key value="REDHAT.DEST.AGENT" name="com.ibm.wmqfte.DestinationAgent" />
       <key value="192.168.243.133" name="com.ibm.wmqfte.OriginatingHost" />
       <key value="fteuser" name="com.ibm.wmqfte.MqmdUser" />
   ```

```
      <key value="414d51205245444841542e434f4f5244ed60b44b03310020"
        name="com.ibm.wmqfte.TransferId" />
      <key value="fteuser" name="com.ibm.wmqfte.OriginatingUser" />
     </metadata>
    </source>
    <destination>
     <agent qmgr="REDHAT.SOURCE.QM" name="REDHAT.SOURCE.AGENT" />
     <metadata>
      <key value="REDHAT.SOURCE.AGENT" name="com.ibm.wmqfte.SourceAgent" />
      <key value="REDHAT.DEST.AGENT" name="com.ibm.wmqfte.DestinationAgent" />
      <key value="fteuser" name="com.ibm.wmqfte.MqmdUser" />
      <key value="192.168.243.133" name="com.ibm.wmqfte.OriginatingHost" />
      <key value="fteuser" name="com.ibm.wmqfte.OriginatingUser" />
      <key value="414d51205245444841542e434f4f5244ed60b44b03310020"
        name="com.ibm.wmqfte.TransferId" />
     </metadata>
    </destination>
      <stats retry-count="0" file-warnings="0" file-failures="0"
        bytes-transferred="67" />
      <transfer-set>
        <file result-code="0" mode="text">
          <source-file name="/home/fteuser/accounts.txt">
       <attribute-values last-modified="2010-03-17T16:55:17.000Z"
            file-size="67" disposition="leave" checksum-method="none" />
          </source-file>
          <destination-file name="/tmp/accounts.txt">
            <attribute-values last-modified="2010-04-01T13:10:04.000+01:00"
             file-size="67" exists-action="error" checksum-method="none" />
          </destination-file>
        </file>
      </transfer-set>
    </transfer>
  </transfers>
```

An invalid request returns an HTTP error code and a WebSphere MQ Managed File Transfer error message. To identify the cause of the error, see Troubleshooting the Web Gateway.

Related reference:
"Uniform Resource Identifier syntax for using the Web Gateway" on page 869
A WebSphere MQ Managed File Transfer Uniform Resource Identifier (URI) is distinguished from other WebSphere MQ URIs by the context root specified at deploy time. The recommended context root is /wmqfte.
"HTTP headers and HTML form fields for using the Web Gateway" on page 867
You can customize a request to create or retrieve a resource by using HTTP headers or HTML form fields. Each parameter maps to a property or function of WebSphere MQ Managed File Transfer.

Example: Querying multiple file transfers using an HTTP request:

You can query the status of multiple file transfers by submitting a request through the WebSphere MQ Managed File Transfer Web Gateway. The Web Gateway returns information in either XML or JSON format that describes the status of the transfers that match the query.

About this task

You can create a URI query that requests transfer information for all transfers that match the query. You can query transfers by their associated details, including the source agent, the destination agent, the source file, the destination file, the transfer status, the metadata, the transfer start time, the transfer end time, and the job name. You can sort the transfer information that is returned by agent, status, start time, end time, or job name, and you can specify the number of results to return. A successful request returns an HTTP status code of 200 and a payload that describes the status of the transfers that match the query.

The following steps describe how to submit a request. In this example, the server hosting the WebSphere MQ Managed File Transfer Web Gateway is example.com. The query requests information that fulfills the following criteria:

- It is from transfers that completed before 1pm UTC on Thursday 26th August 2010, specified by the endbefore=2010-08-26T13:00:00 query
- It is from transfers that have AGENT_TITAN as either the source agent or destination agent, specified by the agent=AGENT_TITAN query
- It is sorted by job name in ascending order, specified by the sortby=jobname and sort=ascending queries
- It includes only the first three transfers that match the full query, specified by the count=3 query
- It is returned in JSON format, specified by the Accept: application/json header.

For more information about query parameters, see "Query parameters" on page 871. For more information about the parameters used to sort the results, see "Result format parameters" on page 874.

The following steps describe how to submit a request. In this example, the server hosting the WebSphere MQ Managed File Transfer Web Gateway is example.com and the HTTP request is submitted using a web browser which identifies itself as mozilla.

Procedure

1. Create an HTTP request with the following format:

```
GET HTTP/1.1 /transfer/?endbefore=2010-08-26T13:00:00&agent=AGENT_TITAN
                        &sortby=jobname&sort=ascending&count=3
Host: example.com
User-Agent: mozilla
Accept: application/json
```

2. Submit the request to the Web Gateway. The Web Gateway returns an HTTP response with the following format:

```
{

    "transfers" : {

        "transfer" : {
            "end-time" : "2010-08-23T14:13:03.260Z",
            "status" : "Complete",
            "start-time" : "2010-08-23T14:12:39.076Z",
            "id" : "414d51205745422e4654452020202020c1a1a34b03720120",
            "result" : {
                "code" : "0",
                "text" : "BFGRP0032I: The file transfer request has successfully completed."
            }
            ,
            "destination" : {

                "metadata" : {

                    "key" : [
                        {
                            "name" : "com.ibm.wmqfte.JobName",
                            "value" : "ALPHA"
                        }
                        ,
                        {
                            "name" : "com.ibm.wmqfte.SourceAgent",
                            "value" : "AGENT_TITAN"
                        }
                        ,
                        {
                            "name" : "com.ibm.wmqfte.DestinationAgent",
                            "value" : "AGENT_MIMAS"
```

```
                }
                ,
                {

                    "name" : "com.ibm.wmqfte.MqmdUser",
                    "value" : "rich"
                }
                ,
                {

                    "name" : "com.ibm.wmqfte.OriginatingHost",
                    "value" : "iceman.example.com."
                }
                ,
                {

                    "name" : "com.ibm.wmqfte.OriginatingUser",
                    "value" : "rich"
                }
                ,
                {

                    "name" : "com.ibm.wmqfte.TransferId",
                    "value" : "414d51205745422e4654452020202020c1a1a34b03720120"
                }

            ]
        }
        ,
        "agent" : {
            "name" : "AGENT_MIMAS",
            "qmgr" : "QM_SATURN"
        }
    }
    ,
    "stats" : {
        "bytes-transferred" : "259354303",
        "retry-count" : "0",
        "file-warnings" : "0",
        "file-failures" : "0"
    }
    ,
    "transfer-set" : {

        "file" : {
            "result-code" : "0",
            "mode" : "text",
            "source-file" : {
                "name" : "\/home\/rich\/file.zip",
                "attribute-values" : {
                    "last-modified" : "2010-08-19T14:16:57.000Z",
                    "checksum-value" : "98611a272a27d373f92d73a08cf0d4f4",
                    "checksum-method" : "MD5",
                    "file-size" : "259354303",
                    "disposition" : "leave"
                }
            }
            ,
            "destination-file" : {
                "name" : "\/tmp\/file.zip",
                "attribute-values" : {
                    "exists-action" : "error",
                    "last-modified" : "2010-08-23T14:13:02.000Z",
                    "checksum-value" : "98611a272a27d373f92d73a08cf0d4f4",
                    "checksum-method" : "MD5",
                    "file-size" : "259354303"
                }
            }
        }
    }
    ,
```

```
        "source" : {

            "metadata" : {

                "key" : [
                    {
                        "name" : "com.ibm.wmqfte.JobName",
                        "value" : "ALPHA"
                    }
                    ,
                    {
                        "name" : "com.ibm.wmqfte.SourceAgent",
                        "value" : "AGENT_TITAN"
                    }
                    ,
                    {
                        "name" : "com.ibm.wmqfte.DestinationAgent",
                        "value" : "AGENT_MIMAS"
                    }
                    ,
                    {
                        "name" : "com.ibm.wmqfte.OriginatingHost",
                        "value" : "iceman.example.com."
                    }
                    ,
                    {
                        "name" : "com.ibm.wmqfte.MqmdUser",
                        "value" : "rich"
                    }
                    ,
                    {
                        "name" : "com.ibm.wmqfte.TransferId",
                        "value" : "414d51205745422e4654452020202020c1a1a34b03720120"
                    }
                    ,
                    {
                        "name" : "com.ibm.wmqfte.OriginatingUser",
                        "value" : "rich"
                    }

                ]
            }
            ,
            "agent" : {
                "name" : "AGENT_TITAN",
                "qmgr" : "QM_SATURN"
            }
        }
    }
  }

  "transfer" : {
        "end-time" : "2010-08-25T15:20:03.260Z",
        "status" : "Complete",
        "start-time" : "2010-08-25T15:19:39.076Z",
        "id" : "414d51205745422e4654452020202020c1a1a34b03720120",
        "result" : {
           "code" : "0",
           "text" : "BFGRP0032I: The file transfer request has successfully completed."
        }
        ,
        "destination" : {

            "metadata" : {

                "key" : [
                    {
```

```
                  "name" : "com.ibm.wmqfte.JobName",
                  "value" : "BRAVO"
              }
              ,
              {
                  "name" : "com.ibm.wmqfte.SourceAgent",
                  "value" : "AGENT_RHEA"
              }
              ,
              {
                  "name" : "com.ibm.wmqfte.DestinationAgent",
                  "value" : "AGENT_TITAN"
              }
              ,
              {
                  "name" : "com.ibm.wmqfte.MqmdUser",
                  "value" : "rich"
              }
              ,
              {
                  "name" : "com.ibm.wmqfte.OriginatingHost",
                  "value" : "iceman.example.com."
              }
              ,
              {
                  "name" : "com.ibm.wmqfte.OriginatingUser",
                  "value" : "rich"
              }
              ,
              {
                  "name" : "com.ibm.wmqfte.TransferId",
                  "value" : "414d51205745422e4654452020202020c1a1a34b03720120"
              }
          ]
      }
      ,
      "agent" : {
          "name" : "AGENT_TITAN",
          "qmgr" : "QM_SATURN"
      }
  }
  ,
  "stats" : {
      "bytes-transferred" : "259354303",
      "retry-count" : "0",
      "file-warnings" : "0",
      "file-failures" : "0"
  }
  ,
  "transfer-set" : {

      "file" : {
          "result-code" : "0",
          "mode" : "text",
          "source-file" : {
              "name" : "\/home\/rich\/file2.zip",
              "attribute-values" : {
                  "last-modified" : "2010-08-19T14:16:57.000Z",
                  "checksum-value" : "98611a272a27d373f92d73a08cf0d4f4",
                  "checksum-method" : "MD5",
                  "file-size" : "259354303",
                  "disposition" : "leave"
              }
          }
          ,
          "destination-file" : {
```

```
                        "name" : "\/tmp\/file2.zip",
                        "attribute-values" : {
                           "exists-action" : "error",
                           "last-modified" : "2010-08-25T15:120:02.000Z",
                           "checksum-value" : "98611a272a27d373f92d73a08cf0d4f4",
                           "checksum-method" : "MD5",
                           "file-size" : "259354303"
                        }
                     }
                  }
               }
               ,
               "source" : {

                  "metadata" : {

                     "key" : [
                        {
                           "name" : "com.ibm.wmqfte.JobName",
                           "value" : "BRAVO"
                        }

                        ,
                        {
                           "name" : "com.ibm.wmqfte.SourceAgent",
                           "value" : "AGENT_RHEA"
                        }

                        ,
                        {
                           "name" : "com.ibm.wmqfte.DestinationAgent",
                           "value" : "AGENT_TITAN"
                        }

                        ,
                        {
                           "name" : "com.ibm.wmqfte.OriginatingHost",
                           "value" : "iceman.example.com."
                        }

                        ,
                        {
                           "name" : "com.ibm.wmqfte.MqmdUser",
                           "value" : "rich"
                        }

                        ,
                        {
                           "name" : "com.ibm.wmqfte.TransferId",
                           "value" : "414d51205745422e4654452020202020c1a1a34b03720120"
                        }

                        ,
                        {
                           "name" : "com.ibm.wmqfte.OriginatingUser",
                           "value" : "rich"
                        }

                     ]
                  }
                  ,
                  "agent" : {
                     "name" : "AGENT_RHEA",
                     "qmgr" : "QM_SATURN"
                  }
               }
            }
         }
      }
      "transfer" : {
            "end-time" : "2010-08-21T14:13:03.260Z",
            "status" : "Complete",
            "start-time" : "2010-08-21T14:12:39.076Z",
            "id" : "414d51205745422e4654452020202020c1a1a34b03720120",
```

```
"result" : {
   "code" : "0",
   "text" : "BFGRP0032I: The file transfer request has successfully completed."
}
,
"destination" : {

   "metadata" : {

      "key" : [
         {
            "name" : "com.ibm.wmqfte.JobName",
            "value" : "CHARLIE"
         }
         ,
         {

            "name" : "com.ibm.wmqfte.SourceAgent",
            "value" : "AGENT_TITAN"
         }
         ,
         {

            "name" : "com.ibm.wmqfte.DestinationAgent",
            "value" : "AGENT_DIONE"
         }
         ,
         {

            "name" : "com.ibm.wmqfte.MqmdUser",
            "value" : "rich"
         }
         ,
         {

            "name" : "com.ibm.wmqfte.OriginatingHost",
            "value" : "iceman.example.com."
         }
         ,
         {

            "name" : "com.ibm.wmqfte.OriginatingUser",
            "value" : "rich"
         }
         ,
         {

            "name" : "com.ibm.wmqfte.TransferId",
            "value" : "414d51205745422e4654452020202020c1a1a34b03720120"
         }

      ]
   }
   ,
   "agent" : {
      "name" : "AGENT_DIONE",
      "qmgr" : "QM_SATURN"
   }
}
,
"stats" : {
   "bytes-transferred" : "259354303",
   "retry-count" : "0",
   "file-warnings" : "0",
   "file-failures" : "0"
}
,
"transfer-set" : {

   "file" : {
      "result-code" : "0",
      "mode" : "text",
      "source-file" : {
```

```
                "name" : "\/home\/rich\/file3.zip",
                "attribute-values" : {
                    "last-modified" : "2010-08-19T14:16:57.000Z",
                    "checksum-value" : "98611a272a27d373f92d73a08cf0d4f4",
                    "checksum-method" : "MD5",
                    "file-size" : "259354303",
                    "disposition" : "leave"
                }
            }
            ,
            "destination-file" : {
                "name" : "\/tmp\/file3.zip",
                "attribute-values" : {
                    "exists-action" : "error",
                    "last-modified" : "2010-08-21T14:13:02.000Z",
                    "checksum-value" : "98611a272a27d373f92d73a08cf0d4f4",
                    "checksum-method" : "MD5",
                    "file-size" : "259354303"
                }
            }
        }
    }
    ,
    "source" : {

        "metadata" : {

            "key" : [
                {
                    "name" : "com.ibm.wmqfte.JobName",
                    "value" : "CHARLIE"
                }
                ,
                {

                    "name" : "com.ibm.wmqfte.SourceAgent",
                    "value" : "AGENT_TITAN"
                }

                ,
                {

                    "name" : "com.ibm.wmqfte.DestinationAgent",
                    "value" : "AGENT_DIONE"
                }

                ,
                {

                    "name" : "com.ibm.wmqfte.OriginatingHost",
                    "value" : "iceman.example.com."
                }

                ,
                {

                    "name" : "com.ibm.wmqfte.MqmdUser",
                    "value" : "rich"
                }

                ,
                {

                    "name" : "com.ibm.wmqfte.TransferId",
                    "value" : "414d51205745422e4654452020202020c1a1a34b03720120"
                }

                ,
                {

                    "name" : "com.ibm.wmqfte.OriginatingUser",
                    "value" : "rich"
                }

            ]
        }
        ,
        "agent" : {
```

```
                    "name" : "AGENT_TITAN",
                    "qmgr" : "QM_SATURN"
                }
            }
        }
    }
}
```

Related reference:

A WebSphere MQ Managed File Transfer Uniform Resource Identifier (URI) is distinguished from other
WebSphere MQ URIs by the context root specified at deploy time. The recommended context root is
`/wmqfte`.

You can customize a request to create or retrieve a resource by using HTTP headers or HTML form fields.
Each parameter maps to a property or function of WebSphere MQ Managed File Transfer.

Example: Listing all files in a file space:

You can list the contents of a file space by submitting an HTTP request to the WebSphere MQ Managed
File Transfer Web Gateway. The Web Gateway returns a response in XML or JSON format that lists the
contents of a file space. You are authorized to list the contents of a file space if you are the owner of the
file space or you have the security role `wmqfte-admin`.

About this task

A successful request returns an HTTP status code of 200 and a payload that lists the first 100 files in the
file space. This response is returned in either XML (the default) or JSON format dependent on the
`'Accept'` header specified in the request.

The following steps describe how to submit a request. In this example, the server hosting the WebSphere
MQ Managed File Transfer Web Gateway is `example.com` and the HTTP request is submitted using a web
browser which identifies itself as `mozilla`. The name of the file space to list is `'john'` and it contains two
files. The header `'Accept: application/xml'` specifies that the Web Gateway should return the results in
XML format. For more information about the formats that are returned by a file space list request, see
.

Procedure

1. Create an HTTP request with the following format:
   ```
   GET HTTP/1.1 /filespace/john
   Host: example.com
   User-Agent: mozilla
   Accept: application/xml
   ```
2. Submit the request to the Web Gateway.

Results

The Web Gateway returns an HTTP response with the following format:
```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<fileSpaces xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:noNamespaceSchemaLocation="WebFileSpaceList.xsd">
  <fileSpace size="2" name="john">
    <file fileLink="/wmqfte/filespace/john/
      414d51205745422e4654452020202020c1a1a34b03720120/filename"
        fsLocation="/var/ibm/WMQFTE/web/fte/transfer/
          414d51205745422e4654452020202020c1a1a34b03720120/file-0"
        transferLink="/wmqfte/transfer/414d51205745422e4654452020202020c1a1a34b03720120"
        transferID="414d51205745422e4654452020202020c1a1a34b03720120">
```

```
            <attribute-values mode="text" created="2010-08-26T11:45:02.000Z" size="259354303"
                        checksum-value="98611a272a27d373f92d73a08cf0d4f4" checksum-method="MD5"/>
    </file>
    <file fileLink="/wmqfte/filespace/john/
      414d51205745422e4654452020202020c1a1a34b06520120/filename"
        fsLocation="/var/ibm/WMQFTE/web/fte/transfer/
          414d51205745422e4654452020202020c1a1a34b06520120/file-0"
        transferLink="/wmqfte/transfer/414d51205745422e4654452020202020c1a1a34b06520120"
        transferID="414d51205745422e4654452020202020c1a1a34b06520120">
      <attribute-values mode="text" created="2010-08-26T12:15:02.260Z" size="259554303"
                        checksum-value="98611a272a27d37bf22d73a08cf0d4f4" checksum-method="MD5"/>
    </file>
  </fileSpace>
</fileSpaces>
```

Related reference:

"Uniform Resource Identifier syntax for using the Web Gateway" on page 869
A WebSphere MQ Managed File Transfer Uniform Resource Identifier (URI) is distinguished from other WebSphere MQ URIs by the context root specified at deploy time. The recommended context root is /wmqfte.

"HTTP headers and HTML form fields for using the Web Gateway" on page 867
You can customize a request to create or retrieve a resource by using HTTP headers or HTML form fields. Each parameter maps to a property or function of WebSphere MQ Managed File Transfer.

"File space query response formats" on page 882
When you request a list of some or all of the files in a file space from the WebSphere MQ Managed File Transfer Service Web Gateway the response is returned in either JSON or XML format, depending on what you have specified using the Accept: header.

Example: Listing a specific subset of the files in a file space:

You can query the contents of a file space by submitting an HTTP request containing a query to the WebSphere MQ Managed File Transfer Web Gateway. The Web Gateway returns a response in XML or JSON format describing only those files in the filespace that match the query.

About this task

You can append a query to your HTTP request that requests information about the files in a file space that match the query. You can query files by their associated details, including the originating user, the transfer start time, the transfer end time, and the transfer ID of the transfer that sent the file to the file space. You can specify the number of results to return.

A successful request returns an HTTP status code of 200 and a payload that describes the files that match the query. You can request that the details of the files are returned in either XML or JSON format. You can write a web application to parse the content of the response and display it in an appropriate format to a web user.

The following steps describe how to submit a request. In this example, the server hosting the WebSphere MQ Managed File Transfer Web Gateway is example.com. The user requesting the information is authorized to access the file space that is being queried. The query requests information that is returned in JSON format, specified by the accept=json query. The query requests a list of files that fulfill the following criteria:

- The file are in the file space james.
- The files were sent to the file space by the user bob, specified by the originatoruser=bob query.
- The files were sent to the file space after 13:00 (UTC) on 26 August 2010, specified by the startafter=2010-08-26T13:00 query.

Procedure

1. Create an HTTP request with the following format:
   ```
   GET HTTP/1.1 /filespace/james/?originatoruser=bob&startafter=2010-08-26T13:00&accept=json
   Host: example.com
   User-Agent: mozilla
   ```

2. Submit the request to the Web Gateway. The Web Gateway returns an HTTP response with the following format. In this example, only one file matches the query.
   ```
   {
       "fileSpaces" : {
           "fileSpace" : {
               "name" : "james",
               "size" : "1",
               "file" : {
                   "transferLink" : "\/wmqfte\/transfer\/
                     414d51205745422e4654452020202020c1a1a34b03720120",
                   "fileLink" : "\/wmqfte\/filespace\/james\/
                     414d51205745422e4654452020202020c1a1a34b03720120\/wibble",
                   "name" : "\/tmp\/bobs_file.zip",
                   "transferID" : "414d51205745422e4654452020202020c1a1a34b03720120",
                   "attribute-values" : {
                       "checksum-value" : "98611a272a27d373f92d73a08cf0d4f4",
                       "checksum-method" : "none",
                       "time" : "2010-08-26T14:13:02.000Z",
                       "file-size" : "259354303",
                       "mode" : "text"
                   }
               }
           }
       }
   }
   ```

Related reference:

"File space query response formats" on page 882
When you request a list of some or all of the files in a file space from the WebSphere MQ Managed File Transfer Service Web Gateway the response is returned in either JSON or XML format, depending on what you have specified using the `Accept:` header.

"Uniform Resource Identifier syntax for using the Web Gateway" on page 869
A WebSphere MQ Managed File Transfer Uniform Resource Identifier (URI) is distinguished from other WebSphere MQ URIs by the context root specified at deploy time. The recommended context root is `/wmqfte`.

"HTTP headers and HTML form fields for using the Web Gateway" on page 867
You can customize a request to create or retrieve a resource by using HTTP headers or HTML form fields. Each parameter maps to a property or function of WebSphere MQ Managed File Transfer.

Example: Retrieving a file from a filespace:

You can retrieve a file from a file space by submitting an HTTP request to the WebSphere MQ Managed File Transfer Web Gateway. The Web Gateway provides the ability to download a file using the HTTP protocol.

About this task

To download a file from a file space, you must be the owner of the file space or have the security role `wmqfte-admin`. A successful request returns an HTTP status code of 200 and the file.

The following steps describe how to submit a request. In this example, the server hosting the WebSphere MQ Managed File Transfer Web Gateway is `example.com`. The file that is downloaded is `Accounts.csv` and the transfer ID of the transfer that sent the file to the file space is `4142452b345f4d2e3c2a333d4ed3e4de43453bc2344a2020`. The name of the file space that contains the file is

john, and the user requesting the information is authorized to access this file space.

Procedure

1. Create an HTTP request with the following format:

```
GET HTTP/1.1 /filespace/john/4142452b345f4d2e3c2a333d4ed3e4de43453bc2344a2020/Accts.csv
Host: example.com
User-Agent: mozilla
```

2. Submit the request to the Web Gateway. The Web Gateway returns the file in the HTTP response. The following headers are set in the HTTP response:
   * `Content-Type: application/x-download`
   * `Content-MD5: 98611a272a27d373f92d73a08cf0d4f4`
   * `Content-Disposition: attachment; filename="Accts.csv"`
   * `Content-Length: 8786`

Related reference:

"Uniform Resource Identifier syntax for using the Web Gateway" on page 869
A WebSphere MQ Managed File Transfer Uniform Resource Identifier (URI) is distinguished from other WebSphere MQ URIs by the context root specified at deploy time. The recommended context root is /wmqfte.

"HTTP headers and HTML form fields for using the Web Gateway" on page 867
You can customize a request to create or retrieve a resource by using HTTP headers or HTML form fields. Each parameter maps to a property or function of WebSphere MQ Managed File Transfer.

Example: Deleting a file from a file space:

You can delete a file from your file space by submitting an HTTP request to the WebSphere MQ Managed File Transfer Web Gateway. If you set the header `x-fte-include-file-in-response` to `true`, the contents of the file are returned in the HTTP response from the Web Gateway.

About this task

A successful deletion request returns an HTTP status code of 200 and, if specified in the request, the contents of the deleted file. The request will fail if the user submitting the request is not the owner of the file space.

Note: The security role `wmqfte-admin` can delete a file from a file space, but cannot receive the contents of the deleted file. If a user with the security role `wmqfte-admin` attempts to delete a file and request the file contents, the request fails with a resource error. For more information, see "User roles for the Web Gateway" on page 113.

The following steps describe how to submit a request. In this example, the server hosting the WebSphere MQ Managed File Transfer Web Gateway is `example.com`. The name of the file space is `jack`, it contains a file `report.txt`, and the user requesting the file deletion is the owner of the file space. The transfer ID `414d5120514d5f676172657457686622020067732c4c20c25a03` is the hexadecimal ID of the transfer that put the file in the file space, and this ID is returned when you list the contents of a file space. For more information about the format of file space query responses, see "File space query response formats" on page 882.

The header `x-fte-include-file-in-response:true` specifies that the contents of `report.txt` is returned in the body of the response. If you do not specify the value of this header, it defaults to `false` and the file is deleted but its contents are not returned.

Procedure

1. Create an HTTP request with the following format:

```
DELETE HTTP/1.1 /filespace/jack/414d5120514d5f676172657746862202067732c4c20c25a03/report.txt
Host: example.com
User-Agent: mozilla
x-fte-include-file-in-response:true
```

2. Submit the request to the Web Gateway. The Web Gateway returns an HTTP response with the following format:

```
HTTP/1.1 200 OK
Server: WAS/6.0
Content-Length: 1762
Content-MD5: 9608f0d8cdcb804d185ab3cb959dba6f
Content-type: text/plain; charset=Cp1252
Content-Disposition: attachment; filename="report.txt"

Account No, Balance
123456, 100.00
234567, 1022.00
345678, 2801.00
456789, 16.75
```

Related reference:

"User roles for the Web Gateway" on page 113
WebSphere MQ Managed File Transfer has defined several different roles that control the actions a user can take.

"Uniform Resource Identifier syntax for using the Web Gateway" on page 869
A WebSphere MQ Managed File Transfer Uniform Resource Identifier (URI) is distinguished from other WebSphere MQ URIs by the context root specified at deploy time. The recommended context root is /wmqfte.

"HTTP headers and HTML form fields for using the Web Gateway" on page 867
You can customize a request to create or retrieve a resource by using HTTP headers or HTML form fields. Each parameter maps to a property or function of WebSphere MQ Managed File Transfer.

Example: Sending a file using an HTML form

You can send a single text file to a destination file system by submitting a request through the WebSphere MQ Managed File Transfer Web Gateway.

About this task

This task demonstrates how to use an HTML form to submit a file transfer request to the Web Gateway. Using an HTML form is an alternative to submitting an HTTP request, which is described in "Example: Sending a file using an HTTP request" on page 286.

The following example uses several optional HTML form fields. For more information on the use of HTML form fields, see "HTTP headers and HTML form fields for using the Web Gateway" on page 867.

Procedure

1. Create an HTML file that includes a form in the following format: V7.5.0.2

```
<form enctype="multipart/form-data"
      action="http://example.org/wmqfte/file/agent/AGENT1@QM1/webuploads"
      method="POST">
 <input type="HIDDEN" name="dest-exists-action" value="overwrite"/>
 <input type="HIDDEN" name="type" value="text"/>
 <input type="HIDDEN" name="jobname" value="TEST"/>
 <input type="HIDDEN" name="priority" value="1"/>
 <input type="HIDDEN" name="checksum" value="NONE"/>
 <input type="HIDDEN" name="metadata" value="fred=awesome,bob=cool"/>
 <input type="HIDDEN" name="metadata" value="lewis=fast,niall=slow"/>
 <input type="HIDDEN" name="postdest"
        value="[command=D:\postdest.cmd,type=executable,successrc=0]"/>
```

```
  <input type="HIDDEN" name="postdest-args" value="[fred]"/>
  File: <input type="FILE" name="file"/>
  <input type="submit" name="Upload" value="Upload" />
</form>
```

▶ V7.5.0.2 The dest-exists-action form field used in this example is new for Version 7.5.0.2. dest-exists-action replaces the action form field, which is deprecated for future releases, but is still supported for 7.5.0.2.

```
<form enctype="multipart/form-data"
      action="http://example.org/wmqfte/file/agent/AGENT1@QM1/webuploads"
      method="POST">
  <input type="HIDDEN" name="action" value="overwrite"/>
  <input type="HIDDEN" name="type" value="text"/>
  <input type="HIDDEN" name="jobname" value="TEST"/>
  <input type="HIDDEN" name="priority" value="1"/>
  <input type="HIDDEN" name="checksum" value="NONE"/>
  <input type="HIDDEN" name="metadata" value="fred=awesome,bob=cool"/>
  <input type="HIDDEN" name="metadata" value="lewis=fast,niall=slow"/>
  <input type="HIDDEN" name="postdest"
          value="[command=D:\postdest.cmd,type=executable,successrc=0]"/>
  <input type="HIDDEN" name="postdest-args" value="[fred]"/>
  File: <input type="FILE" name="file"/>
  <input type="submit" name="Upload" value="Upload" />
</form>
```

2. Open this HTML file in a web browser.
3. Enter a file name in the **File** field, or click **Browse** to navigate to it.
4. Click **Upload** to submit the upload request. The Web Gateway returns an HTTP response with the following format:

```
HTTP/1.1 200 OK
Server: WAS/6.0
Content-Length: 0
x-fte-id: 4d63c28ae6e72eb9c51cd812736acd4362ef5

<transfers>
 <submission id="4d63c28ae6e72eb9c51cd812736acd4362ef5">
 </submission>
</transfers>
```

The value of x-fte-id is the transfer ID. You can use this transfer ID in an HTTP request for information about the status of the transfer. For an example request, see the topic "Example: Viewing the status of a file transfer using an HTTP request" on page 288.

Administering the WebSphere MQ Managed File Transfer Service Web Gateway

You can create and delete file spaces and control the users that have access to individual file spaces.

The Web Gateway can be administered in the following ways:
* By using the Web Gateway administrative console
* By using the RESTful administration API and constructing HTTP requests manually

The examples in this section demonstrate how to create HTTP requests to administer Web Gateway artifacts. For more information about the format of these requests and the Web Gateway administration API, see "Web Gateway administration API reference" on page 885.

These examples are for users with administrative rights. If you are looking for examples of using the Web Gateway for users without administrative rights, for example to upload files or query the files in a file space, see the topic "Using the WebSphere MQ Managed File Transfer Service Web Gateway" on page 285.

Before configuring or using the Web Gateway, refer to "Scenarios for the Web Gateway" on page 278 and "How the Web Gateway fits into your WebSphere MQ Managed File Transfer topology" on page 280. These topics explain the purpose and components of the Web Gateway.

Related concepts:

"Web Gateway administrative console"
The Web Gateway administrative console, which is provided with WebSphere MQ Managed File Transfer, provides a graphical interface for you to use to administer file spaces and user mappings. If you have the security role `wmqfte-admin` you can use the administrative console to complete administrative tasks.

"Example HTTP flows for administration" on page 304
You can construct HTTP requests and submit them to the WebSphere MQ Managed File Transfer Web Gateway. These examples show you sample administration requests and the corresponding HTTP responses from the Web Gateway.

"File spaces" on page 313
A file space is a reserved area of file storage that is associated with a Web Gateway user. A file space has an allocated quota of storage. Access to the file space is restricted to users with authorization to read from it or write to it.

Related reference:

"Web Gateway administration API reference" on page 885
The WebSphere MQ Managed File Transfer Web Gateway defines a RESTful HTTP application programming interface (API) for administration of file transfer artifacts.

Web Gateway administrative console

The Web Gateway administrative console, which is provided with WebSphere MQ Managed File Transfer, provides a graphical interface for you to use to administer file spaces and user mappings. If you have the security role `wmqfte-admin` you can use the administrative console to complete administrative tasks.

Using the administrative console

When you have deployed the Web Gateway to your application server, you can access the administrative console by opening a web browser and typing `http://host:port/wmqfteconsole`. If you changed the context root from the default of **wmqfteconsole** when you deployed the Web Gateway, you must use that value instead of **wmqfteconsole**.

If you are using WebSphere Application Server Community Edition, you might see the following error: `ssl_error_no_cypher_overlap`. To fix this problem, change the value of the **sslProtocol** setting of the **TomcatWebSSLConnector** to SSL then restart the connector.

Tasks you can perform using the administrative console

You can use the Web Gateway administrative console to administer two types of resource: file spaces and user mappings. You can use the administrative console to perform the following tasks:

Create a file space

You can create a file space by clicking the **File spaces** tab, and then clicking **Add**.

Edit the properties of a file space

You can edit the properties of a file space by clicking the **File spaces** tab, and then clicking **Edit**. The properties that you can edit are: quota, authorized users, and unauthorized users.

Remove a file space

You can remove a file space by clicking the **File spaces** tab, and then clicking **Remove**. Ensure that no transfers are in progress to or from the file space before deleting the file space.

Check the integrity of all file spaces

You can check the integrity of all file spaces associated with the Web Gateway by clicking the **File spaces** tab, and then clicking **Check integrity**.

Map web user IDs to MQMD user IDs

You can map web user IDs to MQMD user IDs by clicking the **MQMD user ID** tab, and then clicking **Add**. If you do not specify a mapping between a web user and an MQMD user ID, the value specified by the `defaultMQMDUserID` parameter is used.

Example HTTP flows for administration

You can construct HTTP requests and submit them to the WebSphere MQ Managed File Transfer Web Gateway. These examples show you sample administration requests and the corresponding HTTP responses from the Web Gateway.

You can use various technologies to submit requests to, and interpret responses from, the Web Gateway. For example, you can write a web application. For information about the example web application which is included with the Web Gateway, see "Sample web page" on page 327.

If you want to communicate with the Web Gateway by using a web application, you can use either HTML forms or the Javascript XMLHttpRequest function. To upload a file, you must use an HTML form, because browsers prevent Javascript from accessing files from the local system, for security reasons. The form can be controlled and submitted by Javascript if you prefer. To request the status of a transfer, XMLHttpRequest is most likely to be appropriate, although other techniques are possible; loading content into an invisible iFrame element, for example.

You can also write a client application in a language such as Ruby or Perl to communicate with the Web Gateway API.

Example: Creating a file space:

Before a file can be transferred to a user file space, you must create a file space for that user. You can create a file space by using the WebSphere MQ Managed File Transfer Web Gateway.

About this task

Use the Web Gateway administration API to request the creation of a user file space. For more information about the format of a file space creation request, see "File space create or alter request format" on page 891. A successful request returns an HTTP status code of 200.

You must have either the `wmqfte-filespace-create` role or the `wmqfte-admin` role associated with your user account to create a file space. For more information about security roles for the Web Gateway, see "User roles for the Web Gateway" on page 113 and "Attempting to create a file space without the required authority" on page 388.

If you have the security role `wmqfte-admin`, you can also create a file space by using the administrative console. For more information, see "Web Gateway administrative console" on page 303.

The following steps describe how to submit a POST request to create a file space. In this example, the server hosting the Web Gateway is `example.com` and the HTTP request is submitted using a web browser that identifies itself as `mozilla`. The name of the file space and the name of the user who owns the file space is `andrew` and the file space can take up a maximum of 1,048,576 bytes on the file system. The user `bill` and any user whose user name matches the regular expression pattern `fte.*` are authorized to send files to the file space. The user `clive` is not authorized to access the user file space. You can use Java regular expressions to pattern-match either or both sets of the users in the `authorized` and `unauthorized` XML sections. For more information, see "Regular expressions used by WebSphere MQ Managed File Transfer" on page 685.

In the following example, one of the `agent-user` entries in the `authorized` section uses the regular expression `fte.*`. This regular expression matches any user names starting with `fte`. In the situation that you wanted to authorize all user names starting with `fte` apart from `fteuser`, you could add an additional `agent-user` entry with a value of `fteuser` in the `unauthorized` section. This element would take precedence over the `fte.*` regular expression, because `unauthorized` entries overrule `authorized` entries when they evaluate to the same value.

In the following example, one of the `agent-user` entries in the `authorized` section is the user name `accounts1`. One of the `agent-user` entries in the `unauthorized` section is the regular expression `accounts*`, this overrides the authorization given to the user name `accounts1`. All users that match the regular expression `accounts*`, including the user `accounts1`, are not authorized on this file space.

Procedure

1. Create an HTTP request with the following format:
```
POST HTTP/1.1 /admin/filespace/andrew
Host: example.com
User-Agent: mozilla
Content-Type: application/xml
Content-Length: 266

<?xml version="1.0" encoding="UTF-8"?>
<filespaces>
  <filespace>
    <quota bytes="1048576"/>
    <writers>
      <authorized>
        <agent-user>bill</agent-user>
        <agent-user>accounts1</agent-user>
        <agent-user>fte.*</agent-user>
      </authorized>
      <unauthorized>
        <agent-user>fteuser</agent-user>
        <agent-user>accounts*</agent-user>
      </unauthorized>
    </writers>
  </filespace>
</filespaces>
```

2. Submit the request to the Web Gateway. The Web Gateway returns an HTTP response with the following format:
```
HTTP/1.1 200 OK
Server: WAS/7.0
Content-Length: 0
```

   A file space called `andrew` now exists and files can be transferred to it. The users `andrew`, `bill`, and any user whose name begins with `fte`, except for the user `fteuser`, can transfer files to the file space. No users that match the regular expression `accounts*` can transfer files to the file space.

   For information about how to transfer files to a file space, see "Example: Transferring a file to a file space" on page 285.

   The request to create a file space is logged to the application server event log. For more information, see "File space administration logging format" on page 893.

   An invalid request returns an HTTP error code and a WebSphere MQ Managed File Transfer error message. To identify the cause of the error, see "Troubleshooting the Web Gateway" on page 380.

Example: Modifying file space configuration:

You can modify an existing file space by submitting an HTTP request to the WebSphere MQ Managed File Transfer Web Gateway. You can change the file space quota and the list of users who can access the file space if you have the necessary security role associated with your user account.

About this task

The WebSphere MQ Managed File Transfer roles `wmqfte-admin` and `wmqfte-filespace-modify` can change both the file space quota and the list of users who can access the file space. For more information about securing the Web Gateway, see "User roles for the Web Gateway" on page 113.

If you have the security role `wmqfte-admin`, you can also modify a file space by using the administrative console. For more information, see "Web Gateway administrative console" on page 303.

If you change a file space quota while file transfers to the file space are in progress, the transfers might succeed even if they cause the new quota value to be exceeded. Any file transfers that are started after the quota has been changed are successful only if they do not cause the new quota value to be exceeded.

The following examples show how to change the quota of the file space, add users to the list of people authorized to access the file space, and remove users from the list of people who are not authorized to access the file space. In this example, the server hosting the Web Gateway is `example.com`. The name of the file space, which has already been created, is `finlay`. The name of the file space is denoted by the final part of the URI used by the POST request.

For more information about the format of the XML request to modify a file space, see "File space create or alter request format" on page 891.

Procedure

1. If you want to add to or remove from the existing lists of users, use the `add` action or `remove` action on the `authorized` and `unauthorized` elements. For example, the following request adds two users to the authorized list and removes one user from the unauthorized user:

```
POST HTTP/1.1 /admin/filespace/finlay
Host: example.com
User-Agent: mozilla
Content-Type: application/xml

<?xml version="1.0" encoding="UTF-8"?>
<filespaces>
  <filespace>
    <quota bytes="100000000"/>
    <writers>
      <authorized action="add">
        <agent-user>jonathan</agent-user>
        <agent-user>lauren</agent-user>
      </authorized>
      <unauthorized action="remove">
        <agent-user>marley</agent-user>
      </unauthorized>
    </writers>
  </filespace>
</filespaces>
```

If you want to overwrite the current lists of users, rather than add to or remove from the existing lists, use the `overwrite` action on the `authorized` and `unauthorized` elements. For example, the following request overwrites the current authorized list:

```
POST HTTP/1.1 /admin/filespace/finlay
Host: example.org
User-Agent: mozilla
```

```
Content-Type: application/xml

<?xml version="1.0" encoding="UTF-8"?>
<filespaces>
  <filespace>
    <writers>
      <authorized action="overwrite">
        <agent-user>fte.*</agent-user>
        <agent-user>ella</agent-user>
        <agent-user>jonathan</agent-user>
        <agent-user>lauren</agent-user>
      </authorized>
    </writers>
  </filespace>
</filespaces>
```

You can use Java regular expressions to match multiple user names. For example, one of the `agent-user` entries in the previous example has the value `fte.*`, which will match any user with a name that starts with `fte`.

2. Submit the request to the Web Gateway. The Web Gateway returns an HTTP response with the following format:

```
HTTP/1.1 200 OK
Server: WAS/7.0
Content-Length: 0
```

The request to modify a file space is logged to the application server event log. For more information, see "File space administration logging format" on page 893.An invalid request returns an HTTP error code and a WMQFTE error message. To identify the cause of the error, see "Troubleshooting the Web Gateway" on page 380.

Example: Listing all file spaces:

You can list all file spaces by submitting an HTTP request to the WebSphere MQ Managed File Transfer Web Gateway. The Web Gateway returns a response in XML or JSON format that lists the names of the file spaces, the quota of each file space, and the users who are authorized and not authorized to write to each file space.

About this task

A successful request returns an HTTP status code of 200 and a payload that describes, at most, 100 file spaces.

In this example, the server hosting the Web Gateway is `example.com`. There are currently three file spaces, belonging to the users `richard`, `suzanne` and `hamilton`. There are no file transfers currently in progress into the file space `richard`. There is one transfer in progress into the file space `hamilton`, and two transfers into the file space `suzanne`. The user who is requesting the information is associated with the security role `wmqfte-admin`. The header `Accept: application/xml` specifies that the query returns the results in XML format.

Procedure

1. Create an HTTP request with the following format:

```
GET HTTP/1.1 /admin/filespace/
Host: example.com
User-Agent: mozilla
Accept: application/xml
```

2. Submit the request to the Web Gateway. The Web Gateway returns an HTTP response with the following format:

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: application/xml
```

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<filespaces xsi:noNamespaceSchemaLocation="FileSpaceInfo.xsd"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <filespace transfers="0" location="/mnt/gateway/richard" name="richard">
        <quota bytes="1048576"/>
        <writers>
            <authorized>
                <agent-user>charlene</agent-user>
                <agent-user>alan</agent-user>
            </authorized>
        </writers>
    </filespace>
    <filespace transfers="2" location="/mnt/gateway/suzanne" name="suzanne">
        <quota bytes="20489878"/>
        <writers>
            <authorized>
                <agent-user>charlene</agent-user>
                <agent-user>sammy</agent-user>
            </authorized>
            <unauthorized>
                <agent-user>arnold</agent-user>
                <agent-user>frank</agent-user>
            </unauthorized>
        </writers>
    </filespace>
    <filespace transfers="1" location="/mnt/gateway/hamilton" name="hamilton">
        <quota bytes="666999"/>
        <writers>
            <authorized>
                <agent-user>joseph</agent-user>
            </authorized>
            <unauthorized>
                <agent-user>junior</agent-user>
            </unauthorized>
        </writers>
    </filespace>
</filespaces>
```

Example: Checking the integrity of files in a file space:

You can check the integrity of the files in a file space by submitting an HTTP request to the WebSphere MQ Managed File Transfer Web Gateway. For example, if you are restoring a file system after data loss, you can check that the files in a file space exist in the correct location on the file system. The Web Gateway returns a response in XML or JSON format that lists the contents of a file space with an additional attribute to indicate the result of an integrity check on each file.

About this task

A successful request returns an HTTP status code of 200 and a payload that lists the first 100 files in the file space. You can request that the details of the files are returned in either XML or JSON format. You can write a web application to parse the content of the response and display it in an appropriate format to a web user. Only an administrator is authorized to list the files in a file space with the integrity-check attribute.

The following steps describe how to submit a request. In this example, the server hosting the WebSphere MQ Managed File Transfer Web Gateway is example.com. The name of the file space to list is john and it contains two files. The header Accept: application/xml specifies that the query returns the results in XML format. The header x-fte-check-integrity specifies that the query returns the results with the additional integrity check attribute included for each file.

Procedure

1. Create an HTTP request with the following format:

```
GET HTTP/1.1 /filespace/john
Host: example.com
User-Agent: mozilla
Accept: application/xml
x-fte-check-integrity: true
```

2. Submit the request to the Web Gateway. The Web Gateway returns an HTTP response with the following format:

```
<fileSpaces xsi:noNamespaceSchemaLocation="WebTransferStatus.xsd">
  <fileSpace size="2" name="john">
    <file fileLink="/wmqfte/filespace/john/
      414d51205745422e4654452020202020c1a1a34b03720120/ar5erh"
        transferLink="/wmqfte/transfer/
        414d51205745422e4654452020202020c1a1a34b03720120"
        transferID="414d51205745422e4654452020202020c1a1a34b03720120"
        name="/tmp/file1.zip"
        fsLocation="/filespaces/john/
        414d51205745422e4654452020202020c1a1a34b03720120/file-0">
      <attribute-values mode="text" time="2010-08-26T11:45:02.000Z"
file-size="259354303"
                        checksum-value="98611a272a27d373f92d73a08cf0d4f4"
                        checksum-method="none"
                        integrity-check-result="OK"/>
    </file>
    <file fileLink="/wmqfte/filespace/john/
      414d51205745422e4654452020202020c1a1a34b06520120/ar5erh"
        transferLink="/wmqfte/transfer/
         414d51205745422e4654452020202020c1a1a34b06520120"
        transferID="414d51205745422e4654452020202020c1a1a34b06520120"
        name="/tmp/file2.zip"
        fsLocation="/filespaces/john/
         414d51205745422e4654452020202020c1a1a34b06520120/file-0">
      <attribute-values mode="text" time="2010-08-26T12:15:02.260Z"
        file-size="259554303"
                        checksum-value="98611a272a27d37bf22d73a08cf0d4f4"
                          checksum-method="none"
                        integrity-check-result="MISSING-FILESYSTEM"/>
    </file>
  </fileSpace>
</fileSpaces>
```

Results

This example result indicates that the first file has passed the integrity check. The `integrity-check-result` attribute value of `OK` shows that the file exists in the Web Gateway database and that the matching file has been found on the file system. The second file has failed the integrity check. The `integrity-check-result` attribute value of `MISSING-FILESYSTEM` shows that the file exists in the Web Gateway database but that the file cannot be found on the file system in the location given by the `fsLocation` attribute. In this case it might be necessary for an administrator to delete the file from the file space, or restore the file space directory from a backup.

For the possible values of the `integrity-check-result` attribute, see "File space information response format" on page 889.

Example: Checking the integrity of all file spaces:

You can check the integrity of all file spaces by submitting an HTTP request to the WebSphere MQ Managed File Transfer Web Gateway. For example, if you are restoring a file system after data loss, you can check that the files in a file space exist in the correct location on the file system. The Web Gateway returns a response in XML or JSON format that lists the names of the file spaces, and an attribute to indicate whether the file space entry matches the files in the file system.

About this task

Use the Web Gateway administration API to request a list of all the file spaces that currently exist. A successful request returns an HTTP status code of 200 and a payload that describes at most 100 file spaces. In this example, the server hosting the WebSphere MQ Managed File Transfer Web Gateway is `example.com`. There are currently three file spaces, belonging to the users `richard`, `suzanne` and `hamilton`. The user who is requesting the information is associated with the security role `wmqfte-admin`. The header `Accept: application/xml` specifies that the query returns the results in XML format. The header `x-fte-check-integrity` specifies that every file space should be checked to ensure a matching directory exists on the file system.

Procedure

1. Create an HTTP request with the following format:

```
GET HTTP/1.1 /admin/filespace/
Host: example.com
User-Agent: mozilla
Accept: application/xml
x-fte-check-integrity: true
```

2. Submit the request to the Web Gateway. The Web Gateway returns an HTTP response with the following format:

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: application/xml

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<filespaces xsi:noNamespaceSchemaLocation="FileSpaceInfo.xsd"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <filespace transfers="0" location="/mnt/gateway/richard" name="richard"
        integrity-check-result="OK">
        <quota bytes="1048576"/>
        <writers>
            <authorized>
                <agent-user>charlene</agent-user>
                <agent-user>alan</agent-user>
            </authorized>
        </writers>
    </filespace>
    <filespace transfers="2" location="/mnt/gateway/suzanne" name="suzanne"
        integrity-check-result="MISSING-FILESYSTEM">
        <quota bytes="20489878"/>
        <writers>
            <authorized>
                <agent-user>charlene</agent-user>
                <agent-user>sammy</agent-user>
            </authorized>
            <unauthorized>
                <agent-user>arnold</agent-user>
                <agent-user>frank</agent-user>
            </unauthorized>
        </writers>
    </filespace>
    <filespace transfers="1" location="/mnt/gateway/hamilton" name="hamilton"
        integrity-check-result="OK">
        <quota bytes="666999"/>
        <writers>
            <authorized>
                <agent-user>joseph</agent-user>
            </authorized>
            <unauthorized>
                <agent-user>junior</agent-user>
            </unauthorized>
        </writers>
    </filespace>
</filespaces>
```

Results

This example result indicates that the first and third file spaces in the set of results have passed the integrity check. The `integrity-check-result` attribute value of `OK` shows that the file spaces exist in the Web Gateway database and that matching directories have been found on the file system. The second file space has failed the integrity check. The `integrity-check-result` attribute value of `MISSING-FILESYSTEM` shows that the file space exists in the Web Gateway database but that the directory indicated by the `location` attribute cannot be found on the file system. In this case it might be necessary for an administrator to delete the file space, or restore the file space root directory from a backup.

If you have the security role `wmqfte-admin`, you can also check the integrity of all file spaces by using the administrative console. For more information, see "Web Gateway administrative console" on page 303.

For the possible values of the integrity-check-result attribute, see "File space information response format" on page 889.

Related concepts:

"Web Gateway administrative console" on page 303
The Web Gateway administrative console, which is provided with WebSphere MQ Managed File Transfer, provides a graphical interface for you to use to administer file spaces and user mappings. If you have the security role `wmqfte-admin` you can use the administrative console to complete administrative tasks.

Related tasks:

"Example: Checking the integrity of files in a file space" on page 308
You can check the integrity of the files in a file space by submitting an HTTP request to the WebSphere MQ Managed File Transfer Web Gateway. For example, if you are restoring a file system after data loss, you can check that the files in a file space exist in the correct location on the file system. The Web Gateway returns a response in XML or JSON format that lists the contents of a file space with an additional attribute to indicate the result of an integrity check on each file.

Example: Deleting a file space:

You can delete an existing file space by submitting an HTTP request to the WebSphere MQ Managed File Transfer Web Gateway. The file space is not deleted if a file transfer is in progress into the file space.

About this task

To delete a file space, you must have the appropriate security role associated with your user account. Users associated with the WebSphere MQ Managed File Transfer roles `wmqfte-admin` and `wmqfte-filespace-delete` can delete file spaces. For more information about securing the Web Gateway, see "User roles for the Web Gateway" on page 113.

If you have the security role `wmqfte-admin`, you can also delete a file space by using the administrative console. For more information, see "Web Gateway administrative console" on page 303.

Successful deletion of a file space:
About this task

In this example, the server hosting the WebSphere MQ Managed File Transfer Web Gateway is `example.com`. The file space belongs to the user `richard`. There are no file transfers currently in progress into the file space `richard`. You can find out the number of transfers in progress to the file spaces in your Web Gateway environment by listing the file spaces. For more information, see "Example: Listing all file spaces" on page 307.

Procedure

1. To delete the file space `richard`, create an HTTP request with the following format:

```
DELETE HTTP/1.1 /admin/filespace/richard
Host: example.com
User-Agent: mozilla
```

2. Submit the request to the Web Gateway. The Web Gateway returns an HTTP response with the following format:

```
HTTP/1.1 200 OK
Server: WAS/7.0
```

The file space `richard` and any files it contains are deleted. The deletion of a file space is logged to the application server event log. For more information, see "File space administration logging format" on page 893.

Possible problems when deleting a file space:
About this task

In this example, the server hosting the WebSphere MQ Managed File Transfer Web Gateway is `example.com`. The file space belongs to the user `suzanne`. There are two transfers in progress into the file space `suzanne`. You can find out the number of transfers in progress to the file spaces in your Web Gateway environment by listing the file spaces. For more information, see "Example: Listing all file spaces" on page 307.

Procedure

1. To delete the file space `suzanne`, create an HTTP request with the following format:

```
DELETE HTTP/1.1 /admin/filespace/suzanne
Host: example.com
User-Agent: mozilla
```

2. Submit the request to the Web Gateway. This request fails because there are transfers in progress into the file space. You therefore receive the following response from the Web Gateway:

```
HTTP/1.1 409 Conflict
Server: WAS/7.0

BFGWI0060E: The file space 'suzanne' is currently in use, and cannot be deleted.
```

You must wait for the transfers to the file space to complete before you can delete the file space.

To identify the cause of any other errors you might receive, see "Troubleshooting the Web Gateway" on page 380.

Example: Mapping web user IDs to MQMD user IDs:

When you submit file uploads to the WebSphere MQ Managed File Transfer Web Gateway, the Web Gateway determines which WebSphere MQ Message Descriptor (MQMD) user ID to use for the transfer. You can define a set of mappings between web user ID and MQMD user ID by using the Web Gateway.

About this task

Submit an HTTP request to the Web Gateway, with XML in the body of the request that maps web user IDs to MQMD user IDs. For more information about the format of the XML, see "XML format for mapping web user ID to an MQMD user ID" on page 892. A successful request returns an HTTP status code of 200.

You must have the `wmqfte-admin` role associated with your user account to create a set of mappings. For more information about security roles for the Web Gateway, see "User roles for the Web Gateway" on page 113.

If you have the security role `wmqfte-admin`, you can also map web user IDs to MQMD user IDs by using the administrative console. For more information, see "Web Gateway administrative console" on page 303.

The following steps describe how to submit a POST request to create a set of mappings. In this example, the server hosting the Web Gateway is `example.com` and the HTTP request is submitted using a web browser that identifies itself as `mozilla`. The request contains information for two users who have the web user IDs `jim` and `rachel`.

Procedure

1. Create an HTTP request with the following format:

   ```
   POST HTTP/1.1 /admin/user
   Host: example.com
   User-Agent: mozilla
   Content-Type: application/xml

   <?xml version="1.0" encoding="UTF-8"?>
   <users>
     <user>
       <userID>jim</userID>
       <mqmdUserID>mqjim</mqmdUserID>
     </user>
     <user>
       <userID>rachel</userID>
       <mqmdUserID>mqrachel</mqmdUserID>
     </user>
   </users>
   ```

2. Submit the request to the Web Gateway. The Web Gateway returns an HTTP response with the following format:

   ```
   HTTP/1.1 200 OK
   Server: WAS/7.0
   Content-Length: 0
   ```

   An invalid request returns an HTTP error code and a WebSphere MQ Managed File Transfer error message. To identify the cause of the error, see "Troubleshooting the Web Gateway" on page 380.

Results

When one of the users `jim` or `rachel` submits a file upload request through the Web Gateway, the appropriate MQMD user ID, `mqjim` or `mqrachel`, is used for the transfer. If a user who does not have an MQMD user ID defined submits a file upload request, the value of the **defaultMQMDUserID** parameter is used. In this situation, if this parameter was not defined during Web Gateway deployment, the transfer fails. For more information, see "Deploying the Web Gateway with WebSphere Application Server Version 7.0" on page 184 and "Preparing to deploy the Web Gateway with WebSphere Application Server Community Edition" on page 167.

File spaces

A file space is a reserved area of file storage that is associated with a Web Gateway user. A file space has an allocated quota of storage. Access to the file space is restricted to users with authorization to read from it or write to it.

You can send files from an agent to a user's file space. The files are stored in the file space and can be downloaded using an HTTP client that submits a request to the Web Gateway API. File spaces can be used to make files available to users who do not have access to a system hosting an agent. Transfers into a file space and downloads from a file space are logged in the same way as a normal file transfer.

You do not need a file space to upload a file to an MQMFT agent using the Web Gateway. If you want to make a file available for a user to collect using an HTTP client, you do need to create a file space. For

more information about the behavior of file uploads and downloads using the Web Gateway, see "Scenarios for the Web Gateway" on page 278.

Example: Creating a file space

Before a file can be transferred to a user file space, you must create a file space for that user. You can create a file space by using the WebSphere MQ Managed File Transfer Web Gateway.

About this task

Use the Web Gateway administration API to request the creation of a user file space. For more information about the format of a file space creation request, see "File space create or alter request format" on page 891. A successful request returns an HTTP status code of 200.

You must have either the `wmqfte-filespace-create` role or the `wmqfte-admin` role associated with your user account to create a file space. For more information about security roles for the Web Gateway, see "User roles for the Web Gateway" on page 113 and "Attempting to create a file space without the required authority" on page 388.

If you have the security role `wmqfte-admin`, you can also create a file space by using the administrative console. For more information, see "Web Gateway administrative console" on page 303.

The following steps describe how to submit a POST request to create a file space. In this example, the server hosting the Web Gateway is `example.com` and the HTTP request is submitted using a web browser that identifies itself as `mozilla`. The name of the file space and the name of the user who owns the file space is `andrew` and the file space can take up a maximum of 1,048,576 bytes on the file system. The user `bill` and any user whose user name matches the regular expression pattern `fte.*` are authorized to send files to the file space. The user `clive` is not authorized to access the user file space. You can use Java regular expressions to pattern-match either or both sets of the users in the `authorized` and `unauthorized` XML sections. For more information, see "Regular expressions used by WebSphere MQ Managed File Transfer" on page 685.

In the following example, one of the `agent-user` entries in the `authorized` section uses the regular expression `fte.*`. This regular expression matches any user names starting with `fte`. In the situation that you wanted to authorize all user names starting with `fte` apart from `fteuser`, you could add an additional `agent-user` entry with a value of `fteuser` in the `unauthorized` section. This element would take precedence over the `fte.*` regular expression, because `unauthorized` entries overrule `authorized` entries when they evaluate to the same value.

In the following example, one of the `agent-user` entries in the `authorized` section is the user name `accounts1`. One of the `agent-user` entries in the `unauthorized` section is the regular expression `accounts*`, this overrides the authorization given to the user name `accounts1`. All users that match the regular expression `accounts*`, including the user `accounts1`, are not authorized on this file space.

Procedure

1. Create an HTTP request with the following format:

```
POST HTTP/1.1 /admin/filespace/andrew
Host: example.com
User-Agent: mozilla
Content-Type: application/xml
Content-Length: 266

<?xml version="1.0" encoding="UTF-8"?>
<filespaces>
  <filespace>
    <quota bytes="1048576"/>
    <writers>
      <authorized>
        <agent-user>bill</agent-user>
```

```
        <agent-user>accounts1</agent-user>
        <agent-user>fte.*</agent-user>
      </authorized>
      <unauthorized>
        <agent-user>fteuser</agent-user>
        <agent-user>accounts*</agent-user>
      </unauthorized>
    </writers>
  </filespace>
</filespaces>
```

2. Submit the request to the Web Gateway. The Web Gateway returns an HTTP response with the
   following format:

```
HTTP/1.1 200 OK
Server: WAS/7.0
Content-Length: 0
```

A file space called andrew now exists and files can be transferred to it. The users andrew, bill, and
any user whose name begins with fte, except for the user fteuser, can transfer files to the file space.
No users that match the regular expression accounts* can transfer files to the file space.

For information about how to transfer files to a file space, see "Example: Transferring a file to a file
space" on page 285.

The request to create a file space is logged to the application server event log. For more information,
see "File space administration logging format" on page 893.

An invalid request returns an HTTP error code and a WebSphere MQ Managed File Transfer error
message. To identify the cause of the error, see "Troubleshooting the Web Gateway" on page 380.

Example: Deleting a file space

You can delete an existing file space by submitting an HTTP request to the WebSphere MQ Managed File
Transfer Web Gateway. The file space is not deleted if a file transfer is in progress into the file space.

About this task

To delete a file space, you must have the appropriate security role associated with your user account.
Users associated with the WebSphere MQ Managed File Transfer roles wmqfte-admin and
wmqfte-filespace-delete can delete file spaces. For more information about securing the Web Gateway,
see "User roles for the Web Gateway" on page 113.

If you have the security role wmqfte-admin, you can also delete a file space by using the administrative
console. For more information, see "Web Gateway administrative console" on page 303.

Successful deletion of a file space:
About this task

In this example, the server hosting the WebSphere MQ Managed File Transfer Web Gateway is
example.com. The file space belongs to the user richard. There are no file transfers currently in progress
into the file space richard. You can find out the number of transfers in progress to the file spaces in your
Web Gateway environment by listing the file spaces. For more information, see "Example: Listing all file
spaces" on page 307.

Procedure

1. To delete the file space richard, create an HTTP request with the following format:

```
DELETE HTTP/1.1 /admin/filespace/richard
Host: example.com
User-Agent: mozilla
```

2. Submit the request to the Web Gateway. The Web Gateway returns an HTTP response with the
   following format:

```
HTTP/1.1 200 OK
Server: WAS/7.0
```

The file space `richard` and any files it contains are deleted. The deletion of a file space is logged to the application server event log. For more information, see "File space administration logging format" on page 893.

Possible problems when deleting a file space:
About this task

In this example, the server hosting the WebSphere MQ Managed File Transfer Web Gateway is `example.com`. The file space belongs to the user `suzanne`. There are two transfers in progress into the file space `suzanne`. You can find out the number of transfers in progress to the file spaces in your Web Gateway environment by listing the file spaces. For more information, see "Example: Listing all file spaces" on page 307.

Procedure

1. To delete the file space `suzanne`, create an HTTP request with the following format:
   ```
   DELETE HTTP/1.1 /admin/filespace/suzanne
   Host: example.com
   User-Agent: mozilla
   ```
2. Submit the request to the Web Gateway. This request fails because there are transfers in progress into the file space. You therefore receive the following response from the Web Gateway:
   ```
   HTTP/1.1 409 Conflict
   Server: WAS/7.0

   BFGWI0060E: The file space 'suzanne' is currently in use, and cannot be deleted.
   ```

   You must wait for the transfers to the file space to complete before you can delete the file space.

   To identify the cause of any other errors you might receive, see "Troubleshooting the Web Gateway" on page 380.

Example: Modifying file space configuration

You can modify an existing file space by submitting an HTTP request to the WebSphere MQ Managed File Transfer Web Gateway. You can change the file space quota and the list of users who can access the file space if you have the necessary security role associated with your user account.

About this task

The WebSphere MQ Managed File Transfer roles `wmqfte-admin` and `wmqfte-filespace-modify` can change both the file space quota and the list of users who can access the file space. For more information about securing the Web Gateway, see "User roles for the Web Gateway" on page 113.

If you have the security role `wmqfte-admin`, you can also modify a file space by using the administrative console. For more information, see "Web Gateway administrative console" on page 303.

If you change a file space quota while file transfers to the file space are in progress, the transfers might succeed even if they cause the new quota value to be exceeded. Any file transfers that are started after the quota has been changed are successful only if they do not cause the new quota value to be exceeded.

The following examples show how to change the quota of the file space, add users to the list of people authorized to access the file space, and remove users from the list of people who are not authorized to access the file space. In this example, the server hosting the Web Gateway is `example.com`. The name of the file space, which has already been created, is `finlay`. The name of the file space is denoted by the final part of the URI used by the POST request.

For more information about the format of the XML request to modify a file space, see "File space create or alter request format" on page 891.

Procedure

1. If you want to add to or remove from the existing lists of users, use the `add` action or `remove` action on the `authorized` and `unauthorized` elements. For example, the following request adds two users to the authorized list and removes one user from the unauthorized user:

```
POST HTTP/1.1 /admin/filespace/finlay
Host: example.com
User-Agent: mozilla
Content-Type: application/xml

<?xml version="1.0" encoding="UTF-8"?>
<filespaces>
  <filespace>
    <quota bytes="100000000"/>
    <writers>
      <authorized action="add">
        <agent-user>jonathan</agent-user>
        <agent-user>lauren</agent-user>
      </authorized>
      <unauthorized action="remove">
        <agent-user>marley</agent-user>
      </unauthorized>
    </writers>
  </filespace>
</filespaces>
```

If you want to overwrite the current lists of users, rather than add to or remove from the existing lists, use the `overwrite` action on the `authorized` and `unauthorized` elements. For example, the following request overwrites the current authorized list:

```
POST HTTP/1.1 /admin/filespace/finlay
Host: example.org
User-Agent: mozilla
Content-Type: application/xml

<?xml version="1.0" encoding="UTF-8"?>
<filespaces>
  <filespace>
    <writers>
      <authorized action="overwrite">
        <agent-user>fte.*</agent-user>
        <agent-user>ella</agent-user>
        <agent-user>jonathan</agent-user>
        <agent-user>lauren</agent-user>
      </authorized>
    </writers>
  </filespace>
</filespaces>
```

You can use Java regular expressions to match multiple user names. For example, one of the `agent-user` entries in the previous example has the value `fte.*`, which will match any user with a name that starts with `fte`.

2. Submit the request to the Web Gateway. The Web Gateway returns an HTTP response with the following format:

```
HTTP/1.1 200 OK
Server: WAS/7.0
Content-Length: 0
```

The request to modify a file space is logged to the application server event log. For more information, see "File space administration logging format" on page 893.An invalid request returns an HTTP error code and a WMQFTE error message. To identify the cause of the error, see "Troubleshooting the Web Gateway" on page 380.

Example: Transferring a file to a file space

Transfer a single file to a WebSphere MQ Managed File Transfer file space. You can specify a file space as the destination of a file transfer by using the **-du** parameter with the **fteCreateTransfer** command.

About this task

When transferring a file to a file space, the WebSphere MQ Managed File Transfer Web Gateway checks whether the transfer would cause the file space quota to be exceeded. If the quota would be exceeded, an error is produced and the file transfer fails. The Web Gateway administrator can increase the size of the file space quota by submitting an HTTP request. For an example request, see the topic "Example: Modifying file space configuration" on page 306.

The file space quota is checked before the transfer begins. If you are using more than one agent to transfer files to the same file space, or if the Web Gateway administrator reduces the file space quota while a file is being transferred to that file space, one or more transfers might succeed even though they cause the file space quota to be exceeded.

In this example, the source file is called /tmp/Accounts.csv and is located on the same system as the source agent, *AGENT_1*. The destination file space john, which belongs to the user *john*, is located on the same system as the agent *FS_AGENT*. The user requesting the transfer has write access to the file space john. The agent *FS_AGENT* uses the queue manager *FS_QM*.

Procedure

Type the following command:

```
fteCreateTransfer -sa AGENT_1 -da FS_AGENT -dm FS_QM -du john /tmp/Accounts.csv
```

The file /tmp/Accounts.csv is transferred to the file space john. The user *john* can download this file from the file space when it is required.

Related concepts:

"File spaces" on page 313
A file space is a reserved area of file storage that is associated with a Web Gateway user. A file space has an allocated quota of storage. Access to the file space is restricted to users with authorization to read from it or write to it.

Related tasks:

"Starting a new file transfer" on page 205
You can start a new file transfer from the WebSphere MQ Explorer or from the command line and you can choose to transfer either a single file or multiple files in a group.

Example: Listing all files in a file space

You can list the contents of a file space by submitting an HTTP request to the WebSphere MQ Managed File Transfer Web Gateway. The Web Gateway returns a response in XML or JSON format that lists the contents of a file space. You are authorized to list the contents of a file space if you are the owner of the file space or you have the security role wmqfte-admin.

About this task

A successful request returns an HTTP status code of 200 and a payload that lists the first 100 files in the file space. This response is returned in either XML (the default) or JSON format dependent on the 'Accept' header specified in the request.

The following steps describe how to submit a request. In this example, the server hosting the WebSphere MQ Managed File Transfer Web Gateway is example.com and the HTTP request is submitted using a web browser which identifies itself as mozilla. The name of the file space to list is 'john' and it contains two files. The header 'Accept: application/xml' specifies that the Web Gateway should return the results in

XML format. For more information about the formats that are returned by a file space list request, see
"File space query response formats" on page 882.

Procedure

1. Create an HTTP request with the following format:

   ```
   GET HTTP/1.1 /filespace/john
   Host: example.com
   User-Agent: mozilla
   Accept: application/xml
   ```

2. Submit the request to the Web Gateway.

Results

The Web Gateway returns an HTTP response with the following format:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<fileSpaces xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:noNamespaceSchemaLocation="WebFileSpaceList.xsd">
  <fileSpace size="2" name="john">
    <file fileLink="/wmqfte/filespace/john/
      414d51205745422e4654452020202020c1a1a34b03720120/filename"
        fsLocation="/var/ibm/WMQFTE/web/fte/transfer/
         414d51205745422e4654452020202020c1a1a34b03720120/file-0"
        transferLink="/wmqfte/transfer/414d51205745422e4654452020202020c1a1a34b03720120"
        transferID="414d51205745422e4654452020202020c1a1a34b03720120">
      <attribute-values mode="text" created="2010-08-26T11:45:02.000Z" size="259354303"
                        checksum-value="98611a272a27d373f92d73a08cf0d4f4" checksum-method="MD5"/>
    </file>
    <file fileLink="/wmqfte/filespace/john/
      414d51205745422e4654452020202020c1a1a34b06520120/filename"
        fsLocation="/var/ibm/WMQFTE/web/fte/transfer/
         414d51205745422e4654452020202020c1a1a34b06520120/file-0"
        transferLink="/wmqfte/transfer/414d51205745422e4654452020202020c1a1a34b06520120"
        transferID="414d51205745422e4654452020202020c1a1a34b06520120">
      <attribute-values mode="text" created="2010-08-26T12:15:02.260Z" size="259554303"
                        checksum-value="98611a272a27d37bf22d73a08cf0d4f4" checksum-method="MD5"/>
    </file>
  </fileSpace>
</fileSpaces>
```

Related reference:

"Uniform Resource Identifier syntax for using the Web Gateway" on page 869
A WebSphere MQ Managed File Transfer Uniform Resource Identifier (URI) is distinguished from other
WebSphere MQ URIs by the context root specified at deploy time. The recommended context root is
/wmqfte.

"HTTP headers and HTML form fields for using the Web Gateway" on page 867
You can customize a request to create or retrieve a resource by using HTTP headers or HTML form fields.
Each parameter maps to a property or function of WebSphere MQ Managed File Transfer.

"File space query response formats" on page 882
When you request a list of some or all of the files in a file space from the WebSphere MQ Managed File
Transfer Service Web Gateway the response is returned in either JSON or XML format, depending on
what you have specified using the Accept: header.

Example: Checking the integrity of files in a file space

You can check the integrity of the files in a file space by submitting an HTTP request to the WebSphere
MQ Managed File Transfer Web Gateway. For example, if you are restoring a file system after data loss,
you can check that the files in a file space exist in the correct location on the file system. The Web
Gateway returns a response in XML or JSON format that lists the contents of a file space with an
additional attribute to indicate the result of an integrity check on each file.

About this task

A successful request returns an HTTP status code of 200 and a payload that lists the first 100 files in the file space. You can request that the details of the files are returned in either XML or JSON format. You can write a web application to parse the content of the response and display it in an appropriate format to a web user. Only an administrator is authorized to list the files in a file space with the integrity-check attribute.

The following steps describe how to submit a request. In this example, the server hosting the WebSphere MQ Managed File Transfer Web Gateway is `example.com`. The name of the file space to list is `john` and it contains two files. The header `Accept: application/xml` specifies that the query returns the results in XML format. The header `x-fte-check-integrity` specifies that the query returns the results with the additional integrity check attribute included for each file.

Procedure

1. Create an HTTP request with the following format:

```
GET HTTP/1.1 /filespace/john
Host: example.com
User-Agent: mozilla
Accept: application/xml
x-fte-check-integrity: true
```

2. Submit the request to the Web Gateway. The Web Gateway returns an HTTP response with the following format:

```
<fileSpaces xsi:noNamespaceSchemaLocation="WebTransferStatus.xsd">
  <fileSpace size="2" name="john">
    <file fileLink="/wmqfte/filespace/john/
      414d51205745422e4654452020202020c1a1a34b03720120/ar5erh"
        transferLink="/wmqfte/transfer/
        414d51205745422e4654452020202020c1a1a34b03720120"
        transferID="414d51205745422e4654452020202020c1a1a34b03720120"
        name="/tmp/file1.zip"
        fsLocation="/filespaces/john/
        414d51205745422e4654452020202020c1a1a34b03720120/file-0">
      <attribute-values mode="text" time="2010-08-26T11:45:02.000Z"
file-size="259354303"
                        checksum-value="98611a272a27d373f92d73a08cf0d4f4"
                        checksum-method="none"
                        integrity-check-result="OK"/>
    </file>
    <file fileLink="/wmqfte/filespace/john/
      414d51205745422e4654452020202020c1a1a34b06520120/ar5erh"
        transferLink="/wmqfte/transfer/
         414d51205745422e4654452020202020c1a1a34b06520120"
        transferID="414d51205745422e4654452020202020c1a1a34b06520120"
        name="/tmp/file2.zip"
        fsLocation="/filespaces/john/
         414d51205745422e4654452020202020c1a1a34b06520120/file-0">
      <attribute-values mode="text" time="2010-08-26T12:15:02.260Z"
        file-size="259554303"
                        checksum-value="98611a272a27d37bf22d73a08cf0d4f4"
                          checksum-method="none"
                        integrity-check-result="MISSING-FILESYSTEM"/>
    </file>
  </fileSpace>
</fileSpaces>
```

Results

This example result indicates that the first file has passed the integrity check. The `integrity-check-result` attribute value of `OK` shows that the file exists in the Web Gateway database and that the matching file has been found on the file system. The second file has failed the integrity check. The `integrity-check-result` attribute value of `MISSING-FILESYSTEM` shows that the file exists in the Web Gateway database but that the file cannot be found on the file system in the location given by the `fsLocation` attribute. In this case it might be necessary for an administrator to delete the file from the file space, or restore the file space directory from a backup.

For the possible values of the `integrity-check-result` attribute, see "File space information response format" on page 889.

Example: Listing a specific subset of the files in a file space

You can query the contents of a file space by submitting an HTTP request containing a query to the WebSphere MQ Managed File Transfer Web Gateway. The Web Gateway returns a response in XML or JSON format describing only those files in the filespace that match the query.

About this task

You can append a query to your HTTP request that requests information about the files in a file space that match the query. You can query files by their associated details, including the originating user, the transfer start time, the transfer end time, and the transfer ID of the transfer that sent the file to the file space. You can specify the number of results to return.

A successful request returns an HTTP status code of 200 and a payload that describes the files that match the query. You can request that the details of the files are returned in either XML or JSON format. You can write a web application to parse the content of the response and display it in an appropriate format to a web user.

The following steps describe how to submit a request. In this example, the server hosting the WebSphere MQ Managed File Transfer Web Gateway is `example.com`. The user requesting the information is authorized to access the file space that is being queried. The query requests information that is returned in JSON format, specified by the `accept=json` query. The query requests a list of files that fulfill the following criteria:

- The file are in the file space `james`.
- The files were sent to the file space by the user `bob`, specified by the `originatoruser=bob` query.
- The files were sent to the file space after 13:00 (UTC) on 26 August 2010, specified by the `startafter=2010-08-26T13:00` query.

Procedure

1. Create an HTTP request with the following format:
   ```
   GET HTTP/1.1 /filespace/james/?originatoruser=bob&startafter=2010-08-26T13:00&accept=json
   Host: example.com
   User-Agent: mozilla
   ```
2. Submit the request to the Web Gateway. The Web Gateway returns an HTTP response with the following format. In this example, only one file matches the query.
   ```
   {
      "fileSpaces" : {
         "fileSpace" : {
            "name" : "james",
            "size" : "1",
            "file" : {
               "transferLink" : "\/wmqfte\/transfer\/
                 414d51205745422e4654452020202020c1a1a34b03720120",
               "fileLink" : "\/wmqfte\/filespace\/james\/
                 414d51205745422e4654452020202020c1a1a34b03720120\/wibble",
               "name" : "\/tmp\/bobs_file.zip",
               "transferID" : "414d51205745422e4654452020202020c1a1a34b03720120",
               "attribute-values" : {
                  "checksum-value" : "98611a272a27d373f92d73a08cf0d4f4",
                  "checksum-method" : "none",
                  "time" : "2010-08-26T14:13:02.000Z",
                  "file-size" : "259354303",
                  "mode" : "text"
               }
   ```

```
              }
          }
      }
  }
```

Related reference:

"File space query response formats" on page 882
When you request a list of some or all of the files in a file space from the WebSphere MQ Managed File Transfer Service Web Gateway the response is returned in either JSON or XML format, depending on what you have specified using the `Accept:` header.

"Uniform Resource Identifier syntax for using the Web Gateway" on page 869
A WebSphere MQ Managed File Transfer Uniform Resource Identifier (URI) is distinguished from other WebSphere MQ URIs by the context root specified at deploy time. The recommended context root is `/wmqfte`.

"HTTP headers and HTML form fields for using the Web Gateway" on page 867
You can customize a request to create or retrieve a resource by using HTTP headers or HTML form fields. Each parameter maps to a property or function of WebSphere MQ Managed File Transfer.

Example: Retrieving a file from a filespace

You can retrieve a file from a file space by submitting an HTTP request to the WebSphere MQ Managed File Transfer Web Gateway. The Web Gateway provides the ability to download a file using the HTTP protocol.

About this task

To download a file from a file space, you must be the owner of the file space or have the security role `wmqfte-admin`. A successful request returns an HTTP status code of 200 and the file.

The following steps describe how to submit a request. In this example, the server hosting the WebSphere MQ Managed File Transfer Web Gateway is `example.com`. The file that is downloaded is `Accounts.csv` and the transfer ID of the transfer that sent the file to the file space is `4142452b345f4d2e3c2a333d4ed3e4de43453bc2344a2020`. The name of the file space that contains the file is `john`, and the user requesting the information is authorized to access this file space.

Procedure

1. Create an HTTP request with the following format:
   ```
   GET HTTP/1.1 /filespace/john/4142452b345f4d2e3c2a333d4ed3e4de43453bc2344a2020/Accts.csv
   Host: example.com
   User-Agent: mozilla
   ```

2. Submit the request to the Web Gateway. The Web Gateway returns the file in the HTTP response. The following headers are set in the HTTP response:
   - `Content-Type: application/x-download`
   - `Content-MD5: 98611a272a27d373f92d73a08cf0d4f4`
   - `Content-Disposition: attachment; filename="Accts.csv"`
   - `Content-Length: 8786`

Related reference:

"Uniform Resource Identifier syntax for using the Web Gateway" on page 869
A WebSphere MQ Managed File Transfer Uniform Resource Identifier (URI) is distinguished from other
WebSphere MQ URIs by the context root specified at deploy time. The recommended context root is
/wmqfte.

"HTTP headers and HTML form fields for using the Web Gateway" on page 867
You can customize a request to create or retrieve a resource by using HTTP headers or HTML form fields.
Each parameter maps to a property or function of WebSphere MQ Managed File Transfer.

Example: Deleting a file from a file space

You can delete a file from your file space by submitting an HTTP request to the WebSphere MQ Managed
File Transfer Web Gateway. If you set the header `x-fte-include-file-in-response` to `true`, the contents
of the file are returned in the HTTP response from the Web Gateway.

About this task

A successful deletion request returns an HTTP status code of 200 and, if specified in the request, the
contents of the deleted file. The request will fail if the user submitting the request is not the owner of the
file space.

Note: The security role `wmqfte-admin` can delete a file from a file space, but cannot receive the contents of
the deleted file. If a user with the security role `wmqfte-admin` attempts to delete a file and request the file
contents, the request fails with a resource error. For more information, see "User roles for the Web
Gateway" on page 113.

The following steps describe how to submit a request. In this example, the server hosting the WebSphere
MQ Managed File Transfer Web Gateway is `example.com`. The name of the file space is `jack`, it contains a
file `report.txt`, and the user requesting the file deletion is the owner of the file space. The transfer ID
`414d5120514d5f676172657574686622202067732c4c20c25a03` is the hexadecimal ID of the transfer that put the
file in the file space, and this ID is returned when you list the contents of a file space. For more
information about the format of file space query responses, see "File space query response formats" on
page 882.

The header `x-fte-include-file-in-response:true` specifies that the contents of `report.txt` is returned in
the body of the response. If you do not specify the value of this header, it defaults to `false` and the file is
deleted but its contents are not returned.

Procedure

1. Create an HTTP request with the following format:

```
DELETE HTTP/1.1 /filespace/jack/414d5120514d5f676172657574686622202067732c4c20c25a03/report.txt
Host: example.com
User-Agent: mozilla
x-fte-include-file-in-response:true
```

2. Submit the request to the Web Gateway. The Web Gateway returns an HTTP response with the
   following format:

```
HTTP/1.1 200 OK
Server: WAS/6.0
Content-Length: 1762
Content-MD5: 9608f0d8cdcb804d185ab3cb959dba6f
Content-type: text/plain; charset=Cp1252
Content-Disposition: attachment; filename="report.txt"

Account No, Balance
123456, 100.00
234567, 1022.00
345678, 2801.00
456789, 16.75
```

Related reference:

"User roles for the Web Gateway" on page 113
WebSphere MQ Managed File Transfer has defined several different roles that control the actions a user can take.

"Uniform Resource Identifier syntax for using the Web Gateway" on page 869
A WebSphere MQ Managed File Transfer Uniform Resource Identifier (URI) is distinguished from other WebSphere MQ URIs by the context root specified at deploy time. The recommended context root is `/wmqfte`.

"HTTP headers and HTML form fields for using the Web Gateway" on page 867
You can customize a request to create or retrieve a resource by using HTTP headers or HTML form fields. Each parameter maps to a property or function of WebSphere MQ Managed File Transfer.

Example: Listing all file spaces

You can list all file spaces by submitting an HTTP request to the WebSphere MQ Managed File Transfer Web Gateway. The Web Gateway returns a response in XML or JSON format that lists the names of the file spaces, the quota of each file space, and the users who are authorized and not authorized to write to each file space.

About this task

A successful request returns an HTTP status code of 200 and a payload that describes, at most, 100 file spaces.

In this example, the server hosting the Web Gateway is `example.com`. There are currently three file spaces, belonging to the users `richard`, `suzanne` and `hamilton`. There are no file transfers currently in progress into the file space `richard`. There is one transfer in progress into the file space `hamilton`, and two transfers into the file space `suzanne`. The user who is requesting the information is associated with the security role `wmqfte-admin`. The header `Accept: application/xml` specifies that the query returns the results in XML format.

Procedure

1. Create an HTTP request with the following format:

```
GET HTTP/1.1 /admin/filespace/
Host: example.com
User-Agent: mozilla
Accept: application/xml
```

2. Submit the request to the Web Gateway. The Web Gateway returns an HTTP response with the following format:

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: application/xml

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<filespaces xsi:noNamespaceSchemaLocation="FileSpaceInfo.xsd"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <filespace transfers="0" location="/mnt/gateway/richard" name="richard">
        <quota bytes="1048576"/>
        <writers>
            <authorized>
                <agent-user>charlene</agent-user>
                <agent-user>alan</agent-user>
            </authorized>
        </writers>
    </filespace>
    <filespace transfers="2" location="/mnt/gateway/suzanne" name="suzanne">
        <quota bytes="20489878"/>
        <writers>
```

```
                <authorized>
                    <agent-user>charlene</agent-user>
                    <agent-user>sammy</agent-user>
                </authorized>
                <unauthorized>
                    <agent-user>arnold</agent-user>
                    <agent-user>frank</agent-user>
                </unauthorized>
            </writers>
        </filespace>
        <filespace transfers="1" location="/mnt/gateway/hamilton" name="hamilton">
            <quota bytes="666999"/>
            <writers>
                <authorized>
                    <agent-user>joseph</agent-user>
                </authorized>
                <unauthorized>
                    <agent-user>junior</agent-user>
                </unauthorized>
            </writers>
        </filespace>
    </filespaces>
```

Example: Checking the integrity of all file spaces

You can check the integrity of all file spaces by submitting an HTTP request to the WebSphere MQ
Managed File Transfer Web Gateway. For example, if you are restoring a file system after data loss, you
can check that the files in a file space exist in the correct location on the file system. The Web Gateway
returns a response in XML or JSON format that lists the names of the file spaces, and an attribute to
indicate whether the file space entry matches the files in the file system.

About this task

Use the Web Gateway administration API to request a list of all the file spaces that currently exist. A
successful request returns an HTTP status code of 200 and a payload that describes at most 100 file
spaces. In this example, the server hosting the WebSphere MQ Managed File Transfer Web Gateway is
example.com. There are currently three file spaces, belonging to the users richard, suzanne and hamilton.
The user who is requesting the information is associated with the security role wmqfte-admin. The header
Accept: application/xml specifies that the query returns the results in XML format. The header
x-fte-check-integrity specifies that every file space should be checked to ensure a matching directory
exists on the file system.

Procedure

1. Create an HTTP request with the following format:

   ```
   GET HTTP/1.1 /admin/filespace/
   Host: example.com
   User-Agent: mozilla
   Accept: application/xml
   x-fte-check-integrity: true
   ```

2. Submit the request to the Web Gateway. The Web Gateway returns an HTTP response with the
   following format:

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: application/xml

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<filespaces xsi:noNamespaceSchemaLocation="FileSpaceInfo.xsd"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <filespace transfers="0" location="/mnt/gateway/richard" name="richard"
        integrity-check-result="OK">
        <quota bytes="1048576"/>
        <writers>
            <authorized>
```

```
            <agent-user>charlene</agent-user>
            <agent-user>alan</agent-user>
        </authorized>
    </writers>
</filespace>
<filespace transfers="2" location="/mnt/gateway/suzanne" name="suzanne"
    integrity-check-result="MISSING-FILESYSTEM">
    <quota bytes="20489878"/>
    <writers>
        <authorized>
            <agent-user>charlene</agent-user>
            <agent-user>sammy</agent-user>
        </authorized>
        <unauthorized>
            <agent-user>arnold</agent-user>
            <agent-user>frank</agent-user>
        </unauthorized>
    </writers>
</filespace>
<filespace transfers="1" location="/mnt/gateway/hamilton" name="hamilton"
    integrity-check-result="OK">
    <quota bytes="666999"/>
    <writers>
        <authorized>
            <agent-user>joseph</agent-user>
        </authorized>
        <unauthorized>
            <agent-user>junior</agent-user>
        </unauthorized>
    </writers>
</filespace>
</filespaces>
```

Results

This example result indicates that the first and third file spaces in the set of results have passed the
integrity check. The integrity-check-result attribute value of OK shows that the file spaces exist in the
Web Gateway database and that matching directories have been found on the file system. The second file
space has failed the integrity check. The integrity-check-result attribute value of MISSING-FILESYSTEM
shows that the file space exists in the Web Gateway database but that the directory indicated by the
location attribute cannot be found on the file system. In this case it might be necessary for an
administrator to delete the file space, or restore the file space root directory from a backup.

If you have the security role wmqfte-admin, you can also check the integrity of all file spaces by using the
administrative console. For more information, see "Web Gateway administrative console" on page 303.

For the possible values of the integrity-check-result attribute, see "File space information response
format" on page 889.

Related concepts:

"Web Gateway administrative console" on page 303
The Web Gateway administrative console, which is provided with WebSphere MQ Managed File Transfer,
provides a graphical interface for you to use to administer file spaces and user mappings. If you have the
security role wmqfte-admin you can use the administrative console to complete administrative tasks.

Related tasks:

"Example: Checking the integrity of files in a file space" on page 308
You can check the integrity of the files in a file space by submitting an HTTP request to the WebSphere
MQ Managed File Transfer Web Gateway. For example, if you are restoring a file system after data loss,
you can check that the files in a file space exist in the correct location on the file system. The Web
Gateway returns a response in XML or JSON format that lists the contents of a file space with an
additional attribute to indicate the result of an integrity check on each file.

Sample web page

WebSphere MQ Managed File Transfer Web Gateway provides a sample web page. This sample uses Web Gateway API functions to upload files, view the status of file transfers, view the contents of a file space and download files from a file space.

The sample application file name is `com.ibm.wmqfte.web.samples.war`. You can find this WAR file in the `MQ_INSTALLATION_PATH`/mqft/samples/web/servlet directory of the WebSphere MQ Managed File Transfer Server installation.

Before setting up the sample, you must have the Web Gateway application deployed and running in an application server. For instructions, see "Configuring the Web Gateway" on page 165.

Installing the sample

1. Deploy the sample application into an application server.

   If you deploy the sample into WebSphere Application Server Version 7.0:
   - Define a context root for the sample application. For example, if you use a context root of `/wmqftesamples` then the sample Web page is accessible through the URI `/wmqftesamples`.
   - You must configure the sample application with security roles and users. The sample application uses the same security realm that you defined for the Web Gateway. For more information, see "Deploying the Web Gateway with WebSphere Application Server Version 7.0" on page 184.

   If you deploy the sample into WebSphere Application Server Community Edition:
   - The application uses the context root defined in the `geronimo-web.xml` deployment plan that is located in the Web Gateway EAR file. This context root is `/wmqftesamples`.
   - You must configure the sample application with security roles and users. The sample application uses the same security realm that you defined for the Web Gateway. For more information, see "Defining a security realm" on page 172.

2. Open a web browser and type in the URI of the sample, based on the context root that you defined when deploying the sample. The URI of the sample is *host*:*port*/*context_root*.

   Note: The value of *port* depends on the application server that you are using. For example, for WebSphere Application Server Version 7.0, the default port used by applications is 9080.

3. Log in to the sample application using a user name and password that you configured when defining the security realm.

4. If you defined a context root for the Web Gateway other than the default value of `wmqfte`, use the **Settings** section in the sample application to specify the Web Gateway context root.

5. Use the sample application to upload files to the Web Gateway, view the files in your file space, download and delete files from your file space, and view the status of file transfers.

Related concepts:

"Scenarios for the Web Gateway" on page 278
Use the WebSphere MQ Managed File Transfer Web Gateway to transfer files to WebSphere MQ Managed File Transfer agents and retrieve the status of transfers using an HTTP client.

"How the Web Gateway fits into your WebSphere MQ Managed File Transfer topology" on page 280
Use the WebSphere MQ Managed File Transfer Service Web Gateway to transfer files to WebSphere MQ Managed File Transfer (MQMFT) agents and retrieve the status of transfers using an HTTP client.

"Web Gateway administrative console" on page 303
The Web Gateway administrative console, which is provided with WebSphere MQ Managed File Transfer, provides a graphical interface for you to use to administer file spaces and user mappings. If you have the security role `wmqfte-admin` you can use the administrative console to complete administrative tasks.

Related tasks:

The WebSphere MQ Managed File Transfer Web Gateway SupportPac must be deployed to an application server that is compatible with Java Platform, Enterprise Edition 5. The deployment process for different application servers varies. This section outlines the deployment process for two application servers.

Using Apache Ant with WebSphere MQ Managed File Transfer

WebSphere MQ Managed File Transfer provides tasks that you can use to integrate file transfer function into the Apache Ant tool.

You can use the **fteAnt** command to run Ant tasks in a WebSphere MQ Managed File Transfer environment that you have already configured. You can use file transfer Ant tasks from your Ant scripts to coordinate complex file transfer operations from an interpreted scripting language.

The **fteAnt** command is not applicable to the IBM 4690 environment. For more information on using WebSphere MQ Managed File Transfer in the IBM 4690 environment, see "Using WebSphere MQ Managed File Transfer in a retail environment" on page 35

For more information about Apache Ant, see the Apache Ant project web page: http://ant.apache.org/

Related concepts:

Using Ant scripts with WebSphere MQ Managed File Transfer allows you to coordinate complex file transfer operations from an interpreted scripting language.

Related reference:

WebSphere MQ Managed File Transfer provides a number of Ant tasks that you can use to access file transfer capabilities.

The **fteAnt** command runs Ant scripts in an environment that has WebSphere MQ Managed File Transfer Ant tasks available.

There are a number of sample Ant scripts provided with your installation of WebSphere MQ Managed File Transfer. These samples are located in the directory `MQ_INSTALLATION_PATH`/mqft/samples/fteant. Each sample script contains an `init` target, edit the properties set in the `init` target to run these scripts with your configuration.

Getting started using Ant scripts with WebSphere MQ Managed File Transfer

Using Ant scripts with WebSphere MQ Managed File Transfer allows you to coordinate complex file transfer operations from an interpreted scripting language.

Ant scripts

Ant scripts (or build files) are XML documents defining one or more targets. These targets contain task elements to run. WebSphere MQ Managed File Transfer provides tasks which you can use to integrate file transfer function into Apache Ant. To learn about Ant scripts, see the Apache Ant project web page: http://ant.apache.org/

Examples of Ant scripts that use WebSphere MQ Managed File Transfer tasks are provided with your product installation in the directory `MQ_INSTALLATION_PATH`/mqft/samples/fteant

On protocol bridge agents, Ant scripts are run on the protocol bridge agent system. These Ant scripts do not have direct access to the files on the FTP or SFTP server.

Namespace

A namespace is used to differentiate the file transfer Ant tasks from other Ant tasks that might share the same name. You define the namespace in the project tag of your Ant script.

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns:fte="antlib:com.ibm.wmqfte.ant.taskdefs" default="do_ping">

  <target name="do_ping">
    <fte:ping cmdqm="qm@localhost@1414@SYSTEM.DEF.SVRCONN" agent="agent1@qm1"
      rcproperty="ping.rc" timeout="15"/>
  </target>

</project>
```

The attribute `xmlns:fte="antlib:com.ibm.wmqfte.ant.taskdefs"` tells Ant to look for the definitions of tasks prefixed by `fte` in the library `com.ibm.wmqfte.ant.taskdefs`.

You do not need to use `fte` as your namespace prefix; you can use any value. The namespace prefix `fte` is used in all examples and sample Ant scripts.

Running Ant scripts

To run Ant scripts that contain the file transfer Ant tasks use the **fteAnt** command. For example:
```
fteAnt -file ant_script_location/ant_script_name
```

For more information, see "**fteAnt** (run Ant tasks in a WebSphere MQ Managed File Transfer environment)" on page 421.

Return codes

The file transfer Ant tasks return the same return codes as the WebSphere MQ Managed File Transfer commands. For more information, see "Return codes for WebSphere MQ Managed File Transfer" on page 373.

Related reference:

"Ant tasks provided by WebSphere MQ Managed File Transfer" on page 899
WebSphere MQ Managed File Transfer provides a number of Ant tasks that you can use to access file transfer capabilities.

"**fteAnt** (run Ant tasks in a WebSphere MQ Managed File Transfer environment)" on page 421
The **fteAnt** command runs Ant scripts in an environment that has WebSphere MQ Managed File Transfer Ant tasks available.

"Sample Ant tasks"
There are a number of sample Ant scripts provided with your installation of WebSphere MQ Managed File Transfer. These samples are located in the directory *MQ_INSTALLATION_PATH*/mqft/samples/fteant. Each sample script contains an `init` target, edit the properties set in the `init` target to run these scripts with your configuration.

Sample Ant tasks

There are a number of sample Ant scripts provided with your installation of WebSphere MQ Managed File Transfer. These samples are located in the directory *MQ_INSTALLATION_PATH*/mqft/samples/fteant. Each sample script contains an `init` target, edit the properties set in the `init` target to run these scripts with your configuration.

email

The email sample demonstrates how to use Ant tasks to transfer a file and send an email to a specified email address if the transfer fails. The script checks that the source and destination agents are active and able to process transfers by using the WebSphere MQ Managed File Transfer ping task. If both agents are active, the script uses the WebSphere MQ Managed File Transfer filecopy task to transfer a file between the source and destination agents, without deleting the original file. If the transfer fails the script sends an email containing information about the failure by using the standard Ant email task.

hub

The hub sample is made up of two scripts: `hubcopy.xml` and `hubprocess.xml.` The `hubcopy.xml` script shows how you can use Ant scripting to build 'hub and spoke' style topologies. In this sample, two files are transferred from agents running on spoke machines to an agent running on the hub machine. Both files are transferred at the same time, and when the transfers are complete the `hubprocess.xml` Ant script is run on the hub machine to process the files. If both files transfer correctly, the Ant script concatenates the contents of the files. If the files do not transfer correctly, the Ant script cleans up by deleting any file data that was transferred. For this example to work correctly, you must put the `hubprocess.xml` script on the command path of the hub agent. For more information about setting the command path of an agent, see commandPath.

timeout

The timeout sample demonstrates how to use Ant tasks to attempt a file transfer and to cancel the transfer if it takes longer than a specified timeout value. The script initiates a file transfer by using the WebSphere MQ Managed File Transfer filecopy task. The outcome of this transfer is deferred. The script uses the WebSphere MQ Managed File Transfer "fte:awaitoutcome" on page 899 task to wait a given number of seconds for the transfer to complete. If the transfer does not complete in the given time, the WebSphere MQ Managed File Transfer "fte:cancel" on page 902 task is used to cancel the file transfer.

zip

The zip sample is made up of two scripts: `zip.xml` and `zipfiles.xml`. The sample demonstrates how to use the `presrc` nested element inside the WebSphere MQ Managed File Transfer "fte:filemove" on page 906 task to run an Ant script before performing a file transfer move operation. The `zipfiles.xml` script called by the `presrc` nested element in the `zip.xml` script compresses the contents of a directory. The `zip.xml` script transfers the compressed file. This sample requires that the `zipfiles.xml` Ant script is present on the command path of the source agent. This is because the `zipfiles.xml` Ant script contains the target used to compress the contents of the directory at the source agent. For more information about setting the command path of an agent, see commandPath.

Customizing WebSphere MQ Managed File Transfer with user exit routines

You can customize the features of WebSphere MQ Managed File Transfer by using your own programs known as user exit routines.

WebSphere MQ Managed File Transfer provides points in the code where WebSphere MQ Managed File Transfer can pass control to a program that you have written (a user exit routine). These points are known as user exit points. WebSphere MQ Managed File Transfer can then resume control when your program has finished its work. You do not have to use any of the user exits, but they are useful if you want to extend and customize the function of your WebSphere MQ Managed File Transfer system to meet your specific requirements.

There are two points during file transfer processing where you can invoke a user exit at the source system and two points during file transfer processing where you can invoke a user exit at the destination system. The following table summarizes each of these user exit points and the Java interface that you must implement to use the exit points.

Table 9. Summary of source-side and destination-side exit points and Java interfaces

Exit point	Java interface to implement
Source-side exit points:	
Before the entire file transfer starts	SourceTransferStartExit.java
After the entire file transfer is complete	SourceTransferEndExit.java
Destination-side exit points:	
Before the entire file transfer starts	DestinationTransferStartExit.java
After the entire file transfer is complete	DestinationTransferEndExit.java

The user exits are invoked in the following order:

1. SourceTransferStartExit
2. DestinationTransferStartExit
3. DestinationTransferEndExit
4. SourceTransferEndExit

Changes made by the SourceTransferStartExit and DestinationTransferStartExit exits are propagated as input to subsequent exits. For example if the SourceTransferStartExit exit modifies the transfer metadata, the changes are reflected in the input transfer metadata to the other exits.

Building your user exit

The interfaces to build a user exit are contained in *MQ_INSTALL_DIRECTORY*/mqft/lib/ com.ibm.wmqfte.exitroutines.api.jar. You must include this .jar file in the class path when you build your exit. To run the exit, extract the exit as a .jar file and place this .jar file in a directory as described in the following section.

User exit locations

You can store your user exit routines in two possible locations:

* The `exits` directory. There is an exits directory under each agent directory. For example: `var\mqm\mqft\config\QM_JUPITER\agents\AGENT1\exits`
* You can set the exitClassPath property to specify an alternative location. If there are exit classes in both the `exits` directory and the class path set by exitClassPath, the classes in the `exits` directory take priority, which means that if there are classes in both locations with the same name, the classes in the `exits` directory take priority.

Configuring an agent to use user exits

There are four agent properties that can be set to specify the user exits that an agent invokes. These agent properties are `sourceTransferStartExitClasses`, `sourceTransferEndExitClasses`, `destinationTransferStartExitClasses`, and `destinationTransferEndExitClasses`. For information about how to use these properties, see "Agent properties for user exits" on page 927.

Running user exits on protocol bridge agents

If you run user exits on a protocol bridge agent, the exits have access to only the system where the bridge agent is located. The exits do not have direct access to the files on the FTP or SFTP server.

Running user exits on Connect:Direct bridge agents

You cannot run user exits on Connect:Direct bridge agents.

WebSphere MQ Managed File Transfer source and destination user exit routines

Directory separators

Directory separators in source file specifications are always represented using forward slash (/) characters, regardless of how you have specified directory separators in the **fteCreateTransfer** command or in the WebSphere MQ Explorer. You must take this into account when you write an exit. For example, if you want to check that the following source file exists: c:\a\b.txt and you have specified this source file using the **fteCreateTransfer** command or the WebSphere MQ Explorer, note the file name is actually stored as: c:/a/b.txt So if you search for the original string of c:\a\b.txt, you will not find a match.

Source side exit points

Before the entire file transfer starts
> This exit is called by the source agent when a transfer request is next in the list of pending transfers and the transfer is about to start.

> Example uses of this exit point are to send files in stages to a directory that the agent has read/write access to using an external command, or to rename the files on the destination system.

> Pass the following arguments to this exit:
> - Source agent name
> - Destination agent name
> - Environment metadata
> - Transfer metadata
> - File specifications (including file metadata)

> The data returned from this exit is as follows:
> - Updated transfer metadata. Entries can be added, modified, and deleted.
> - Updated list of file specifications, which consists of source file name and destination file name pairs. Entries can be added, modified, and deleted
> - Indicator that specifies whether to continue the transfer
> - String to insert to the Transfer Log.

> Implement the SourceTransferStartExit.java interface to call user exit code at this exit point.

After the entire file transfer is complete
> This exit is called by the source agent after the entire file transfer has completed.

> An example use of this exit point is to perform some completion tasks, such as sending an e-mail or a WebSphere MQ message to flag that the transfer has completed.

> Pass the following arguments to this exit:
> - Transfer exit result
> - Source agent name
> - Destination agent name
> - Environment metadata
> - Transfer metadata
> - File results

The data returned from this exit is as follows:

- Updated string to insert to the Transfer Log.

Implement the SourceTransferEndExit.java interface to call user exit code at this exit point.

Destination side exit points

Before the entire file transfer starts
    An example use of this exit point is to validate the permissions at the destination.

    Pass the following arguments to this exit:
    - Source agent name
    - Destination agent name
    - Environment metadata
    - Transfer metadata
    - File specifications

    The data returned from this exit is as follows:
    - Updated set of destination file names. Entries can be modified but not added or deleted.
    - Indicator that specifies whether to continue the transfer
    - String to insert into the Transfer Log.

    Implement the DestinationTransferStartExit.java interface to call user exit code at this exit point.

After the entire file transfer is complete
    An example use of this user exit is to start a batch process that uses the transferred files or to send an e-mail if the transfer has failed.

    Pass the following arguments to this exit:
    - Transfer exit result
    - Source agent name
    - Destination agent name
    - Environment metadata
    - Transfer metadata
    - File results

    The data returned from this exit is as follows:
    - Updated string to insert to the Transfer Log.

    Implement the DestinationTransferEndExit.java interface to call user exit code at this exit point.

Related concepts:

"Java interfaces for user exit routines" on page 930
Use the topics in this section for reference information about Java interfaces for user exit routines.

Related reference:

"Enabling remote debugging for user exits" on page 335
While you are developing your user exits, you might want to use a debugger to help locate problems in your code.
"Sample source transfer end user exit" on page 335

Using WebSphere MQ Managed File Transfer transfer I/O user exits

You can use WebSphere MQ Managed File Transfer transfer I/O user exits to configure custom code to perform the underlying file system I/O work for WebSphere MQ Managed File Transfer transfers.

Usually for MQMFT transfers, an agent selects from one of the built-in I/O providers to interact with the appropriate file systems for the transfer. Built-in I/O providers support the following types of file system:

- Regular UNIX-type and Windows-type file systems
- z/OS sequential and partitioned data sets (on z/OS only)
- IBM WebSphere MQ queues
- Remote FTP and SFTP protocol servers (for protocol bridge agents only)
- Remote Connect:Direct nodes (for Connect:Direct bridge agents only)

For file systems that are not supported, or where you require custom I/O behavior, you can write a transfer I/O user exit.

Transfer I/O user exits use the existing infrastructure for user exits. However, these transfer I/O user exits differ from other user exits because their function is accessed multiple times throughout the transfer for each file.

Use the agent property IOExitClasses (in the `agent.properties` file) to specify which I/O exit classes to load. Separate each exit class with a comma, for example:

```
IOExitClasses=testExits.TestExit1,testExits.testExit2
```

The Java interfaces for the transfer I/O user exits are as follows:

IOExit The main entry point used to determine if the I/O exit is used. This instance is responsible for making IOExitPath instances.

> You need specify only the IOExit I/O exit interface for the agent property IOExitClasses.

IOExitPath
> Represents an abstract interface; for example, a data container or wildcard representing a set of data containers. You cannot create a class instance that implements this interface. The interface allows the path to be examined and derived paths to be listed. The IOExitResourcePath and IOExitWildcardPath interfaces extend IOExitPath.

IOExitChannel
> Enables data to be read from or written to an IOExitPath resource.

IOExitRecordChannel
> Extends the IOExitChannel interface for record-oriented IOExitPath resources, which enables data to be read from or written to an IOExitPath resource in multiples of records.

IOExitLock
> Represents a lock on an IOExitPath resource for shared or exclusive access.

IOExitRecordResourcePath
> Extends the IOExitResourcePath interface to represent a data container for a record-oriented file; for example, a z/OS data set. You can use the interface to locate data and to create IOExitRecordChannel instances for read or write operations.

IOExitResourcePath
> Extends the IOExitPath interface to represent a data container; for example, a file or directory. You can use the interface to locate data. If the interface represents a directory, you can use the listPaths method to return a list of paths.

IOExitWildcardPath
> Extends the IOExitPath interface to represent a path that denotes a wildcard. You can use this interface to match multiple IOExitResourcePaths.

IOExitProperties
> Specifies properties that determine how WebSphere MQ Managed File Transfer handles IOExitPath for certain aspects of I/O. For example, whether to use intermediate files or whether to reread a resource from the beginning if a transfer is restarted.

Enabling remote debugging for user exits

While you are developing your user exits, you might want to use a debugger to help locate problems in your code.

Because exits run inside the Java virtual machine that runs the agent, you cannot use the direct debugging support that is typically included in an integrated development environment. However, you can enable remote debugging of the JVM and then connect a suitable remote debugger.

To enable remote debugging, use the standard JVM parameters **-Xdebug** and **-Xrunjdwp**. These properties are passed to the JVM that runs the agent by the FTE_JVM_PROPERTIES environment variable. For example, on UNIX the following commands start the agent and cause the JVM to listen for debugger connections on TCP port 8765.

```
export FTE_JVM_PROPERTIES="-Xdebug -Xrunjdwp:transport=dt_socket,server=y,address=8765"
fteStartAgent -F TEST_AGENT
```

The agent does not start until the debugger connects. Use the **set** command on Windows instead of the **export** command.

You can also use other communication methods between the debugger and JVM. For example, the JVM can open the connection to the debugger instead of vice versa, or you can use shared memory instead of TCP. See the Java Platform Debugger Architecture documentation for further details.

You must use the **-F** (foreground) parameter when you start the agent in remote debug mode.

Using the Eclipse debugger

The following steps apply to the remote debugging capability in the Eclipse development environment. You can also use other remote debuggers that are JPDA-compatible.

1. Click **Run** > **Open Debug Dialog** (or **Run** > **Debug Configurations** or **Run** > **Debug Dialog** depending on your version of Eclipse).
2. Double-click **Remote Java Application** in the list of configuration types to create a debug configuration.
3. Complete the configuration fields and save the debug configuration. If you have already started the agent JVM in debug mode, you can connect to the JVM now.

Sample source transfer end user exit

```
/*
 * A Sample Source Transfer End Exit that prints information about a transfer to standard output.
 * If the agent is run in the background the output will be sent to the agent's event log file. If
 * the agent is started in the foreground by specifying the -F parameter on the fteStartAgent
 * command the output will be sent to the console.
 *
 * To run the exit execute the following steps:
 *
 * Compile and build the exit into a jar file. You need the following in the class path:
 * {MQ_INSTALLATION_PATH}\mqft\lib\com.ibm.wmqfte.exitroutines.api.jar
 *
 * Put the jar in your agent's exits directory:
 * {MQ_DATA_PATH}\config\<coordQmgrName>\agents\<agentName>\exits\
 *
 * Update the agent's properties file:
 * {MQ_DATA_PATH}\config\<coordQmgrName>\agents\<agentName>\agent.properties
 * to include the following property:
 * sourceTransferEndExitClasses=[<packageName>.]SampleEndExit
 *
 * Restart agent to pick up the exit
```

```
 *
 * Send the agent a transfer request:
 * For example: fteCreateTransfer -sa myAgent -da YourAgent -df output.txt input.txt
 */

import java.util.List;
import java.util.Map;
import java.util.Iterator;

import com.ibm.wmqfte.exitroutine.api.SourceTransferEndExit;
import com.ibm.wmqfte.exitroutine.api.TransferExitResult;
import com.ibm.wmqfte.exitroutine.api.FileTransferResult;

public class SampleEndExit implements SourceTransferEndExit {

 public String onSourceTransferEnd(TransferExitResult transferExitResult,
   String sourceAgentName,
   String destinationAgentName,
   Map<String, String>environmentMetaData,
   Map<String, String>transferMetaData,
   List<FileTransferResult>fileResults) {

  System.out.println("Environment Meta Data: " + environmentMetaData);
  System.out.println("Transfer Meta Data:    " + transferMetaData);

  System.out.println("Source agent:          " +
    sourceAgentName);
  System.out.println("Destination agent:     " +
    destinationAgentName);

  if (fileResults.isEmpty()) {
   System.out.println("No files in the list");
   return "No files";
  }
  else {

   System.out.println( "File list: ");

   final Iterator<FileTransferResult> iterator = fileResults.iterator();

   while (iterator.hasNext()){
    final FileTransferResult thisFileSpec = iterator.next();
    System.out.println("Source file spec: "  +
      thisFileSpec.getSourceFileSpecification() +
      ", Destination file spec: " +
      thisFileSpec.getDestinationFileSpecification());
   }
  }
  return "Done";
 }
}
```

Sample protocol bridge credential user exit

For information about how to use this sample user exit, see "Mapping credentials for a file server using exit classes" on page 259

```
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.Enumeration;
import java.util.HashMap;
import java.util.Map;
import java.util.Properties;
import java.util.StringTokenizer;

import com.ibm.wmqfte.exitroutine.api.CredentialExitResult;
import com.ibm.wmqfte.exitroutine.api.CredentialExitResultCode;
import com.ibm.wmqfte.exitroutine.api.CredentialPassword;
import com.ibm.wmqfte.exitroutine.api.CredentialUserId;
import com.ibm.wmqfte.exitroutine.api.Credentials;
import com.ibm.wmqfte.exitroutine.api.ProtocolBridgeCredentialExit;

/**
 * A sample protocol bridge credential exit
 *
 * This exit reads a properties file that maps mq user ids to server user ids
 * and server passwords. The format of each entry in the properties file is:
 *
 * mqUserId=serverUserId,serverPassword
 *
 * The location of the properties file is taken from the protocol bridge agent
 * property protocolBridgeCredentialConfiguration.
 *
 * To install the sample exit compile the class and export to a jar file.
 * Place the jar file in the exits subdirectory of the agent data directory
 * of the protocol bridge agent on which the exit is to be installed.
 * In the agent.properties file of the protocol bridge agent set the
 * protocolBridgeCredentialExitClasses to SampleCredentialExit
 * Create a properties file that contains the mqUserId to serverUserId and
 * serverPassword mappings applicable to the agent. In the agent.properties
 * file of the protocol bridge agent set the protocolBridgeCredentialConfiguration
 * property to the absolute path name of this properties file.
 * To activate the changes stop and restart the protocol bridge agent.
 *
 * For further information on protocol bridge credential exits refer to
 * the WebSphere MQ Managed File Transfer documentation online at:
 * http://publib.boulder.ibm.com/infocenter/wmqfte/v7r0/
 */
public class SampleCredentialExit implements ProtocolBridgeCredentialExit {

// The map that holds mq user id to serverUserId and serverPassword mappings
final private Map<String,Credentials> credentialsMap = new HashMap<String, Credentials>();

/* (non-Javadoc)
  * @see com.ibm.wmqfte.exitroutine.api.ProtocolBridgeCredentialExit#initialize(java.util.Map)
  */
 public synchronized boolean initialize(Map<String, String> bridgeProperties) {

 // Flag to indicate whether the exit has been successfully initialized or not
 boolean initialisationResult = true;

 // Get the path of the mq user id mapping properties file
  final String propertiesFilePath = bridgeProperties.get("protocolBridgeCredentialConfiguration");

  if (propertiesFilePath == null || propertiesFilePath.length() == 0) {
   // The properties file path has not been specified. Output an error and return false
   System.err.println("Error initializing SampleCredentialExit.");
    System.err.println("The location of the mqUserID mapping properties file has not been specified in the
      protocolBridgeCredentialConfiguration property");
   initialisationResult = false;
  }

  if (initialisationResult) {

   // The Properties object that holds mq user id to serverUserId and serverPassword
    // mappings from the properties file
    final Properties mappingProperties = new Properties();
```

```
   // Open and load the properties from the properties file
   final File propertiesFile = new File (propertiesFilePath);
   FileInputStream inputStream = null;
   try {
    // Create a file input stream to the file
    inputStream = new FileInputStream(propertiesFile);

    // Load the properties from the file
    mappingProperties.load(inputStream);
   }
   catch (FileNotFoundException ex) {
    System.err.println("Error initializing SampleCredentialExit.");
     System.err.println("Unable to find the mqUserId mapping properties file: " + propertiesFilePath);
    initialisationResult = false;
   }
   catch (IOException ex) {
    System.err.println("Error initializing SampleCredentialExit.");
     System.err.println("Error loading the properties from the mqUserId mapping properties file: " + propertiesFilePath);
    initialisationResult = false;
   }
   finally {
    // Close the inputStream
    if (inputStream != null) {
     try {
      inputStream.close();
     }
     catch (IOException ex) {
      System.err.println("Error initializing SampleCredentialExit.");
       System.err.println("Error closing the mqUserId mapping properties file: " + propertiesFilePath);
      initialisationResult = false;
     }
    }
   }

   if (initialisationResult) {
    // Populate the map of mqUserId to server credentials from the properties
    final Enumeration<?> propertyNames = mappingProperties.propertyNames();
     while ( propertyNames.hasMoreElements()) {
     final Object name = propertyNames.nextElement();
     if (name instanceof String ) {
      final String mqUserId = ((String)name).trim();
       // Get the value and split into serverUserId and serverPassword
      final String value = mappingProperties.getProperty(mqUserId);
      final StringTokenizer valueTokenizer = new StringTokenizer(value, ",");
       String serverUserId = "";
       String serverPassword = "";
       if (valueTokenizer.hasMoreTokens()) {
        serverUserId = valueTokenizer.nextToken().trim();
       }
       if (valueTokenizer.hasMoreTokens()) {
        serverPassword = valueTokenizer.nextToken().trim();
       }
       // Create a Credential object from the serverUserId and serverPassword
       final Credentials credentials = new Credentials(new CredentialUserId(serverUserId), new CredentialPassword(serverPassword));
       // Insert the credentials into the map
       credentialsMap.put(mqUserId, credentials);
      }
     }

   }
   }

   return initialisationResult;
 }
 /* (non-Javadoc)
  * @see com.ibm.wmqfte.exitroutine.api.ProtocolBridgeCredentialExit#mapMQUserId(java.lang.String)
  */
 public synchronized CredentialExitResult mapMQUserId(String mqUserId) {
  CredentialExitResult result = null;
  // Attempt to get the server credentials for the given mq user id
  final Credentials credentials = credentialsMap.get(mqUserId.trim());
  if ( credentials == null) {
   // No entry has been found so return no mapping found with no credentials
   result = new CredentialExitResult(CredentialExitResultCode.NO_MAPPING_FOUND, null);
  }
  else {
   // Some credentials have been found so return success to the user along with the credentials
   result = new CredentialExitResult(CredentialExitResultCode.USER_SUCCESSFULLY_MAPPED, credentials);
```

```
    }
    return result;
  }
  /* (non-Javadoc)
   * @see com.ibm.wmqfte.exitroutine.api.ProtocolBridgeCredentialExit#shutdown(java.util.Map)
   */
  public void shutdown(Map<String, String> bridgeProperties) {
    // Nothing to do in this method because there are no resources that need to be released
  }
}
```

Sample protocol bridge properties user exit

For information about how to use this sample user exit, see "Looking up protocol file server properties
by using exit classes (ProtocolBridgePropertiesExit2)" on page 253

SamplePropertiesExit2.java

```java
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.HashMap;
import java.util.Map;
import java.util.Map.Entry;
import java.util.Properties;

import com.ibm.wmqfte.exitroutine.api.ProtocolBridgePropertiesExit2;
import com.ibm.wmqfte.exitroutine.api.ProtocolServerPropertyConstants;

/**
 * A sample protocol bridge properties exit. This exit reads a properties file
 * that contains properties for protocol servers.
 * <p>
 * The format of each entry in the properties file is:
 * {@literal <serverName>=<type>://<host>:<port>}
 * Ensure there is a default entry such as
 * {@literal default=<type>://<host>:<port>}
 * otherwise the agent will fail to start with a BFGBR0168 as it must have a
 * default server.
 * <p>
 * The location of the properties file is taken from the protocol bridge agent
 * property {@code protocolBridgePropertiesConfiguration}.
 * <p>
 * The methods {@code getCredentialLocation} returns the location of the associated
 * ProtocolBridgeCredentials.xml, this sample it is defined to be stored in a directory
 * defined by the environment variable CREDENTIALSHOME
 * <p>
 * To install the sample exit:
 * <ol>
 * <li>Compile the class and export to a jar file.
 * <li>Place the jar file in the {@code exits} subdirectory of the agent data directory
 * of the protocol bridge agent on which the exit is to be installed.
 * <li>In the {@code agent.properties} file of the protocol bridge agent
 * set the {@code protocolBridgePropertiesExitClasses} to
 * {@code SamplePropertiesExit2}.
 * <li>Create a properties file that contains the appropriate properties to specify the
 * required servers.
 * <li>In the {@code agent.properties} file of the protocol bridge agent
 * set the <code>protocolBridgePropertiesConfiguration</code> property to the
 * absolute path name of this properties file.
 * <li>To activate the changes stop and restart the protocol bridge agent.
 * </ol>
 * <p>
 * For further information on protocol bridge properties exits refer to the
 * WebSphere MQ Managed File Transfer documentation online at:
 * <p>
 * {@link http://publib.boulder.ibm.com/infocenter/wmqfte/v7r0/}
 */
public class SamplePropertiesExit2 implements ProtocolBridgePropertiesExit2 {

  /**
   * Helper class to encapsulate protocol server information.
   */
  private static class ServerInformation {
    private final String type;
```

```
 private final String host;
 private final int port;

 public ServerInformation(String url) {
  int index = url.indexOf("://");
  if (index == -1) throw new IllegalArgumentException("Invalid server URL: "+url);
  type = url.substring(0, index);

  int portIndex = url.indexOf(":", index+3);
  if (portIndex == -1) {
   host = url.substring(index+3);
   port = -1;
  } else {
   host = url.substring(index+3,portIndex);
   port = Integer.parseInt(url.substring(portIndex+1));
  }
 }

 public String getType() {
  return type;
 }

 public String getHost() {
  return host;
 }

 public int getPort() {
  return port;
 }
}

/** A {@code Map} that holds information for each configured protocol server */
final private Map<String, ServerInformation> servers = new HashMap<String, ServerInformation>();


/* (non-Javadoc)
 * @see com.ibm.wmqfte.exitroutine.api.ProtocolBridgePropertiesExit#getProtocolServerProperties(java.lang.String)
 */
public Properties getProtocolServerProperties(String protocolServerName) {
 // Attempt to get the protocol server information for the given protocol server name
 // If no name has been supplied then this implies the default.
 final ServerInformation info;
 if (protocolServerName == null || protocolServerName.length() == 0) {
  protocolServerName = "default";
 }
 info = servers.get(protocolServerName);

 // Build the return set of properties from the collected protocol server information, when available.
 // The properties set here is the minimal set of properties to be a valid set.
 final Properties result;
 if (info != null) {
  result = new Properties();
  result.setProperty(ProtocolServerPropertyConstants.SERVER_NAME, protocolServerName);
  result.setProperty(ProtocolServerPropertyConstants.SERVER_TYPE, info.getType());
  result.setProperty(ProtocolServerPropertyConstants.SERVER_HOST_NAME, info.getHost());
  if (info.getPort() != -1) result.setProperty(ProtocolServerPropertyConstants.SERVER_PORT_VALUE, ""+info.getPort());
  result.setProperty(ProtocolServerPropertyConstants.SERVER_PLATFORM, "UNIX");
  if (info.getType().toUpperCase().startsWith("FTP")) {  // FTP & FTPS
   result.setProperty(ProtocolServerPropertyConstants.SERVER_TIMEZONE, "Europe/London");
   result.setProperty(ProtocolServerPropertyConstants.SERVER_LOCALE, "en-GB");
  }
  result.setProperty(ProtocolServerPropertyConstants.SERVER_FILE_ENCODING, "UTF-8");
 } else {
  System.err.println("Error no default protocol file server entry has been supplied");
  result = null;
 }

 return result;
}

/* (non-Javadoc)
 * @see com.ibm.wmqfte.exitroutine.api.ProtocolBridgePropertiesExit#initialize(java.util.Map)
 */
public boolean initialize(Map<String, String> bridgeProperties) {
 // Flag to indicate whether the exit has been successfully initialized or not
 boolean initialisationResult = true;

 // Get the path of the properties file
 final String propertiesFilePath = bridgeProperties.get("protocolBridgePropertiesConfiguration");
```

```java
  if (propertiesFilePath == null || propertiesFilePath.length() == 0) {
   // The protocol server properties file path has not been specified. Output an error and return false
   System.err.println("Error initializing SamplePropertiesExit.");
   System.err.println("The location of the protocol server properties file has not been specified in the
       protocolBridgePropertiesConfiguration property");
   initialisationResult = false;
  }

  if (initialisationResult) {
   // The Properties object that holds protocol server information
   final Properties mappingProperties = new Properties();

   // Open and load the properties from the properties file
   final File propertiesFile = new File (propertiesFilePath);
   FileInputStream inputStream = null;
   try {
    // Create a file input stream to the file
    inputStream = new FileInputStream(propertiesFile);

    // Load the properties from the file
    mappingProperties.load(inputStream);
   } catch (final FileNotFoundException ex) {
    System.err.println("Error initializing SamplePropertiesExit.");
    System.err.println("Unable to find the protocol server properties file: " + propertiesFilePath);
    initialisationResult = false;
   } catch (final IOException ex) {
    System.err.println("Error initializing SamplePropertiesExit.");
    System.err.println("Error loading the properties from the protocol server properties file: " + propertiesFilePath);
    initialisationResult = false;
   } finally {
    // Close the inputStream
    if (inputStream != null) {
     try {
      inputStream.close();
     } catch (final IOException ex) {
      System.err.println("Error initializing SamplePropertiesExit.");
      System.err.println("Error closing the protocol server properties file: " + propertiesFilePath);
      initialisationResult = false;
     }
    }
   }

   if (initialisationResult) {
    // Populate the map of protocol servers from the properties
    for (Entry<Object, Object> entry : mappingProperties.entrySet()) {
     final String serverName = (String)entry.getKey();
     final ServerInformation info = new ServerInformation((String)entry.getValue());
     servers.put(serverName, info);
    }
   }
  }

  return initialisationResult;
 }

 /* (non-Javadoc)
  * @see com.ibm.wmqfte.exitroutine.api.ProtocolBridgePropertiesExit#shutdown(java.util.Map)
  */
 public void shutdown(Map<String, String> bridgeProperties) {
  // Nothing to do in this method because there are no resources that need to be released
 }

 /* (non-Javadoc)
  * @see com.ibm.wmqfte.exitroutine.api.ProtocolBridgePropertiesExit2#getCredentialLocation()
  */
 public String getCredentialLocation() {
  String envLocationPath;
  if (System.getProperty("os.name").toLowerCase().contains("win")) {
   // Windows style
   envLocationPath = "%CREDENTIALSHOME%\\ProtocolBridgeCredentials.xml";
  }
  else {
   // Unix style
   envLocationPath = "$CREDENTIALSHOME/ProtocolBridgeCredentials.xml";
  }
  return envLocationPath;
 }

}
```

Controlling WebSphere MQ Managed File Transfer by putting messages on the agent command queue

You can write an application that controls WebSphere MQ Managed File Transfer by putting messages on agent command queues.

You can put a message on the command queue of an agent to request that the agent performs one of the following actions:

- Create a file transfer
- Create a scheduled file transfer
- Cancel a file transfer
- Cancel a scheduled file transfer
- Call a command
- Create a monitor
- Delete a monitor
- Return a ping to indicate that the agent is active

To request that the agent performs one of these actions, the message must be in an XML format that complies with one of the following schema:

FileTransfer.xsd
> Messages in this format can be used to create a file transfer or scheduled file transfer, to call a command, or to cancel a file transfer or scheduled file transfer. For more information, see "File transfer request message format" on page 803.

Monitor.xsd
> Messages in this format can be used to create or delete a resource monitor. For more information, see "Monitor request message formats" on page 820.

PingAgent.xsd
> Messages in this format can be used to ping an agent to check that it is active. For more information, see "Ping agent request message format" on page 831.

The agent returns a reply to the request messages. The reply message is put to a reply queue that is defined in the request message. The reply message is in an XML format defined by the following schema:

Reply.xsd
> For more information, see "Reply message format" on page 832.

Troubleshooting WebSphere MQ Managed File Transfer

Use the following reference information to help you to diagnose errors in WebSphere MQ Managed File Transfer:

Related reference:

"Tracing WebSphere MQ Managed File Transfer commands"
You can trace any of the WebSphere MQ Managed File Transfer commands to help with problem determination from the command line.

General troubleshooting

Use the following reference information to help you to diagnose errors in WebSphere MQ Managed File Transfer:

Running trace on WebSphere MQ Managed File Transfer

You can trace WebSphere MQ Managed File Transfer by using the following methods:

- Dynamically change the current level of agent trace by using the fteSetAgentTraceLevel command.
- Dynamically change the current level of logger trace by using the fteSetLoggerTraceLevel command.
- Trace any of the **fte** commands by using the **-trace** parameter. For more information, see Tracing commands.
- Configure an agent to start with trace enabled by setting the trace properties in the `agent.properties` file. For more information, see Advanced agent properties.

Tracing WebSphere MQ Managed File Transfer commands

You can trace any of the WebSphere MQ Managed File Transfer commands to help with problem determination from the command line.

Purpose

Use the **-trace** parameter for any command to enable trace at a specified level. The trace files produced are located in your current working directory.

Because running trace can affect your performance significantly and can produce a large amount of trace data, run trace with care and only when necessary. Typically, enable trace only when asked to do so by your IBM service representative.

You can set further trace properties, for example trace file size and the number of trace files to keep, in the `agent.properties` file. These properties are described in Advanced agent properties.

Syntax

```
►►──fteCommandName──-trace──(classes=level)─────────────────────►◄
                            └─-tracePath──(directory path)─┘
```

Parameters

-trace (`classes=level`**)**
> Required. Level to set the trace and which classes to apply the trace to. Specify the following format:
>
> `classes=level`

For example:

```
com.ibm.wmqfte=all
```

which traces all WebSphere MQ Managed File Transfer classes.

Specify a colon-separated list of class specifications that you want the level of trace to apply to. If you do not specify this parameter, the trace level is applied to all agent classes.

If (*classes*) start with a plus sign (+), the list of trace classes following the plus sign are added to any existing trace classes currently being traced.

The valid trace level options are as follows and are listed in ascending order of trace file size and detail:

off   Switches the agent trace off but continues to write information to the log files. This is the default option.

flow   Captures data for trace points associated with processing flow in the agent.

moderate
> Captures a moderate amount of diagnostic information in the trace.

verbose
> Captures a verbose amount of diagnostic information in the trace.

all   Sets agent trace to run on all agent classes.

-tracePath (*directory path***)**
> Optional. Specify the directory that you want the trace to be written to. For example, c:\temp.
>
> This parameter is valid only when the **-trace** parameter is specified.

Example

In this example the trace level is set to all, meaning that all of the classes belonging to AGENT.NAME are traced for the **fteStartAgent** command:

Note: When the agent is started, the trace goes to <mft config>/logs<coordination qmgr>/agents/ <agent>

```
fteStartAgent -trace com.ibm.wmqfte=all -tracePath=/u/mft/trace AGENT.NAME
```

In this example the trace level is set to moderate for the com.ibm.wmqfte.common classes for the agent AGENT.NAME. A moderate amount of trace is captured for the **ftePingAgent** command:

```
ftePingAgent -trace com.ibm.wmqfte.common=moderate AGENT.NAME
```

In this example the trace level is set to moderate for the com.ibm.wmqfte.common classes for the agent AGENT.NAME, and the trace is written to the c\$user directory. A moderate amount of trace is captured for the **ftePingAgent** command:

```
ftePingAgent -trace com.ibm.wmqfte.common=moderate -tracePath c:\$user AGENT.NAME
```

Related reference:

"Troubleshooting WebSphere MQ Managed File Transfer" on page 343
Use the following reference information to help you to diagnose errors in WebSphere MQ Managed File Transfer:

fteSetAgentTraceLevel (set WebSphere MQ Managed File Transfer agent trace level)

Use the **fteSetAgentTraceLevel** command to modify the current trace level for an agent dynamically.

Purpose

Use this command to switch agent trace on and off or to change the level of agent trace that is set. When you use the **fteSetAgentTraceLevel** command, you do not have to shut down and restart an agent to modify the trace level. The trace files produced are located in *MQ_DATA_PATH*/mqft/logs/ *coordination_qmgr_name*/agents/*agent_name*/logs/trace%PID%/trace%PID%.txt, where *%PID%* is the process ID for the agent instance.

Attention: For WebSphere MQ V7.5, only the user that the agent process is running under can run the **fteSetAgentTraceLevel** command.

In WebSphere MQ Managed File Transfer Version 7.5, the **fteSetAgentTraceLevel** command also writes a trace for the agent process controller. The trace files produced are located in *MQ_DATA_PATH*/mqft/logs/ *coordination_qmgr_name*/agents/*agent_name*/logs/pctrace%PID%/pctrace%PID%.txt, where *%PID%* is the process ID for the agent instance.

You can also use the command to cause the agent process to generate a Javacore. The agent generates a Javacore file in the following directory: *MQ_DATA_PATH*/mqft/logs/*coordination_qmgr_name*/agents/ *agent_name*.

Because running trace can affect your performance significantly and can produce a large amount of trace data, run trace with care and only when necessary. Typically, enable trace only when asked to do so by your IBM service representative.

You can set further trace properties, for example trace file size and the number of trace files to keep, in the agent.properties file. These properties are described in Advanced agent properties.

Specify the optional -p parameter for this command only if you want to use a set of configuration options different from your default set. See "The agent.properties file" on page 548 for more information.

Syntax

fteSetAgentTraceLevel

```
►►──fteSetAgentTraceLevel──-traceAgent────classes=level──────────────────────────►
                                              ├──-disableOnAnyFFDC─────────────┤
                                              └──-disableOnFFDC──FFDC_specification─┘

►──────────────────────────────────agent_name─────────────────────────────────►◄
    └──-jc─┘  └──-p──configuration_options─┘
```

Parameters

-traceAgent *classes=level*
> Required. Level to set the agent trace and which classes to apply the trace to. Specify the following format:
>
> classes=level

For example:

```
com.ibm.wmqfte=all
```

Specify a comma-separated list of class specifications that you want the level of trace to apply to. If you do not specify this parameter, the trace level is applied to all agent classes.

You can substitute *classes* with an MQMFT package name to trace a specific package only. However, because this option captures just a subset of the agent's behavior, you are generally not recommended to use package filtering.

If (*classes*) start with a plus sign (+), the list of trace classes following the plus sign are added to any existing trace classes currently being traced.

The valid trace level options are as follows and are listed in ascending order of trace file size and detail:

off   Switches the agent trace off but continues to write information to the log files. This is the default option.

flow   Captures data for trace points associated with processing flow in the agent.

moderate
    Captures a moderate amount of diagnostic information in the trace.

verbose
    Captures a verbose amount of diagnostic information in the trace.

all   Sets agent trace to run on all agent classes.

    To start full tracing for the agent, run the following command:

```
fteSetAgentTraceLevel -traceAgent =all AGENT_NAME
```

    To stop full tracing for the agent, run the following command:

```
fteSetAgentTraceLevel -traceAgent =off AGENT_NAME
```

-disableOnAnyFFDC
Optional. If this parameter is specified, trace is disabled on the agent when it generates a First Failure Data Capture (FFDC) file.

You can specify only one of the **-disableOnAnyFFDC** and **-disableOnFFDC** parameters.

-disableOnFFDC *FFDC_specification*
Optional. If this parameter is specified, trace is disabled on the agent when it generates a First Failure Data Capture (FFDC) file that matches the *FFDC_specification*. *FFDC_specification* is a comma-separated list of values. The format of the values can be either:

class_name
    The name of the class where the FFDC originated. For example, `com.ibm.wmqfte.classA`.

class_name:probe_ID
    The name of the class and the probe ID of the location in the class that the FFDC originated from. For example, `com.ibm.wmqfte.classB:1`.

You can specify only one of the **-disableOnAnyFFDC** and **-disableOnFFDC** parameters.

-jc
Optional. Requests that the agent generates a Javacore file. The IBM service team may request that you run the command with this parameter to assist with problem diagnosis. This parameter cannot be used with any other parameter.

-p *configuration_options*
Optional. This parameter determines the set of configuration options that is used to set the agent

trace level. Use the name of a non-default coordination queue manager as the input for this parameter. The command then uses the set of properties files associated with this non-default coordination queue manager.

If you do not specify this parameter, the set of configuration options based on the default coordination queue manager is used.

agent_name
> Required. The name of the WebSphere MQ Managed File Transfer agent that you want to set the trace level for.

-? or -h
> Optional. Displays command syntax.

Example

In this example, the trace level is set to `all` for all classes for AGENT1:

```
fteSetAgentTraceLevel -traceAgent com.ibm.wmqfte=all AGENT1
```

In this example, the trace level is set to `all` for the classes `com.ibm.wmqfte.agent.Agent` and `com.ibm.wmqfte.cmdhandler` for AGENT1:

```
fteSetAgentTraceLevel -traceAgent com.ibm.wmqfte.agent.Agent,com.ibm.wmqfte.cmdhandler=moderate
AGENT1
```

In this example, subclasses are excluded from the trace because the **-traceLevel** parameter is set to off. All classes starting with com.ibm.outer are traced at verbose level except classes starting with com.ibm.outer.inner:

```
fteSetAgentTraceLevel -traceAgent com.ibm.outer=verbose AGENT1
fteSetAgentTraceLevel -traceAgent +com.ibm.outer.inner=off AGENT1
```

Return codes

0   Command completed successfully.

1   Command ended unsuccessfully.

Common problems

Common problems that might occur in your WebSphere MQ Managed File Transfer network.

- If a text transfer fails with the following error:

  ```
  BFGIO0060E: Text data conversion has failed
  ```

  This can occur for one of two reasons:

  1. One or more characters in the source file cannot be converted from the source file code page to the destination file code page. This problem can occur when code pages have different character sets and certain characters cannot be converted between them.

     If it is acceptable for conversion of some characters to not be converted, a replacement character sequence can be defined at the destination agent so that the transfer does not fail. Specify the agent property **textReplacementCharacterSequence** to define a replacement character sequence. For more information, see Table 34 on page 550.

  2. The source file encoding does not match the default encoding of the source agent. In this case performing a text transfer by using the default settings corrupts the character data. To transfer a source file that does not have the same encoding as the source agent, perform one of the following steps:

     a. Specify the file encoding in a transfer definition file. For more information, see "Using transfer definition files" on page 206.

b. Specify the file encoding by using the **-sce** parameter with the **fteCreateTransfer** command. For more information, see the topic "**fteCreateTransfer** (create new file transfer)" on page 461.

   c. Specify the file encoding as part of an Ant move or copy task. For more information, see "Using Apache Ant with WebSphere MQ Managed File Transfer" on page 328.

To check that you have selected the correct source file encoding for a transfer perform the following steps:

1. Set the destination file encoding to UTF-8.

2. Transfer the file in text mode.

3. Use a UTF-8 file viewer to view the contents of the file. If all characters in the file are correctly displayed, the source file encoding is correct.

- If you see the following output from the **fteCreateAgent** command:

```
BFGMQ1007I: The coordination queue manager cannot be contacted or has refused a
  connection attempt.
The WebSphere MQ reason code was 2058. The agent's presence will not be published.
```

  it indicates that the coordination queue manager cannot be contacted and provides the WebSphere MQ reason code for why. This information message can indicate that the coordination queue manager is currently unavailable or that you have defined the configuration incorrectly.

- If you are using user exit routines and there is a failure while the user exit is being called or just after the exit has been called, for example a product failure or power cut, it is possible the user exit will be called more than once.

- If you have an agent with a queue manager on a system with an IP address that is assigned by DHCP (rather than a static IP address), *and* the agent connects to that system by using a client TCP/IP connection, you must start the agent with the following system environment variable set:

  – On Windows:

    ```
    set FTE_JVM_PROPERTIES="-Dsun.net.inetaddr.ttl=<value>"
    ```

  – On UNIX:

    ```
    export FTE_JVM_PROPERTIES="-Dsun.net.inetaddr.ttl=<value>"
    ```

  where <value> is the time interval in seconds between each flush of the cached DNS values of the JVM. If the IP address of the queue manager system is reassigned for any reason (for example, because of a network outage, an IP lease expiry, or a system reboot), the agent reports its lost connection to the queue manager. After the JVM DNS cache is flushed, the agent can successfully reconnect. If this environment variable is not set, the agent cannot reconnect in this scenario without a JVM restart. This behavior is because the JVM internally caches the IP addresses of host names and does not refresh them by default.

- If you run the **fteStartAgent** command and see the following error message, your environment probably has additional library paths that conflict with WebSphere MQ Managed File Transfer:

```
BFGCL0001E: An internal error has occurred. The exception was: 'CC=2;RC=2495;AMQ8568:
 The native JNI library 'mqjbnd' was not found. [3=mqjbnd]
```

  If the LD_LIBRARY_PATH or LIBPATH environment variable is set to reference a 64-bit version of the library before the 32-bit version when the agent is running with a 32-bit version of Java (as is currently the case for most platforms), this error occurs.

  To resolve this issue, set the WebSphere MQ Managed File Transfer agent property javaLibraryPath to reference the correct location for the library. For example, for mqjbnd on AIX, set to: /usr/mqm/java/lib. For mqjbnd on Linux, set to: /opt/mqm/java/lib

- If you have enabled user authority checking by specifying authorityChecking=true in the agent property file and all authority checks are failing even if the user has the required authority on the relevant authority queue:

  – Ensure that the user that runs the agent has ALT_USER access control on the agent queue manager.

- If you have enabled user authority checking by specifying `authorityChecking=true` in the agent property file and WebSphere MQ error messages are written to the agent `output0.log` file perform one of the following actions:
  - Ignore the messages, the agent is not affected.
  - Grant the user who runs the agent GET authority on the SYSTEM.FTE.AUTH* queues belonging to the agent.
- If you have edited the agent property file and the agent has not picked them up:
  - Restart the agent, to ensure that the agent reads the new properties.

What to do if your agent is not listed by the `fteListAgents` command

If your agent is not listed by the **fteListAgents** command or is not displayed in the WebSphere MQ Explorer, or your file transfers are not displayed in the Transfer Log of the WebSphere MQ Explorer, you can carry out a number of problem determination steps to investigate the cause.

Use the following flowchart to help you to diagnose problems and decide what action to take next:
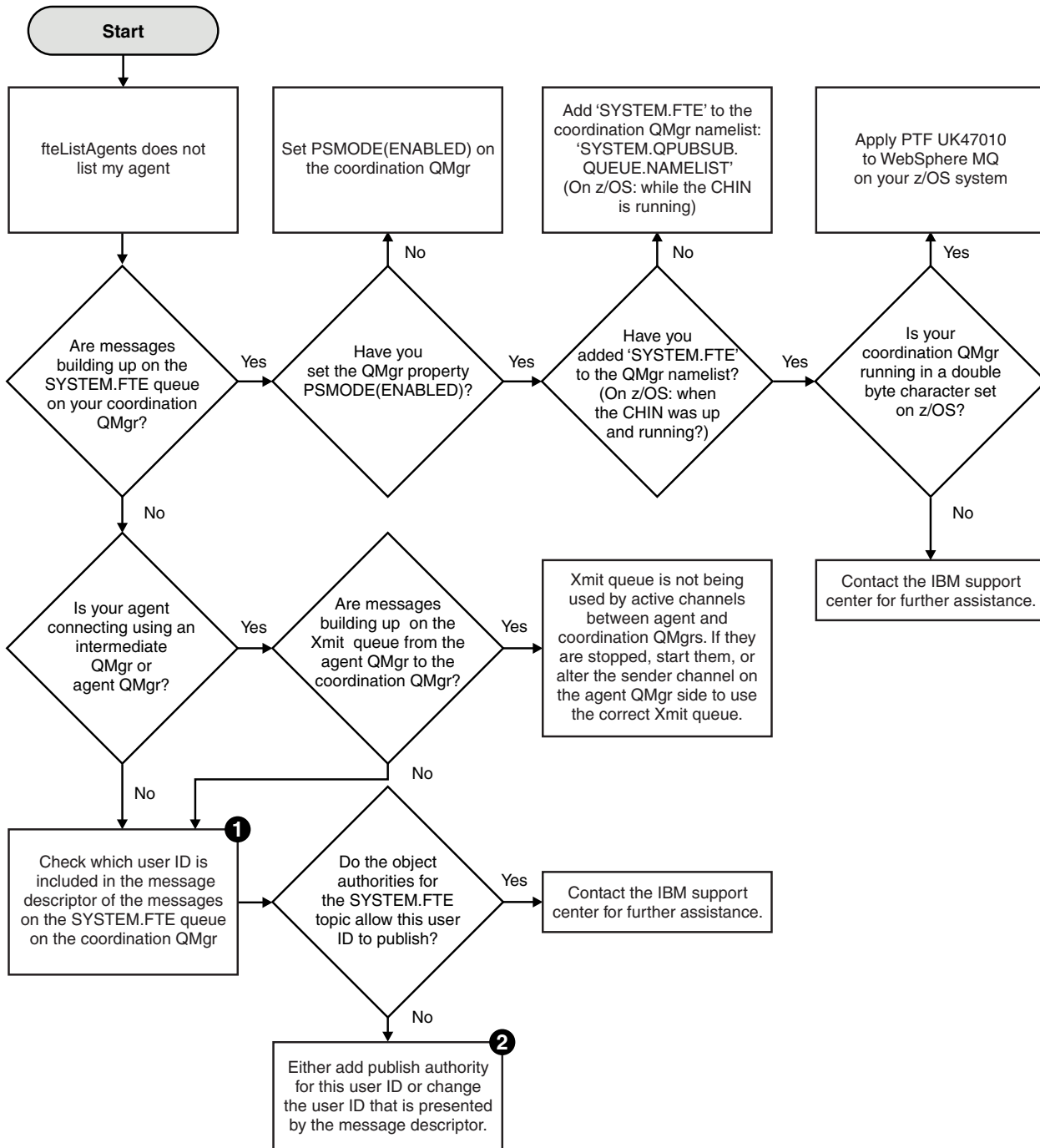
Flowchart key:

1. For more information about how to check the user ID that is presented, see "Examining messages before publication" on page 361. User IDs must conform to the MQ user name 12 character limit. If a user name is longer than 12 characters (`Administrator`, for example) the user name will be truncated before being checked for authorisation. In an example using `Administrator`, the following error message is added to the queue manager error log:

   `AMQ8075: Authorization failed because the SID for entity 'administrato' cannot be obtained.`

2. For more information about the authority needed for the SYSTEM.FTE queue, see "Authority to publish log and status messages" on page 413.

What to do if your agent process disappears but no diagnostic information is logged

On UNIX platforms, if an agent process has disappeared but the agent log files do not contain any explanation, this might be caused by the way the agent has been started.

You can check for agent diagnostic information in the following ways:
- Check whether the agent's log files state that the agent has been stopped.
- Check whether the agent lock file `agent.lck` still exists.

If you start the agent from a shell script for example, all child processes associated with that script are removed when the script completes (including the agent process). To keep the agent running past the duration of the script that called the agent, complete the following step:

1. Prefix the **fteStartAgent** command with the **nohup** command to disassociate the **fteStartAgent** process (and any child processes) from the script.

In future when the script terminates, the agent now continues to run.

What to do if the fteListAgents command shows an agent status of UNREACHABLE

Your agent is running and responds successfully to the **ftePingAgent** command, and files are being transferred normally, but the agent is listed as UNREACHABLE by the **fteListAgents** command.

Why this problem occurs

Periodically, the agent publishes its status to the coordination queue manager. The frequency that the agent publishes its status is controlled by the following two agent properties:

agentStatusPublishRateLimit
>   The maximum rate in seconds that the agent republishes its status because of a change in file transfer status.

agentStatusPublishRateMin
>   The minimum rate in seconds that the agent publishes its status. This value must be greater than or equal to the value of the agentStatusPublishRateLimit property.

Using the default settings, clocks that are out-of-sync between the agent system and the coordination queue manager system cause this issue, if the difference between the times is greater than 303 seconds. Agent status messages are considered stale if the message was sent more than the value of agentStatusPublishRateMin + the value of agentStatusJitterTolerance seconds ago. An agent with a stale status message is reported as UNREACHABLE by the **fteListAgents** command.

By default, the value of the agentStatusJitterTolerance property is 3000 milliseconds and the value of the agentStatusPublishRateMin property is 300 seconds. If the time difference between the machines plus the effective publish rate is greater than the sum of agentStatusPublishRateMin + agentStatusJitterTolerance, the time difference causes the UNREACHABLE agent status.

Resolving the problem

You can resolve this problem in either of the following ways:
- Correct the time setting differences between the agent host machine and the machine hosting the coordination queue manager, so that they are in sync.
- Increase the value of the agentStatusJitterTolerance property to account for the time difference. When you run the **fteListAgents** command, the value of agentStatusJitterTolerance is determined by the

coordination.properties configuration file in the MQMFTconfig directory. Therefore, set the property in the coordination.properties file of the MQMFT installation that the **fteListAgents** command is being run on.

Related reference:

"The agent.properties file" on page 548

Each agent has its own properties file, agent.properties, that must contain the information that an agent uses to connect to its queue manager. The agent.properties file can also contain properties that alter the behavior of the agent.

What to do if your agent or logger configuration is not secure

If a WebSphere MQ Managed File Transfer process detects a condition that a configuration file contains sensitive information, is a keystore or truststore file, and has system-wide read, write, or delete permissions, the process will fail to start if detected at startup time. If the condition was not detected at startup time but was detected at runtime, WebSphere MQ Managed File Transfer generates a warning message and ignores the contents of the configuration file. This is relevant to the protocol bridge and the Connect:Direct bridge capabilities which reload a configuration if it changes while the agent is running.

Complete the following checks to determine the cause of the problem:

1. Identify the configuration file that has been reported as not secure from the error message provided.
2. Ensure that the file access permissions match the requirements needed. For more information, see "Permissions for configuration files containing sensitive information" on page 412.
3. Restart the agent or logger. Or, in the case of the protocol bridge or Connect:Direct credentials files, wait for the next reload.

Example

In this example of an error message, a database logger is failing to start:

```
BFGDB0066E: The logger encountered a problem accessing its credentials file and will stop.
Reported error: BFGNV0145E: The 'Everyone' group has access to the file 'C:\mqmftcredentials.xml'.
```

In this example of an error message, a protocol bridge agent is failing to start:

```
BFGIO0383E: The security permissions defined for credentials file 'C:\ProtocolBridgeCredentials.xml' do not meet the
minimum requirements for a file of this type.
Reported problem: BFGNV0145E: The 'Everyone' group has access to the file C:\ProtocolBridgeCredentials.xml'.
```

Related reference:

"Permissions for configuration files containing sensitive information" on page 412

Any file used to store sensitive configuration information, meaning any file referenced from the WebSphere MQ configuration tree, must not have system-wide read, write, or (where applicable), delete permissions. These restrictions are also be applied to truststore and keystore files.

What to do if your transfer does not complete

If your transfer does not complete you can carry out a number of problem determination steps to investigate the cause.

Use the following flowchart to help you to diagnose problems and decide what action to take next:

```
                    ┌──────────────┐
                    │    Start     │
                    └──────┬───────┘
                           │
        ┌──────────────────────────────────┐      ┌──────────────────────────────┐
        │                                   │      │  Check the failure details:  │
        │          Why did my               │      │ (right click the failed      │
        │     transfer not complete?        │      │  transfer > Properties > xml)│
        │                                   │      │                              │
        └──────────────────────────────────┘      │   Correct the issue and      │
                           │                       │   resubmit the transfer.     │
                                                   └──────────────────────────────┘
```

Is MQ Explorer configured to list MFT transfers? — Yes → Is the transfer listed in the transfer log? — Yes → Is the transfer listed as failed? — Yes → Check the failure details: (right click the failed transfer > Properties > xml). Correct the issue and resubmit the transfer.

Is MQ Explorer configured to list MFT transfers? — No

Is a database or file logger running? — No / Yes

Is the transfer listed in the transfer log? — No

Is the source agent running and reporting status to the coordination queue manager? (Agents section in Explorer has a status of running) **4** — No / Yes

Is the transfer listed as failed? — No

Is the destination agent running and reporting status to the coordination queue manager? (Agents section in Explorer has a status of running) **4** — No / Yes

Is the transfer listed in the transfer log file or database table? — No → Is the source agent running? (use ftePingAgent). — No / Yes

Is the transfer listed in the transfer log file or database table? — Yes

Yes

Yes

Does
fteShowAgentDetails
report the agent as
operating correctly?

❶

No

Yes

Is the transfer
listed here?

❷

No

Yes

Is the transfer
listed
as failed?

No

Is the
destination agent
running?
(use ftePingAgent).

No

Yes

Yes

Yes

Check the failure details in
the file logger log files or
database transfer details
table for the specific transfer.

Correct the issue and
resubmit the transfer.

Flowchart key:

1. Check the agent `output0.log` for errors. If the agent reports it has successfully started, but neither WebSphere MQ Explorer or **fteShowAgentDetails** report the agent as running, then check the connectivity between the agent queue manager and the coordination queue manager. It may be that a queue manager to queue manager channel is unavailable.

2. If the source agent lists the transfer ID as an `In progress` transfer but the destination agent does not, there might be a connectivity issue between the source and destination queue managers. Use the **ftePingAgent** command from the destination agent machine to the source agent using the destination agent queue manager as the command queue manager, in the `command.properties` file. You can also run this command the other way round, from source to destination.

3. If both the source and destination agents list the transfer ID as `In progress`, this suggests there has been a connectivity issue between the source and destination queue managers since the transfer was initiated. Use the **ftePingAgent** command from the destination agent machine to the source agent

using the destination agent queue manager as the command queue manager, in the `command.properties` file. You can also run this command the other way round, from source to destination.

4. If you have been round this loop already, check whether either of statements are relevant to your situation:
   - Both source and destination agents report as `Running`, but no transfer is listed. Either the transfer request did not reach the agent command queue, or the agent although reporting as `Running`, is no longer monitoring the command queue. Check for errors in the source agent `output0.log`. Use the **ftePingAgent** command from the same machine the transfer was sent from, to the source agent, to verify the connectivity between the command queue manager and the agent queue manager, and that the agent is servicing the command queue.
   - Both source and destination agents report as `Running`, and the transfer is listed as `In progress, recovering`. Use the **ftePingAgent** command from the destination agent machine to the source agent using the destination agent queue manager as the command queue manager, in the `command.properties` file. You can also run this command the other way round, from source to destination.

What to do if you think that your transfer is stuck

On a heavily loaded system or when there are network problems between the source and destination agents, transfers can occasionally appear to be stuck in a queued or recovering state. There are a number of factors that can cause this.

Complete the following checks to determine the cause of the problem:

1. Use the **ftePingAgent** command, or in the WebSphere MQ Explorer **Agents** panel right-click on the agent name and select **Ping**, to check whether the source and destination agents are active and responding to new requests. Look at the agent logs to see if there is a current network connection problem.

2. Check whether the destination agent is running at capacity. It might be that there are numerous source agents all requesting file transfers to the same destination agent. Use the **fteShowAgentDetails** command with the **-v** (verbose) parameter, or in the WebSphere MQ Explorer **Agents** panel right-click on the agent name and select **Properties**, to see the current transfer activity for an agent. If the number of running destination transfers is at or close to the agent's maximum number of destination transfers, that can explain why some transfers for source agents appear to be stuck.

3. Transfers to and from protocol bridge agents enter a recovering state if there is a problem contacting the protocol file server. Look at the agent logs to see if there is a current connection problem.

4. Transfers are processed by an agent in priority order. Therefore in a loaded system, a low-priority transfer can remain in the queued state for some time while the agent is loaded with higher priority transfers. Eventually a low-priority transfer is started if that transfer has been queued for a while, even though there are newer higher priority transfers.

What to do if your scheduled transfer does not run or is delayed

If you have a scheduled transfer that does not run when it is due or is delayed, it might be because the agent is processing commands on its command queue. Because the agent is busy, scheduled transfers are not checked and are therefore not run.

To work around this problem, use one of the following steps:
- Configure the maxSchedulerRunDelay property in the `agent.properties` file to set the maximum interval in minutes that the agent waits to check for scheduled transfers. Setting this property ensures that the agent keeps checking for scheduled transfers even when the agent is busy. For more information about the property, see "The agent.properties file" on page 548.
- Alternatively, use a resource monitor instead of a scheduled transfer. Resource monitors work differently from scheduled transfers and are not affected by the agent being busy. For example, if you want an up-to-date file on the destination system, resource monitors reduce network traffic. This is

because the file is transferred only when a new version becomes available, rather than the file being transferred automatically. However, resource monitoring is not supported on protocol bridge agents or Connect:Direct bridge agents.

For more information, see "Resource monitoring" on page 215.

What to do if your protocol bridge agent reports that a file is not found

When the protocol bridge agent reports that the SFTP or FTP server that the protocol bridge connects to returns a `File not found` error message, this message can mean that one of a number of different error cases has occurred.

The following possible scenarios can result in a `File not found` error being returned by the SFTP or FTP server.

- The file does not exist. Check that the file you are attempting to transfer exists on the system hosting the SFTP or FTP server.
- The file path does not exist. Check that the file path exists on the system hosting the SFTP or FTP server. Check that you have entered the file path correctly into the transfer request. If necessary, correct the file path and submit the transfer request again.
- The file is locked by another application. Check whether the file is locked by another application. Wait until the file is no longer locked then submit the transfer request again.
- The file permissions do not allow the file to be read. Check whether the file has the correct file permissions. If necessary, change the file permissions and submit the transfer request again.
- The SFTP or FTP server uses a virtualized root path. If a relative file path is specified in a transfer request, the protocol bridge agent will attempt to convert the relative path into an absolute file path based on the home directory used to login to the protocol server.The WebSphere MQ Managed File Transfer protocol bridge agent can support only SFTP or FTP servers that allow files to be accessed by their absolute file path. Those protocol servers that allow access to files based only on the current directory are not supported by the protocol bridge agent.

Related concepts:

"The protocol bridge" on page 248
The protocol bridge enables your WebSphere MQ Managed File Transfer (MQMFT) network to access files stored on a file server outside your MQMFT network, either in your local domain or a remote location. This file server can use the FTP, FTPS, or SFTP network protocols. Each file server needs at least one dedicated agent. The dedicated agent is known as the protocol bridge agent. A bridge agent can interact with multiple file servers.

What to do if a directory resource monitor is not triggering files

A directory resource monitor polls a directory for files that match a trigger specification. For each file that matches the trigger specification, a transfer request is generated to the agent. When the request is submitted, the triggering file is ignored until the file is changed.

Possible reasons why the files are not triggering

1. The directory resource monitor found a file that matched the trigger specification, but the generated transfer request was invalid and the agent was unable to process the request. The reasons can include the following:

   - Invalid destination agent
   - Missing destination agent
   - Transfer canceled by program invocation

   In all these examples, the directory resource monitor marks the triggering file as processed and ignores the file even though the transfer failed.

2. The file is outside the scope of the resource monitor trigger specification. The reasons can include the following:

- Incorrect trigger pattern
- Monitoring the incorrect directory
- Insufficient file permissions
- Failure to connect to remote file system

Why a file can trigger a second transfer

A trigger file can generate a WebSphere MQ Managed File Transfer transfer request for the following reasons:
- If the presence of the trigger file is detected, when it was not there before.
- If the trigger file has been updated, causing the last modified date to change.

Potential scenarios for a second trigger are:
- The file is removed, then replaced.
- The file is locked by one application, then unlocked by another application.
- The monitor file system fails. For example, if the network connection fails, this can give the appearance of the file being removed, then replaced.
- The file directory is updated by another application, causing the last modified date to change.

How to investigate why a directory resource monitor is not being triggered

With the agent running and the directory resource monitor configured with a trigger specification, use the following command:

fteSetAgentTraceLevel -traceAgent com.ibm.wmqfte.monitor=all *agent_name*

Example

In this example, a trace file is generated for AGENT1, and shows a single poll of one monitor with a trigger specification of *.packet. The file tomato.tin does not match the trigger specification. The file rice.packet does match the trigger specification but is unchanged since the last poll, so does not trigger a transfer. The file biscuit.packet triggers a transfer because the file has the correct ending and is either new or changed since the last poll.

fteSetAgentTraceLevel -traceAgent com.ibm.wmqfte.monitor=all AGENT1

```
08:36:53.908.00  0004 ... dftStartPoll data [@MON0001SP0001:HOTEL]
08:36:54.178.00  0004 ... dftItem data [@MON0001FL0002:Ignored:/home/mondir/shop/tomato.tin:::Pattern mis-match]
08:36:54.335.02  0004 ... dftItem data [@MON0001FL0003:Ignored:/home/mondir/shop/rice.packet:11 secs::Unchanged since last trigger]
08:36:54.487.00  0004 ... dftItem data [@MON0001FL0004:Triggerred:/home/mondir/shop/biscuit.packet:::]
08:36:54.488.00  0004 ... dftGeneral data [@MON0001GN0005:Task items matches = 1]
08:36:54.553.00  0004 ... dftTask data [@MON0001TK0006:01:[AGENTNAME=PETER,FILEPATH=/home/mondir/shop/biscuit.packet, ... ]]
08:36:55.151.02  0004 ... dftTransferRequest data [@MON0001TK0007: ...]
08:36:55.632.00  0004 ... dftEndPoll data [@MON0001EP0008:HOTEL]
```

Each line includes an ID in the following format:

@MONmmmmAAssss

- The mmmm value is a number that is assigned to the monitor for this trace.
- The AA is a code for the action that is taken.
- The ssss is a statement number starting from 0001.

The AA code can take one of the following values:
- SP - Start Poll
- FL - File
- GN - General
- EX - Exception

- TK - Task
- FD - FFDC
- EP - End Poll

If there are a considerable number of Ignored files in the trace file, you can eliminate these entries by using the following command:

```
fteSetAgentTraceLevel -traceAgent com.ibm.wmqfte.monitor=moderate AGENT1
```

Related reference:

"fteSetAgentTraceLevel (set WebSphere MQ Managed File Transfer agent trace level)" on page 345
Use the **fteSetAgentTraceLevel** command to modify the current trace level for an agent dynamically.

What to do if destination files created by a transfer started by a queue resource monitor contain the wrong data

You can create a resource monitor to monitor a queue and transfer a message or a group of messages on a queue to a file. The file name can be specified by using the MQMD message descriptors on the message or the first message in a group. If a message-to-file transfer fails and the message or group is left on the queue, the next time the monitor is triggered it might result in files being created that contain the wrong data.

Why this problem occurs

1. A message-to-file transfer fails and the message or group is left on the queue.
2. A new message or group arrives on the queue.
3. The new message or group triggers the resource monitor.
4. The resource monitor creates a new transfer that uses the MQMD message descriptors from the new message or group and the data from the first message or group on the queue.
5. Files are created that contain the wrong data.

Avoiding this problem

To avoid experiencing this problem, you must manually create a transfer definition file by using the **fteCreateTransfer** command and edit the <queue> element of the file to include the attribute groupId="${GROUPID}". Then submit the transfer definition file by using the **fteCreateMonitor** command.

Example

In this example: the source agent, which is also the monitoring agent, is called AGENT_MON; the destination agent is called AGENT_DEST; the destination file name is /out/files/${WMQFTEFileName}. This example requires that the message has the MQMD message descriptor WMQFTEFileName set. The queue being monitored is LIVE_QUEUE.

1. Create a transfer definition file by running the following command:

   ```
   fteCreateTransfer -sa AGENT_MON -da AGENT_DEST -df "/out/files/${WMQFTEFileName}"
                -de error -gt /tmp/TransferDefinition1.xml -sqgi -sq LIVE_QUEUE
   ```

   The transfer definition file /tmp/TransferDefinition1.xml is generated.

2. Edit the <queue> element to include the attribute groupId="${GROUPID}". Change the line

   ```
   <queue useGroups="true">LIVE_QUEUE</queue>
   ```

   to

   ```
   <queue useGroups="true" groupId="${GROUPID}">LIVE_QUEUE</queue>
   ```

   This attribute is required so that the transfer reads the group or message that triggered the transfer from the queue instead of the first group or message on the queue.

3. Create the monitor by running the following command:

```
fteCreateMonitor -ma AGENT_MON -mq LIVE_QUEUE -mn QueueMon1 -mt /tmp/TransferDefinition1.xml
          -tr completeGroups -dv WMQFTEFileName=UNKNOWN
```

This monitor polls the queue every 60 seconds to see if a new group or message has arrived on the queue.

What to do if the destination queue is a clustered queue, or an alias to a clustered queue

When using WebSphere MQ Managed File Transfer to transfer a file into a queue, if you use a destination that is a clustered queue, or an alias to a clustered queue, you get reason code 2085, or 2082.

Why this problem occurs

The queue manager name of the destination agent is being appended to the queue name of the **-dq** parameter, when there is no explicit queue manager name on the **-dq**. The reason code 2085, or 2082, occurs because the queueManager object cannot be specified on an MQOPEN call when connecting to a clustered MQ queueManager that does not have that local clustered queue.

Avoiding this problem

1. Create a clustered queue on the queue manager.
2. Set up a remote queue definition that points to a clustered queue.

Example

This example uses a remote queue definition.

Configuration:
- Source Agent: *SAGENT*
- Source Agent Queue Manager: *SQM*
- Destination Agent: *DAGENT*
- Destination Agent Queue Manager: *DQM*
- The destination queue of the transfer is *CQ6* on queue manager *SQM*

To define remote queue definition Q6_SQM on DQM to clustered queue CQ6 in SQM (assuming that the clustered queue CQ6 is already defined in SQM), issue the MQSC command on the DQM queue manager:

```
define qremote(Q6_SQM) rname(CQ6) rqmname(SQM) xmitq(SQM)
```

Note: rname points to the clustered queue.

You can now transfer to the queue. For example:

```
 fteCreateTransfer -sa SAGENT -sm SQM -da DAGENT -dm DQM -dq Q6_SQM  /tmp/single_record.txt
```

What to do if messages are building up on your SYSTEM.MANAGED.DURABLE queues or filling your file system

If your WebSphere MQ Explorer plug-in uses a durable subscription on the coordination queue manager, messages can build up on the SYSTEM.MANAGED.DURABLE queues. If you have a high-volume WebSphere MQ Managed File Transfer network, use the WebSphere MQ Explorer plug-in infrequently, or both, this message data can fill the local file system.

To remove the buildup of messages on the SYSTEM.MANAGED.DURABLE queues, you can perform one of the following actions:
* Start the WebSphere MQ Explorer that uses the durable subscription. The WebSphere MQ Managed File Transfer plug-in for WebSphere MQ Explorer consumes the messages from the queue.
* Delete the messages from the queues manually.

To avoid this happening, you can specify that the WebSphere MQ Explorer plug-in use a non-durable subscription to the coordination queue manager. Perform the following steps in your WebSphere MQ Explorer:
1. Select **Window** > **Preferences** > **WebSphere MQ Explorer** > **Managed File Transfer**
2. From the **Transfer Log subscription type** list, choose NON_DURABLE.

Examining messages before publication

Because agents can connect to WebSphere MQ Version 6 queue managers, agents do not use the direct publication approach introduced in WebSphere MQ Version 7. Instead, agents send ordinary messages to the coordination queue manager that contain an MQRFH header. The MQRFH header requests that the message's payload is published. These messages are sent to the SYSTEM.FTE queue on the coordination queue manager, and the messages are typically published immediately from that queue. If error conditions stop this publication, you can examine the messages on the queue before publication is attempted to help with diagnosis. You can do this by completing these following steps:

1. Disable the publish/subscribe engine in the coordination queue manager.

   You can either complete this step using the WebSphere MQ Explorer or using MQSC commands. Be aware that this temporarily stops all publish/subscribe activity on the queue manager, including activity unrelated to WebSphere MQ Managed File Transfer if your coordination queue manager is also used for other purposes.

   WebSphere MQ Explorer:
         a. In the Navigator view, right-click the coordination queue manager and select **Properties**.
         b. From the **Properties** pane, select **Publish/Subscribe**.
         c. Select **Compatibility** from the **Publish/Subscribe mode** list.

   MQSC:
   ```
   ALTER QMGR PSMODE(COMPAT)
   ```
2. Send another message.

   Perform the WebSphere MQ Managed File Transfer action that has publication problems. For example, for agent registration, a message is sent whenever the agent is started (you do not need to repeatedly delete and create the agent to generate registration messages). Because the publish/subscribe engine is disabled, no publication takes place.
3. Browse the SYSTEM.FTE queue on the coordination queue manager.

   You are recommended to use the WebSphere MQ Explorer to browse your coordination queue manager's SYSTEM.FTE queue.

   WebSphere MQ Explorer:

a. In the Navigator view, expand the coordination queue manager and click **Queues**. In the Content view, right-click the SYSTEM.FTE queue and select **Browse Messages**. The **Message browser** window opens and shows the messages that would have been published.

b. The **User identifier** column shows the user ID contained in the message descriptor. A common reason for publication failure is that this user ID does not have publish authorization on the SYSTEM.FTE topic.

c. You can find out more information about each message (including the XML that will be published) by right-clicking the message and selecting **Properties**.

There is no MQSC command to inspect the contents of messages. If you are do not have the WebSphere MQ Explorer, you must use a different program that can browse queues and display all aspects of the messages found. You can use the amqsbcg sample program, if installed, as described in the following topic: Browsing queues. The `UserIdentifier` line shows the user ID. Alternatively, you can use SupportPac MO03 (Queue Load/Unload Utility); the user ID for a message is found in lines like:

```
A RTM MQ24
A USR HUGHSON
A ACC 1A0FD4D8F2F4C3C8C9D5F1F9C6F7C1C3F3F00019F7AC30000000000000000000000
```

The second line in the example is the message descriptor user ID for that message.

4. Re-enable the coordination queue manager publish/subscribe engine.

   You can either complete this step using the WebSphere MQ Explorer or using MQSC commands. After you have re-enabled the publish/subscribe engine in the coordination queue manager, any messages on the SYSTEM.FTE queue are processed immediately.

   WebSphere MQ Explorer:
   a. In the Navigator view, right-click the coordination queue manager and select **Properties**.
   b. From the **Properties** pane, select **Publish/Subscribe**.
   c. Select **Enabled** from the **Publish/Subscribe mode** list.

   MQSC:
   ```
   ALTER QMGR PSMODE(ENABLED)
   ```

Hints and tips for using WebSphere MQ Managed File Transfer

Here are some suggestions to help you to make best use of WebSphere MQ Managed File Transfer:

- If you change the `agent.properties` file, stop and restart the agent to pick up the changes.
- If you start a file transfer and there is no sign of transfer progress and no errors are reported, check that the source agent is running. If the transfer is shown but does not progress, check that the destination agent is also running. You can check the current state of agents in the agent log or verify that the agent is active with an **ftePingAgent** command.
- When you cancel an individual transfer using the **fteCancelTransfer** command, you can use either the source or destination agent in the **-agentName** parameter. However, when you delete a transfer schedule using the **fteDeleteScheduledTransfer** command, you must use the source agent name in the **-agentName** parameter.
- When you create a file transfer the source and destination file paths, either absolute or relative, are significant only on the source and destination agents. The system and directory that the **fteCreateAgent** command is issued from has no relevance to the file being transferred.
- Your default environment setup might not be able to fully support WebSphere MQ Managed File Transfer, particularly if you are running multiple concurrent transfers. If an agent has an error indicating it has run out of memory, check and update the following parameters as required:
  - For UNIX-type platforms: run the command: `ulimit -m 1048576` (or approximately 1 GB). This maximum resident set size is enough to allow a maximum of 25 concurrent transfers (25 concurrent transfers is the default for the maximum number of transfers for an agent).

– For all platforms: set the FTE_JVM_PROPERTIES environment variable as follows:
`FTE_JVM_PROPERTIES="-Xmx1024M"`

If you want to allow numbers of concurrent transfers greater than the maximum default of 25, use larger sizes for ulimit and FTE_JVM_PROPERTIES than those suggested.

Note: For Connect:Direct bridge agents the default for the maximum number of concurrent transfers is 5.

- When you use WebSphere MQ Managed File Transfer to transfer files in text mode between different platforms, the default file encoding of the source platform might not be supported by the destination platform. This causes a transfer to fail with the following error:

`BFGIO0058E: The transfer source encoding xxx is illegal or for an unsupported character set.`

You can resolve this error by setting the source encoding to one that is supported by the destination platform using an environment variable. Set the FTE_JVM_PROPERTIES system environment variable on the source system as follows: `FTE_JVM_PROPERTIES="-Dfile.encoding=xxx"`, where *xxx* is an encoding supported by the destination platform. For example, if you are transferring files in text mode from a Sun Solaris platform to a different platform and the source locale is set to "ja", set FTE_JVM_PROPERTIES as follows: `FTE_JVM_PROPERTIES="-Dfile.encoding=EUC-JP"`. If the source locale is set to "ja_JP.PCK", set FTE_JVM_PROPERTIES as follows: `FTE_JVM_PROPERTIES="-Dfile.encoding=Shift_JIS"`.

You can also resolve this error for an individual transfer by using the **-sce** parameter when you start a new transfer. For more information, see the topic "**fteCreateTransfer** (create new file transfer)" on page 461.

Guidance for setting WebSphere MQ attributes and WebSphere MQ Managed File Transfer properties associated with message size

You can change WebSphere MQ attributes and WebSphere MQ Managed File Transfer properties to affect the behavior of WebSphere MQ Managed File Transfer when reading or writing messages of various sizes.

If the size of messages being read from a source queue or written to a destination queue exceeds 1048576 bytes (1 MB), you must increase the value of the WebSphere MQ Managed File Transfer agent property **maxInputOutputMessageLength** to a value that is greater than or equal to the maximum message size to be read or written.

If the messages on the source queue are greater than 1048576 bytes, you must set the **maxInputOutputMessageLength** property on the source agent. If the messages on the destination queue are greater than 1048576 bytes you must set the **maxInputOutputMessageLength** property on the destination agent. For more information about the **maxInputOutputMessageLength** property, see Advanced agent properties.

- If the queue that the agent is writing to or reading from is local to the agent queue manager, you might have to change the WebSphere MQ queue manager, queue, and channel **MAXMSGL** attributes.

Ensure that the value of the maximum message size of the source or destination queue is greater than or equal to the value of the **maxInputOutputMessageLength** agent property.

Ensure that the value of each of the following WebSphere MQ attributes, in bytes:

– The maximum message size of the agent queue manager

– The maximum message size of the SYSTEM.FTE.STATE.*<agent_name>* queue

– The client channel maximum message size, if your agent connects to the queue manager in client mode

is greater than or equal to the result of the following calculation:

For a file-to-message transfer:
The value of **maxInputOutputMessageLength**

For a message-to-file transfer:
> The value of 3 * (**maxInputOutputMessageLength**) + 1048576

(This calculation is derived from the fact that three checkpoints can be stored in a state message and each checkpoint might have to buffer up to the maximum size of a message amount of data.)

- If the queue that the agent is writing to is a remote queue, you might have to change the WebSphere MQ queue manager, queue, and channel **MAXMSGL** attributes.

Ensure that the value of each of the following WebSphere MQ attributes is greater than or equal to the value of the **maxInputOutputMessageLength** agent property:

– The maximum message size of the remote queue manager transmission queue on the agent queue manager

– The maximum message size of the channel from the agent queue manager to the remote queue manager

– The maximum message size of the destination queue on the remote queue manager

– The maximum message size of the remote queue manager

Ensure that the value of each of the following WebSphere MQ attributes, in bytes:

– The maximum message size of the agent queue manager

– The maximum message size of the SYSTEM.FTE.STATE.*<agent_name>* queue

– The client channel maximum message size, if your agent connects to the queue manager in client mode

is greater than or equal to the result of the following calculation:

For a file-to-message transfer:
> The value of **maxInputOutputMessageLength**

For a message-to-file transfer:
> The value of 3 * (**maxInputOutputMessageLength**) + 1048576

(This calculation is derived from the fact that three checkpoints can be stored in a state message and each checkpoint might have to buffer up to the maximum size of a message amount of data.)

If you exceed the value of one of these properties, the agent stops with the following error in the agent event log:

```
BFGUT0002E: An internal error has occurred. Product failure data was captured in file
"FFDC.FTE.20100928170828514.8172766022149157013.log".
BFGSS0025E: An internal error has occurred.  The exception is: cc=2 rc=2010 op=put - MQPUT to SYSTEM.FTE.STATE.<agent_name>
BFGAG0061E: The agent ended abnormally
```

The following WebSphere MQ reason codes might be included in this message in the agent event log:

- rc=2010 This reason code maps to MQRC_DATA_LENGTH_ERROR and indicates that the value of the client channel maximum message size was exceeded. To resolve this problem ensure that the client channel maximum message size of the agent queue manager is greater than or equal to the result of the following calculation:

  `3 * (`**maxInputOutputMessageLength**`) + 1048576`

- rc=2030 This reason code maps to MQRC_MSG_TOO_BIG_FOR_Q and indicates that the value of the maximum message size of the SYSTEM.FTE.STATE.*<agent_name>* queue was exceeded. To resolve this problem ensure that the maximum message size of the SYSTEM.FTE.STATE.*<agent_name>* queue is greater than or equal to the result of the following calculation:

  `3 * (`**maxInputOutputMessageLength**`) + 1048576`

- rc=2031 This reason code maps to MQRC_MSG_TOO_BIG_FOR_Q_MGR and indicates that the value of the maximum message size of the agent queue manager was exceeded. To resolve this problem ensure that the maximum message size of the agent queue manager is greater than or equal to the result of the following calculation:

```
3 * (maxInputOutputMessageLength) + 1048576
```

If you are transferring many small messages

If the average size of the messages that the agent is reading from or writing to a queue is less than 1310 bytes and the agent is reading or writing more than 10000 messages, you must increase the maximum number of uncommitted messages property on the queue manager or reduce the amount of data in a checkpoint interval.

When the agent is reading messages from or writing messages to a queue the corresponding **GET**s or **PUT**s are grouped together into transactions. The number of **GET**s or **PUT**s in a transaction is determined by the number required to process all of the data within a checkpoint interval. The approximate amount of the data in a checkpoint interval is determined from agent properties using the following calculation:

```
Checkpoint interval data size (in bytes) = agentCheckpointInterval * agentFrameSize *
                                           agentWindowSize * agentChunkSize.
```

The default checkpoint data size is 1 * 5 * 10 * 262144 bytes = 13107200 bytes (12.5MB). The maximum number of uncommitted messages in a transaction that a queue manager supports is controlled by the **MaxUncommittedMsgs** queue manager attribute. The default value of this attribute is 10000 messages. If the average message size is less than approximately 1310 bytes the default maximum number of uncommitted messages is exceeded if there are more than 10000 messages to be written.

If you exceed the **MaxUncommittedMsgs** limit, the agent stops with the following error in the agent event log:

```
BFGSS0024E: The agent has received a reason code of '2024' from the message queue interface (MQI).
The agent cannot continue processing and will now end.
BFGAG0139I: The agent has suspended its current transfers and is now stopping.
```

The reason code 2024 maps to: MQRC_SYNCPOINT_LIMIT_REACHED.

To resolve this problem perform one of the following actions

- Increase the value of the **MaxUncommittedMsgs** queue manager property of the queue manager that the agent reading from or writing to a queue connects to. See MaxUncommittedMsgs (MQLONG).
- Reduce the amount of data in a checkpoint interval. To do this, decrease the value of one or more of the following agent properties:
  - agentCheckpointInterval
  - agentFrameSize
  - agentWindowSize
  - agentChunkSize

  For information about these agent properties, see Advanced agent properties.

If you are writing messages to a queue persistently

If you are transferring to a queue and writing the messages to the queue persistently, you might have to increase the size of the queue manager log file space to be able to log all of the data in a checkpoint interval.

If you exceed the queue manager log file space, the agent stops with the following error in the agent event log:

```
BFGSS0024E: The agent has received a reason code of '2102' from the message queue interface (MQI).
The agent cannot continue processing and will now end.
BFGAG0062E: The agent has received MQI reason code '2102'.  The agent cannot continue processing and will now end.
BFGAG0061E: The agent ended abnormally
```

The reason code '2102' maps to: MQRC_RESOURCE_PROBLEM.

To resolve this problem increase the size of the destination agent queue manager log file space.

Working with User Account Control (UAC) and virtual store

User account control is present in Windows Server 2008 R2 and other similar operating systems. This is a security infrastructure and one of its features is to divert user data stored in the central `Program Files` directory to a user location, which is known as virtual store.

If only the WebSphere MQ Managed File Transfer tools are used to manage the data structures, WebSphere MQ Managed File Transfer is not affected by UAC and virtual store. However, if the directory structure is changed or rebuilt using standard operating system tools by a non-WebSphere MQ administrator, it is possible the new structure will be diverted into a virtual store. This can cause one or more of the following situations:

- Users, including the WebSphere MQ administrator, can no longer see files in their expected location.
- An agent might fail to start, reporting message BFGCL0315 but give no supporting reason code.
- The log files cannot be found at the location reported by the agent.
- An agent when started with the **-F** parameter might fail to start, reporting message:

  `The current directory is invalid`

  .

To correct all of these situations:

- As a WebSphere MQ administrator, use the **fteDeleteAgent** and **fteCreateAgent** commands to rebuild the agent structure.
- As an operating system administrator, remove the WebSphere MQ entries in the virtual store of the affected users. For example, on Windows the location of the virtual store is as follows:
  `%USERPROFILE%\AppData\Local\VirtualStore\`

Related reference:

"fteDeleteAgent (delete a WebSphere MQ Managed File Transfer agent)" on page 485
The **fteDeleteAgent** command deletes a WebSphere MQ Managed File Transfer agent and its configuration. If the agent is protocol a bridge agent, the user credentials file is left on the file system.

"fteCreateAgent (create a WebSphere MQ Managed File Transfer agent)" on page 429
The **fteCreateAgent** command creates an agent and its associated configuration.

Guidance for running an agent or logger as a Windows service

You can run a WebSphere MQ Managed File Transfer agent, a stand-alone database logger, and a stand-alone file logger, as Windows services. If you are having a problem with these Windows services, you can use the service log files and the information in this topic to diagnose the issue.

For information about configuring your agent, stand-alone logger, or stand-alone file logger, to run as a Windows service, see "Starting an agent as a Windows service" on page 201 and "fteModifyLogger (run a WebSphere MQ Managed File Transfer logging application as a Windows service)" on page 509.

Location of log files

When you use the **fteCreateAgent**, **fteCreateWebAgent**, **fteCreateCDAgent**, **fteCreateBridgeAgent**, **fteModifyAgent**, **fteCreateLogger**, or **fteModifyLogger** command to run an agent or logger as a Windows service, you can choose the level of logging by using the **-sl** parameter. The possible values for this parameter are error, info, warn, and debug. The default value is info.

The log file for the Windows service has the file name service*date*.log, where *date* is the date when the service was started. The file for an agent is written to the directory *MQ_DATA_PATH*\mqft\logs\ *coordination_qmgr_name*\agents*agent_name*. This directory is the same directory that WebSphere MQ

Managed File Transfer agent trace files are written to. The file for the logger is written to the directory *MQ_DATA_PATH*\mqft\logs*coordination_qmgr_name*\loggers*logger_name*.

If you have problems starting an agent, or a stand-alone logger as a Windows service, try setting the logging level to debug using the **-sl** parameter. Additional information is written to the service*date*.log file.

Note: When the logging level is set to debug, the user account and password that you are using to run the Windows service are shown in the log file in plain text.

Number of log files

When you use the **fteCreateAgent**, **fteCreateWebAgent**, **fteCreateCDAgent**, **fteCreateBridgeAgent**, **fteModifyAgent**, **fteCreateLogger**,or **fteModifyLogger** command to run an agent or a stand-alone logger as a Windows service, you can choose the number of log files by using the **-sj** parameter. Specify the following text as part of your command to change the number of log files: **-sj -Dcom.ibm.wmqfte.daemon.windows.windowsServiceLogFiles=***number*, where *number* is the number of log files that you want expressed as a positive integer. If you do not specify the number of log files, the default is five.

"Log on as a service" authority

The Windows account that you use to run the service must have the **Log on as a service** right. If you try to start the service, either with the **fteStartAgent**, **fteStartLogger** command, or with the Windows **Sc.exe** command, and you are using a user account that does not have this right, a Services window opens. If the service you wanted to start was to run an agent, this window contains the following message:

```
Unable to start Windows service mqmftAgentAGENT@QMGR.
System error 1069: The service did not start due to a logon failure.
```

In this message, *AGENT* is your agent name and *QMGR* is your agent queue manager name. If you are trying to run a stand-alone logger as a service, a similar message is produced, which refers to the logger rather than an agent.

To prevent this error, give the Windows account that you use to run the service the **Log on as a service** right. For example, on Windows 7 complete the following steps:

1. From the **Start** menu, click **Administrative Tools** > **Local Security Policy**.
2. In the **Security Settings** pane, expand **Local Policies**, and then click **User Rights Assignments**.
3. In the **Policy and Security Setting** pane, double-click **Log on as a service**.
4. Click **Add User or Group**, and then add the user that you want to run the service to the list of users that have the **Log on as a service** right. You provided this user name when you ran the **fteCreateAgent**, **fteCreateWebAgent**, **fteCreateCDAgent**, **fteCreateBridgeAgent**, **fteModifyAgent**,**fteCreateLogger**, or **fteModifyLogger** command.

Note: The error System error 1069: The service did not start due to a logon failure. can also be caused by an incorrect password.

Hiding your Windows account password

When you configure your agent or stand-alone logger to run as a Windows service, you specify a user name and password to use. In the following example, the agent AGENT1 is created, which has an agent queue manager QMGR1 and is configured to run as a Windows service:

```
fteCreateAgent -agentName AGENT1 -agentQMgr QMGR1 -s -su fteuser -sp ftepassword
```

In this example, the Windows service runs with a user name of `fteuser`, which has an associated password `ftepassword`. When you run the **fteCreateAgent** command, or one of the other commands that accepts the **-s** parameter, you specify the password for the Windows account in plain text. If you prefer not to display your password, carry out the following steps:

1. Run the command (**fteCreateAgent**, **fteCreateWebAgent**, **fteCreateCDAgent**, **fteCreateBridgeAgent**, **fteModifyAgent**, **fteCreateLogger** or **fteModifyLogger**) without specifying the **-sp** parameter. For example:

   `fteCreateAgent -agentName AGENT1 -agentQMgr QMGR1 -s -su fteuser`

   Note: The command produces a message that warns you that you must set the password by using the Windows Services tool before the service starts successfully.

2. Open the Windows Services window.

3. In the list of services, right-click the agent or stand-alone logger service and select **Properties**. The agent service display name is `WebSphere MQ Managed File Transfer agent AGENT @ QMGR`, where *AGENT* is the agent name and *QMGR* is your agent queue manager name. The logger service display name is `WebSphere MQ Managed File Transfer logger for property set coordination_qmgr_name`, where *coordination_qmgr_name* is the coordination queue manager that you specified for the stand-alone logger to use as its property set. For more information about the property set, see "fteStartLogger (start a logger)" on page 531 and "fteModifyLogger (run a WebSphere MQ Managed File Transfer logging application as a Windows service)" on page 509.

4. In the **Properties** window, select the **Log On** tab.

5. Enter the password for the user account that runs the service in the **Password** and **Confirm password** fields. The password characters are hidden as you enter them.

6. Click **OK**.

Known issues

Problem using the JAVA_HOME system environment variable `V 7.5.0.2` **(applies to WebSphere MQ Managed File Transfer V7.5.0.1 or earlier only). (applies to WebSphere MQ Managed File Transfer V7.5, or later).**
> The JAVA_HOME system environment variable must not be set, otherwise the agent or logger Windows Service is unlikely to start. The agent or logger Windows Service must be run with the Websphere MQ Java runtime.

Guidance for configuring a resource monitor to avoid overloading an agent.

You can configure the property and parameter values of a WebSphere MQ Managed File Transfer resource monitor to reduce the load on an agent. Reducing the load on the agent improves the performance of that agent. There are several settings you can use, and you may need to use trial and error to find the best settings for your system configuration.

Overview of resource monitoring

When a resource monitor polls a directory or a queue, the agent completes the following stages:

- Finds all the files that match a trigger pattern (for example, all the `*.txt` files in the directory). Or finds all complete groups of messages on the queue.
- Determines which files are new or changed, or determines which groups are new on the queue.
- Initiates transfers for the files or groups that match the criteria in the two previous stages.
- Adds to the list of files and groups already transferred so they are not transferred again until they change.

For a directory monitor, the more files in the source directory and the broader the triggering pattern, the bigger the list of files the agent has to parse and compare against the list of files already transferred.

For a queue monitor, the more groups on the queue the bigger the list of groups the agent has to compare against the list of groups already transferred.

Consider the following key settings:

- Use agent property `monitorMaxResourcesInPoll` to set the maximum number of files or groups the agent includes on each poll. Using this parameter limits the number of transfers in a polling interval. It also means that the agent has less parsing to do before initiating a transfer for that number of files or groups. The next time the directory monitor or queue monitor polls, the agent includes the next set of files or groups. Agent property `monitorMaxResourcesInPoll` is available in WebSphere MQ Managed File Transfer Version 7.0.4.1 and later, for earlier versions of WebSphere MQ Managed File Transfer it is available as an interim fix for APAR IC78011.
- When creating a directory monitor, ensure that the transfer definition you configure has a source disposition of `delete`. Setting this disposition means that when the file transfer completes it is removed from the monitored directory and the agent no longer keeps it on its internal list.
- When creating a directory monitor, use the **-rl** parameter in the **fteCreateMonitor** command to limit the number of levels of the directory the agent has to recurse through. Using this parameter means that lower-level directories are not scanned unnecessarily.

Further considerations when creating a resource monitor

The process of resource monitor polling consumes agent resources. Increasing the polling interval of a monitor reduces the load placed on the agent. However, the setting of the polling interval must be balanced against generating too many transfers per polling interval. Consider the following when you set the polling interval for a resource monitor:

- How quickly you need a transfer to be initiated after a file is placed in a directory, or a group on a queue.
- The rate which files are placed into a directory, or groups onto a queue.
- The maximum transfer rate of the agent. The agent must be able to handle all the transfers that a monitor generates.

The polling interval is specified when the resource monitor is created with the **fteCreateMonitor** command by specifying the **–pi** (polling interval) and **-pu** (polling interval units) parameters. You may need to experiment to determine the best settings for your configuration.

An option to improve the stability of highly loaded agents that run resource monitors, is to reduce the agent property value of `maxSourceTransfers`. With this option the agent splits its processing time between the resource monitor and transferring files. The higher the value of agent property `maxSourceTransfers`, the more processing time is consumed by transferring files and less is available for the resource monitor. If you reduce the value of agent property `maxSourceTransfers`, the agent does fewer transfers in parallel, but it should have enough processing time to poll its resource monitors. If you lower the value of this agent property you should consider increasing the value of agent property `maxQueuedTransfers` because the number of queued transfers may increase.

If after optimizing your monitor you find that some transfers enter recovery, consider increasing an agent timeout value. The heavy load placed on the agent, may mean that the transfers timeout when negotiating the start of the transfer with the destination agent. This timeout causes the transfer to go into recovery and delays the completion of the transfer. The agent property `maxTransferNegotiationTime` specifies the time the source agent waits for a response from the destination agent. If this time is exceeded the transfer goes into recovery. The default value of this property is 30000 milliseconds (30 seconds). Increasing the value of the property, for example to 300000 Milliseconds (5 minutes), may allow the transfers to continue without timing out and avoid going into recovery.

Related concepts:

"Resource monitoring" on page 215
You can monitor WebSphere MQ Managed File Transfer resources; for example, a queue or a directory. When a condition on this resource is satisfied, the resource monitor starts a task, such as a file transfer. You can create a resource monitor by using the **fteCreateMonitor** command or the Monitors view in the WebSphere MQ Managed File Transfer plug-in for WebSphere MQ Explorer.

"Using transfer definition files" on page 206
You can specify a transfer definition file which can be used to create a file transfer. The transfer definition file is an XML file that defines some or all of the information required to create the transfer.

Related reference:

"**fteCreateMonitor** (create new resource monitor)" on page 444
The **fteCreateMonitor** command creates and starts a new resource monitor from the command line. You can monitor a resource (for example, the contents of a directory) using WebSphere MQ Managed File Transfer so that when a trigger condition is satisfied, a specified task, such as a file transfer, is started.

If you receive an error when updating your database schema on an Oracle database

You might receive the following error message when updating your database schema to the latest level by using the `ftelog_tables_oracle_702_703.sql` file: `ERROR at line 1: ORA-02289: sequence does not exist`. This error occurs because the sequences and triggers used by the tables are not in the same schema as the tables.

About this task

To fix this problem, you must edit the contents of the `ftelog_tables_oracle_702_703.sql` before running it.

Procedure

1. Find out which schema the sequences and triggers used by the WebSphere MQ Managed File Transfer database logger tables are located in.
   - On Db2, you can use the Control Center to view the tables and schema.
   - On Oracle, you can use the Enterprise Manager to view the tables and schema.
2. Open the `ftelog_tables_oracle_702_703.sql` file in a text editor.
3. In each occurrence of the text `SELECT FTELOG.`*sequence_name*`.nextval` replace the text `FTELOG` with the name of the schema where your existing sequences are located.
4. Before each occurrence of the text `CREATE OR REPLACE TRIGGER FTELOG.`*trigger_name*, insert the text `DROP TRIGGER `*schema_name*`.`*trigger_name*, where *schema_name* is the name of the schema where your existing triggers are located.
5. Use the edited `ftelog_tables_oracle_702_703.sql` file to update the database tables.

Logger error handling and rejection

The logger identifies two types of error: per-message errors and general errors.

Per-message errors are likely to be caused by a problem with one or a few individual messages. Some examples of situations, which are identified as per-message errors are as follows:

- The result code, which is a required item of data, is missing from a message
- A transfer specifies a job name that is 3000 characters long and too large for the associated database column
- A progress message is received for a transfer, but there is no record of the transfer having been started (perhaps because of a misrouted or delayed transfer start message)
- A message is received, which is not a WebSphere MQ Managed File Transfer log message

General errors are all those errors that are not per-message errors. These are likely to be because of configuration problems or program errors.

When a per-message error is encountered, the logger rejects the message by placing the message on the reject queue. Nothing is written to the output log, so periodically inspect or continuously monitor the reject queue to detect rejected messages.

If too many messages are rejected consecutively, without any messages being successfully written to the database, this is treated as a general error. For example, consider a site that always uses 10 character codes as job names, but which has inadvertently reconfigured the job name column to be two characters wide. Although data that is too wide is usually a per-message error, in this case the configuration problem is general and is detected as a general error. You can tune the number of consecutive per-message errors needed to cause a general error using the `wmqfte.max.consecutive.reject` property.

If a general error is detected the logger rolls back any messages not yet committed to the queue manager, and then retries periodically. A message identifying the problem is written to the output log and to the console if the logger was started in foreground mode with the `-F` parameter.

The location of the output logs for the logger is dependent on whether it is a stand-alone or JEE database logger. For a stand-alone database logger it is located in the directory *MQ_DATA_PATH*/mqft/logs/*coordination_qmgr_name*/loggers/*logger_name*. For a JEE database logger it is located in the standard output log of the application server.

The reject queue

Messages that result in per-message errors are moved to the reject queue. On each rejected message, a message property is set to indicate why the message was rejected. The full name of the property is `usr.WMQFTE_ReasonForRejection`, although usr. is omitted in some contexts (including JMS and the WebSphere MQ Explorer).

If you are using WebSphere MQ Explorer, you can view the contents of the reject queue by right-clicking the queue and clicking **Browse Messages**. To see why a message was rejected, double-click the message to open its properties dialog, then select the Named Properties page. You will see a property called `WMQFTE_ReasonForRejection`. Alternatively, you could write or configure a monitoring tool to obtain this information automatically.

Sometimes, you might want to reprocess messages from the reject queue. In the example described previously in this topic, with a two-character job name column in the database, the messages could be successfully processed after the width of the database column had been increased. As another example, when a transfer-complete message is rejected because its associated transfer-start was missing, the transfer-start message might be received later. Reprocessing the transfer-complete will then be successful.

To reprocess messages, move them from the reject queue to the input queue. In a normal installation, where the logger created its own managed subscription, the input queue is defined by the queue manager and has a name like SYSTEM.MANAGED.DURABLE.49998CFF20006204. You can identify the input queue by looking at the **Destination name** in the properties for the subscription SYSTEM.FTE.DATABASELogger.AUTO, or using the following MQSC command:

```
DISPLAY SUB(SYSTEM.FTE.DATABASELogger.AUTO) DEST
```

One way of moving messages between queues is to use the MA01 SupportPac, for example:

```
q -IFTE.REJECT -oSYSTEM.MANAGED.DURABLE.49998CFF20006204
```

The reject queue might contain messages rejected for various reasons, only some of which have been resolved. In this case you can still reprocess all the messages; those messages that can now be accepted are consumed, and those messages that cannot are again moved to the reject queue.

`Malformed` log messages in the transfer log are not logged by the logger. These messages are not viewed as being significant and so these messages are sent to the reject queue. For more information about transfer log messages, see "File transfer log message formats" on page 626.

If the logger is started, but no transfer information is being logged to the database

The database tables used by the WebSphere MQ Managed File Transfer logger require the database to have a page size of 8 KB or larger. If the page size of the database is not large enough, the tables are not created properly and you see the error SQLSTATE=42704.

If you are using the Java Platform, Enterprise Edition database logger, you might see the following message in the WebSphere Application Server system out log; if you are using the stand-alone database logger, you might see the following error in the `output0.log` file:

```
DB2 SQL Error: SQLCODE=-204, SQLSTATE=42704
SQLERRMC=FTELOG.TRANSFER_EVENT, DRIVER=3.40.152
```

The SQLSTATE value of 42704 indicates that a table that the logger expected to exist, in this case FTELOG.TRANSFER_EVENT, does not exist.

To fix this problem perform the following steps:

1. Check that the table exists and is complete. For information about the tables that the logger uses and their columns, see "Database tables used by the logger" on page 691.
2. If the table does not exist or is incomplete, check the page size of the database.
3. If the database size is less than 8 KB, increase the page size of your database.
   - If your database is on a test system or has no data in it, you can drop the tables and re-create the database with a page size greater than 8 KB.
   - For information about how to increase the page size, see "Increasing the page size of the log database on Db2 on Windows, UNIX or Linux" on page 29.

What to do if keystore properties failed to be read from the keystore configuration file in AMS

The keystore configuration file location, if not present in the default location, must be specified by the *MQS_KEYSTORE_CONF* variable in order for the Java AMS to run in client mode. If the location is not specified, the WebSphere MQ File Transfer Edition agent logs will show the error message: "Failed to read keystore properties from the keystore configuration file."

The default location for the keystore configuration file is `<home_directory>/.mqs/keystore.conf`. If the location of the keystore configuration file is not the default location, complete the following steps:

1. Start the FTE agent in client mode.
2. Apply AMS security to SYSTEM.FTE.DATA.<agent name> queue. If the keystore configuration file is not in this location, all transfers will fail with no acknowledgment.
3. Set the system variable *FTE_JVM_PROPERTIES* to *FTE_JVM_PROPERTIES=-DMQS_KEYSTORE_CONF=<path to keystore_config file>* for the **fteStartAgent** command.
4. Set the system variable *MQS_KEYSTORE_CONF* to *MQS_KEYSTORE_CONF<=path to keystore_config file>* for the **fteStartAgent** command. This must be set to ensure all agents run, regardless of the mode they are running in.

Note: If the Java AMS is running in bindings mode, error AMQ9062 will be shown in the queue manager's error log if the keystore configuration file is not in the default location.

BFGSS0023E errors and how to avoid them

If you uninstall a Fix Pack from an installation in order to move back to a previous version of the product, and an agent associated with the installation was involved with managed transfers at the time the uninstall took place, then that agent cannot start and will report an BFGSS0023E error. You can avoid this error by completing a number of steps that should prevent BFGSS0023E messages from appearing when the agents are restarted.

For every in-flight managed transfer that an agent is currently involved in, there is a message on the agent's SYSTEM.FTE.STATE.*agent_name* queue. This message stores checkpoint information on the managed transfer, and is used if the managed transfer goes into recovery. Once a managed transfer has finished, then the corresponding message on the SYSTEM.FTE.STATE.*agent_name* queue is removed.

Each state message contains some internal header information indicating which version of the IBM WebSphere MQ Managed File Transfer component was being used by an agent when the managed transfer was running. The version information shows the specific Fix Pack level, so, for example, if a Version 7.5.0.6 agent was running a managed transfer, then the state message for that managed transfer would contain a reference to Version 7.5.0.6.

If a Fix Pack is uninstalled from an installation, and an agent associated with that installation has in-flight transfers associated with it, then the agent fails to start and reports the following error:

```
BFGSS0023E: The agent is configured to use WebSphere MQ queues that contain data created using a later version
of the product. The agent cannot run in this configuration and will end.
```

For example, if a Version 7.5.0.6 agent has some in-flight transfers running when it is stopped and then downgraded to the Version 7.5.0.5 level, the next time the agent is started, it checks the messages on its SYSTEM.FTE.STATE.*agent_name* queue and finds that they were written when it was using Version 7.5.0.5. As it is now using Version 7.5.0.5, the agent reports the BFGSS0023E error described in the previous paragraph and shuts itself down.

As a general rule, if you want to remove a Fix Pack to the IBM WebSphere MQ Managed File Transfer component, completing the following steps should prevent the BFGSS0023E messages from appearing when the agents are restarted:

1. Ensure that all of their agents have completed their managed transfers.
2. Stop the agents.
3. Remove the Fix Pack.
4. Restart the agents.

Related tasks:

"Starting a WebSphere MQ Managed File Transfer agent" on page 201
Before you can use a WebSphere MQ Managed File Transfer agent for a file transfer, you must first start the agent.

Related reference:

"Agent queues for WebSphere MQ Managed File Transfer" on page 662
The MQSC command scripts generated by the **fteCreateAgent** command create the agent queues with parameters set to the following values. If you do not use the MQSC scripts provided to create the queues, but create the queues manually, ensure you set the following parameters to the values given.

Related information:

BFGSS0001 - BFGSS9999

Return codes for WebSphere MQ Managed File Transfer

WebSphere MQ Managed File Transfer commands, Ant tasks, and log messages provide return codes to indicate whether functions have successfully completed.

The following table lists the product return codes with their meanings:

Table 10. Return codes

Return code	Short name	Description
0	Success	The command was successful
1	Command unsuccessful	The command ended unsuccessfully.
2	Command timed out	The agent did not reply with the status of the command within a specified timeout. By default, this timeout is unlimited for managed call and transfer commands. For example, when you specify the **-w** parameter with the **fteCreateTransfer** command. By default, this timeout is 5 seconds for other commands.
3	Acknowledgement timed out	The agent did not acknowledge receipt of the command within a specified timeout. By default, this timeout is 5 seconds.
4	Wrong agent	The command was sent to the wrong agent. The agent specified in the command XML is not the agent that is reading the command queue, on which the message was placed.
20	Transfer partially successful	The transfer completed with partial success and some files were transferred.
21	Transfer stopped	The transfer was stopped by one of the user exits.
22	Cancel transfer timed out	The agent received a request to cancel a transfer but the cancellation could not be completed within 30 seconds. The transfer was not canceled.
26	Cancel ID not found	The agent received a request to cancel a transfer but the transfer cannot be found. This might be because the transfer completed before the cancel request was processed by the agent. It might also be caused because you supplied an incorrect transfer ID to the **fteCancelTransfer** command. The cancel request was ignored.
27	Cancel in progress	The agent received a request to cancel a transfer, but the transfer is already in the process of being canceled. The new cancel transfer request was ignored.
40	Failed	The transfer failed and none of the files specified were transferred.
41	Cancelled	The transfer was canceled.

Table 10. Return codes (continued)

Return code	Short name	Description
42	Trigger failed	The transfer did not take place because the transfer was conditional and the required condition was not met.
43	Malformed XML	An XML message was malformed.
44	Source agent capacity exceeded	The source agent did not have sufficient capacity to carry out the transfer.
45	Destination agent capacity exceeded	The destination agent did not have sufficient capacity to carry out the transfer.
46	Source agent maximum number of files exceeded	The number of files being transferred exceeded the limit of the source agent.
47	Destination agent maximum number of files exceeded	The number of files transferred exceeded the limit of the destination agent.
48	Invalid log message attributes	A log message is malformed. This error is an internal error. If you receive this return code contact the IBM support center for further assistance.
49	Destination unreachable	The source agent is unable send a message to the destination agent due to a WebSphere MQ problem. For example if the source agent queue manager has not been configured correctly to communicate with the destination agent queue manager.
50	Trial version violation	An attempt was made by a trial version agent to communicate with an agent that is not a trial version agent.
51	Source transfer not permitted	The `maxSourceTransfers` agent property has been set to 0. It is not permitted for this agent to be the source of any transfers.
52	Destination transfer not permitted	The `maxDestinationTransfers` agent property has been set to 0. It is not permitted for this agent to be the destination for any transfers.
53	Not authorized	The user is not authorized to perform the operation. See the accompanying message for further details.
54	Authority levels do not match	The `authorityChecking` agent property value of the source agent and destination agent do not match.
55	Trigger not supported	An attempt has been made to create a transfer with a trigger on a protocol bridge agent. This behavior is not supported.

Table 10. Return codes (continued)

Return code	Short name	Description
56	Destination file to message not supported	The destination agent does not support writing the file to a destination queue
57	File space not supported	The destination agent does not support file spaces.
58	File space rejected	The file space transfer was rejected by the destination agent.
59	Destination message to file not supported	The destination agent does not support message-to-file transfers.
60	File space lookup exception	The Web Gateway agent file space lookup was unsuccessful.
61	File space not found exception	The Web Gateway agent found no data in the database.
62	File space not authorized exception	The Web Gateway agent file space user is not authorized by the permissions database to complete the transfer.
63	File space delete action exception	The Web Gateway agent file space is being deleted by the Web Gateway.
64	Both queues disallowed	The source and destination of a transfer is a queue.
65	General data queue error	An error occurred when the WebSphere MQ Managed File Transfer agent data queue was accessed.
66	Data queue put authorization error	An error occurred when the WebSphere MQ Managed File Transfer agent data queue was accessed. WebSphere MQ Advanced Message Security is not enabled.
67	Data queue put AMS error	An authorization error occurred when the WebSphere MQ Managed File Transfer agent data queue was accessed. WebSphere MQ Advanced Message Security is enabled.
68	Transfer not supported	An attempt has been made to create a transfer that is not supported by a web agent. A web agent supports only transfers where it acts as the destination agent and the destination is a file space.
100	Monitor substitution not valid	The format of a variable substitution within a monitor task XML script was malformed.
101	Monitor resource incorrect	The number of monitor resource definitions was not valid.
102	Monitor trigger incorrect	The number of monitor trigger definitions was not valid.
103	Monitor task incorrect	The number of monitor task definitions was not valid.

Table 10. Return codes (continued)

Return code	Short name	Description
104	Monitor missing	The requested monitor is not present.
105	Monitor already present	The requested monitor is already present.
106	Monitor user exit error	A monitor user exit has generated an error during a resource monitor poll.
107	Monitor user exit canceled	A monitor user exit has requested a transaction to be canceled.
108	Monitor task failed	A monitor task has failed to complete due to error in processing the task.
109	Monitor resource failed	A monitor resource definition cannot be applied to the given resource.
110	Monitor task variable substitution failed	A variable has been specified in a monitor task but no matching name has been found in the metadata. Therefore the variable cannot be substituted with a value.
111	Monitor task source agent not valid	The source agent of the monitor transfer task does not match the agent of the resource monitor.
112	Monitor task source queue manager not valid	The source agent queue manager of the monitor transfer task does not match the agent queue manager of the resource monitor.
113	Monitor not supported	An attempt has been made to create or delete a resource monitor on a protocol bridge agent or web agent. This behavior is not supported.
114	Monitor resource denied	The directory that is scanned by the monitor resource is denied access.
115	Monitor resource queue in use	The monitor resource queue is already open, and is not compatible for input with shared access.
116	Monitor resource queue unknown	The monitor resource queue does not exist on the associated queue manager of the monitor.
118	Monitor resource expression invalid	An error occurred evaluating the XPath expression. The XPath expression is evaluated to access the user defined properties in the header of the message. The message is on a queue which is monitored by the resource monitor.
119	Monitor task source agent queue manager missing	The source agent name or source agent queue manager name is missing from the monitor task definition.
120	Monitor queue not enabled	The monitor resource queue is not enabled.

Table 10. Return codes (continued)

Return code	Short name	Description
122	Monitor command queue not enabled for context id	The monitor agent command queue is not enabled for set context identification.

The following table lists the product intermediate reply codes with their meanings:

Table 11. Intermediate reply codes

Reply code	Short name	Description
-2	ACK	The request has been received but is pending completion.
-3	PROGRESS	The request is for a number of files and some are still pending completion.

Note:

Reply codes are only present if the process that generates the request supplies a reply queue. These are intermediate replies and WebSphere MQ Managed File Transfer commands return the final reply code only.

Related reference:

"Return codes for files in a transfer"
Individual files within a transfer have their own result codes which have different meanings to the overall return code from a command.

"HTTP response codes" on page 379
Status codes are returned in HTTP responses to requests made to the .

Return codes for files in a transfer

Individual files within a transfer have their own result codes which have different meanings to the overall return code from a command.

In a transfer log progress message that has an `<action>` element set to a value of "progress", each file reported has a `<status>` element with a `resultCode`. For example:

```
<action time="2009-11-23T21:28:09.593Z">progress</action>

...
    <status resultCode="1">
        <supplement>BFGIO0006E: File &quot;C:\destinationfiles\dest1.doc&quot;
            already exists.</supplement>
    </status>
```

The following table describes the possible values for `resultCode`:

Table 12. File result codes in a transfer

Result code value	Description
0	Success. The file transferred successfully.
1	Failed. The file failed to transfer. See the <supplement> element for more details of the error.

Table 12. File result codes in a transfer (continued)

Result code value	Description
2	Warning. The file transferred but a warning message has been reported. For example, the source file cannot be deleted although the source disposition is set to delete. See the <supplement> element for more details of the warning.

HTTP response codes

Status codes are returned in HTTP responses to requests made to the .

The header of a response returned by the Web Gateway contains an HTTP response code. The HTTP header in the following example contains the HTTP response code 200 OK:

```
HTTP/1.1 200 OK
Server: WAS/6.0
Content-length: 0
```

The following table describes the possible values for the HTTP response code and an example of an associated WebSphere MQ Managed File Transfer error code that can be returned. For more information about the WebSphere MQ Managed File Transfer error codes, see Diagnostic messages.

Table 13. HTTP response codes

HTTP response code	Example WebSphere MQ Managed File Transfer error code	Example description
200 OK	None	A valid request has been handled correctly and optionally a response has been provided to the user.
202 Accepted	None	A valid request has been handled correctly but WebSphere MQ Managed File Transfer does not guarantee that the requested action has completed. For example, a file upload transfer request has been handled and submitted to a WebSphere MQ Managed File Transfer agent but the transfer has not yet taken place.
400 Bad Request	BFGWI0001	The URI is not valid because it is missing a resource type.
403 Forbidden	BFGWI0056	There is no WebSphere MQ Message Descriptor (MQMD) user identifier defined for the user.
404 Not Found	BFGWI0015	The requested resource cannot be found.
405 Method Not Allowed	BFGWI0016	The requested resource does not support the HTTP verb that has been used in the request. For example, a GET has been used against a resource that only allows POST or DELETE.

Table 13. HTTP response codes (continued)

HTTP response code	Example WebSphere MQ Managed File Transfer error code	Example description
410 Resource Gone	BFGWI0031	The requested resource is no longer available. For example, the requested file has been deleted from the file space.
413 Request Entity Too Large	BFGWI0026	The request contains a file that is too large to be handled by the server.
415 Unsupported Media Type	BFGWI0017	A request has been received with a media type, specified by the `Content-type` HTTP header, that is not supported.
500 Internal Server Error	BFGWI0018	An internal error has been encountered when handling the request. An FFDC or ABEND file has been produced.
502 Bad Gateway	BFGWI0019	The request cannot be completed because an error occurred outside WebSphere MQ Managed File Transfer. For example, a WebSphere MQ queue manager is not available.
503 Service Unavailable	BFGWI0020	The destination is temporarily unavailable. For example, a WebSphere MQ queue is full.
504 Gateway Timeout	BFGWI0021	An attempt to complete the request has timed out because of time limits imposed by WebSphere MQ Managed File Transfer, or because of time limits imposed by the HTTP client.

Troubleshooting the Web Gateway

Use the following reference information and examples to help you diagnose errors returned from the Web Gateway.

Verifying your Web Gateway installation

Follow these instructions to check that your WebSphere MQ Managed File Transfer Web Gateway application is deployed correctly.

Before you begin

Before verifying your Web Gateway configuration, you must follow the instructions to deploy the Web Gateway application. See "Configuring the Web Gateway" on page 165.

About this task

Procedure

1. Ensure that you are logged on to the application server environment with a user ID that has the `wmqfte-admin` security role. For more information, see "User roles for the Web Gateway" on page 113.
2. In a web browser, type the following URI:

   `http://`*host*`/wmqfte/ivt?logdbschema=FTELOG&webdbschema=FTEWEB`

If you defined a context root for the Web Gateway application other than the default value of `wmqfte`, use the following URI:

`http://host/context_root/ivt?logdbschema=FTELOG&webdbschema=FTEWEB`

Note: During configuration of the Web Gateway, you set up database tables for storing information about file spaces and transfer history. The Web Gateway installation verification application assumes that you used the default values for the database schema names. If you defined database schema names other than the default values of FTELOG for the transfer history database and FTEWEB for the file space information database, you must change the schema names that are specified in the URI. Use the following query terms to specify the database schema names:

logdbschema
> Schema name for the transfer history database

webdbschema
> Schema name for the file space information database

For example, if your transfer history database has a schema name of MYLOG and your file space information database has a schema name of MYWEB, use the following URI:

`http://host/wmqfte/ivt?logdbschema=MYLOG&webdbschema=MYWEB`

For more information about setting up databases, see "Setting up a database for use with file spaces" on page 165 and "Configuring the database logger for use with the Web Gateway" on page 187.

Results

The web browser displays a page that lists configuration information for your Web Gateway installation, and the results of testing some basic Web Gateway functions. For more information, see "The Web Gateway installation verification application" on page 188.

The Web Gateway installation verification application

WebSphere MQ Managed File Transfer provides a Web Gateway installation verification application. Use this application to view configuration values for your Web Gateway installation and test basic Web Gateway functions.

For information about how to access the installation verification application, see "Verifying your Web Gateway installation" on page 187. The application displays two types of information: configuration values for your Web Gateway installation, and the results of testing basic Web Gateway functions.

Configuration values

When you deploy the Web Gateway in an application server, you provide values for several initialization parameters. If you are using WebSphere Application Server Version 7.0, you provide these values using the **Initialize parameters for servlets** step in the administration console. If you are using WebSphere Application Server Community Edition, you set these values in the `web.xml` file.

Under the heading **Web Gateway configuration information**, the application lists the values for the following Web Gateway settings:

Servlet information
> The name and version of the Web Gateway servlet that you have deployed.

Web Gateway name
> The name of the Web Gateway that you deployed. You provided this value for the `webGatewayName` initialization parameter.

Context root
> The context root that you defined for the Web Gateway application. In WebSphere Application

Server Community Edition, this is the value of the `<web:context-root>` element in the `WEB-INF/geronimo-web.xml` file. In WebSphere Application Server Version 7.0, this value is set in the **Map context roots for Web modules** step when you install the Web Gateway application. The default value is `wmqfte`.

File space root directory
> The root directory path for file spaces created and managed by the Web Gateway. You provided this value for the `fileSpaceRoot` initialization parameter.

Temporary file upload root directory
> The directory path for the storage of temporary files related to Web Gateway-initiated transfers. You provided this value for the `tempFileUploadDir` initialization parameter.

Maximum size of temporary file upload directory
> The maximum amount of space, in MB, that a user is allowed for storing temporary files related to Web Gateway-initiated transfers. You provided this value for the `maxTempFileUploadSpace` initialization parameter.

MQMFT web agent name
> The name of the WebSphere MQ Managed File Transfer agent that acts as the source for Web Gateway-initiated transfers. You provided this value for the `agentName` initialization parameter. This is the name that you specified for your web agent, using the `-agentName` parameter, when you ran the `fteCreateWebAgent` command.

Coordination queue manager name
> The name of the coordination queue manager that is used by the Web Gateway for logging of transfer information. You provided this value for the `coordinationQMgr` initialization parameter.

Default MQMD user ID
> The default WebSphere MQ Message Descriptor (MQMD) user ID to associate with a requesting user when there is no specific MQMD user ID defined for the user. You provided this value for the `defaultMQMDUserID` initialization parameter.

▶ V 7.5.0.1 **CSRF Protection**
> Indicates whether the Web Gateway is currently configured to perform CSRF token validation to prevent cross-site request forgery attacks. You provided this value for the `CSRFProtection` initialization parameter.

Application server information
> The name and version of the application server hosting the Web Gateway application.

Web Gateway tests

Under the heading **Results of Web Gateway tests**, the installation verification application shows the results of several tests. If a test fails, a WebSphere MQ Managed File Transfer error code and message are displayed in the **Information** column. For more information about error messages, see Diagnostic messages. The following tests are listed:

Upload file to temporary storage

> Tests the directory that is named in the **Temporary file upload root directory** field. The application tests that the directory exists and is readable and writeable, and that data written to the directory can be read back.

Upload file to file space storage

> Tests the directory that is named in the **File space root directory** field. The application tests that the directory exists and is readable and writeable, and that data written to the directory can be read back.

Transfer history database access

Tests that the connection to the transfer history database exists. If you are using WebSphere Application Server Version 7, the application tests the data source that you configured when deploying the Web Gateway. For more information, see "Preparing to deploy the Web Gateway with WebSphere Application Server Version 7.0" on page 178. If you are using WebSphere Application Server Community Edition, the application tests the database pool that you configured when deploying the Web Gateway. For more information, see "Preparing to deploy the Web Gateway with WebSphere Application Server Community Edition" on page 167. The application checks that the database can be accessed using the credentials that you supplied when you set up the data source or database pool.

The application also checks that the required database tables exist. For more information, see "Setting up a database for use with file spaces" on page 165 and "Configuring the database logger for use with the Web Gateway" on page 187.

The final part of the test checks that Java Persistence API (JPA) objects have been correctly defined.

File space information database access

Tests that the connection to the file space information database exists. If you are using WebSphere Application Server Version 7, the application tests the data source that you configured when deploying the Web Gateway. For more information, see "Preparing to deploy the Web Gateway with WebSphere Application Server Version 7.0" on page 178. If you are using WebSphere Application Server Community Edition, the application tests the database pool that you configured when deploying the Web Gateway. For more information, see "Preparing to deploy the Web Gateway with WebSphere Application Server Community Edition" on page 167. The application checks that the database can be accessed using the credentials that you supplied when you set up the data source or database pool.

The application also checks that the required database tables exist. For more information, see "Setting up a database for use with file spaces" on page 165 and "Configuring the database logger for use with the Web Gateway" on page 187.

The final part of the test checks that Java Persistence API (JPA) objects have been correctly defined.

Enabling trace for the Web Gateway

Enable trace on the application server hosting the Web Gateway to diagnose problems with the Web Gateway.

Related tasks:

"Enabling trace with WebSphere Application Server Community Edition"
If the Web Gateway application is running in WebSphere Application Server Community Edition, follow these instructions to enable trace of the Web Gateway application. Trace is produced by the Web Gateway application when it receives and processes requests.

"Enabling trace with WebSphere Application Server Version 7.0" on page 384
If the Web Gateway application is running in WebSphere Application Server Version 7.0, follow these instructions to enable trace of the Web Gateway application. Trace is produced by the Web Gateway application when it receives and processes requests.

Enabling trace with WebSphere Application Server Community Edition

If the Web Gateway application is running in WebSphere Application Server Community Edition, follow these instructions to enable trace of the Web Gateway application. Trace is produced by the Web Gateway application when it receives and processes requests.

About this task

Trace files are written to the application server standard output (STDOUT) file. To enable trace in WebSphere Application Server Community Edition perform the following steps:

Procedure

1. Open the `logging.properties` file for the application server Java Runtime Environment in a text editor. The `logging.properties` file can be found in the `<WASCE_JRE>`/jre/lib directory, where *WASCE_JRE* is the location of the Java Runtime Environment that is used by WebSphere Application Server Community Edition.

2. Add the following lines to the `logging.properties` file:

```
com.ibm.wmqfte.level=FINEST
com.ibm.wmqfte.handlers=com.ibm.wmqfte.ras.container.EventLogFileHandler,com.ibm.wmqfte.ras.container.TraceLogFileHandler
java.util.logging.ConsoleHandler.level=FINEST
```

3. Save the `logging.properties` file.

4. Restart WebSphere Application Server Community Edition.

Related tasks:

"Preparing to deploy the Web Gateway with WebSphere Application Server Community Edition" on page 167

Use these instructions to set up your environment before deploying the WebSphere MQ Managed File Transfer Service Web Gateway enterprise application to WebSphere Application Server Community Edition. Customize the example deployment plan for your environment.

Enabling trace with WebSphere Application Server Version 7.0

If the Web Gateway application is running in WebSphere Application Server Version 7.0, follow these instructions to enable trace of the Web Gateway application. Trace is produced by the Web Gateway application when it receives and processes requests.

About this task

You do not need to restart the application server to enable trace. Trace files are written to the application server log directory. To enable trace in WebSphere Application Server Version 7.0 perform the following steps:

Procedure

1. Select **Troubleshooting-> Logs and Trace** from the WebSphere Application Server Version 7.0 administration console.

2. On the **Logging and Tracing** panel, click the name of the application server that the Web Gateway application is deployed on. A new panel opens.

3. Click **Change Log Detail Levels** to view the current logging levels for the application server.

4. Select the **Runtime** tab to enable trace on the currently running instance of the application server.

   a. Add the trace level `com.ibm.wmqfte=all` to the existing configuration. If existing trace levels are configured, use a colon to separate the trace level. For example, if your server is already configured with the trace level `*=info`, add Web Gateway trace by setting `*=info:com.ibm.wmqfte=all`.

   b. Click **OK** to save the changes.

5. Optional: If you want trace to be enabled when the application server is restarted, select the **Configuration** tab.

   a. Add the trace level `com.ibm.wmqfte=all` to the existing configuration. If existing trace levels are configured, use a colon to separate the trace level. For example, if your server is already configured with the trace level `*=info`, add Web Gateway trace by setting `*=info:com.ibm.wmqfte=all`.

   b. Click **OK** to save the changes.

Related tasks:

"Preparing to deploy the Web Gateway with WebSphere Application Server Version 7.0" on page 178
Use these instructions to define required resources before deploying the WebSphere MQ Managed File
Transfer Web Gateway enterprise application to WebSphere Application Server Version 7.0. You must
customize the example deployment plan for your environment.

Common problems

The following reference and task information includes examples of errors returned by the Web Gateway
and tips about how to avoid causing errors.

Case-sensitivity of Uniform Resource Identifiers

The URI of a request through the Web Gateway has some parts that are case-sensitive and some parts
that are not case-sensitive.

For more information, see "Uniform Resource Identifier syntax for using the Web Gateway" on page 869.
The following example shows the result of addressing a *transfer* resource using uppercase in the URI.

1. This HTTP request submits a request for information about a transfer:

   ```
   GET HTTP/1.1 /TRANSFER/414d51204d554e474f4e474f4d55474d512474f4e4ca74f2
   Host: example.com
   User-Agent: mozilla
   ```

2. The Web Gateway returns an HTTP response with the following format:

   ```
   HTTP/1.1 404 Not Found
   Content-Type text/html;charset=ISO-8859-1
   Content-Language en-US
   Content-Length 97
   Connection Close
   Date Wed, 28 Apr 2010 15:34:28 GMT
   Server WebSphere Application Server/7.0
   Error 404: SRVE0190E:
   File not found: /TRANSFER/414d51204d554e474f4e474f4d55474d512474f4e4ca74f2
   ```

   The error message is returned from the application server. The exact wording of the error message
   depends on the application server that you have deployed the Web Gateway into.

To make the request valid specify the resource name in the URI of the request in lowercase, as shown in
the following example:
1. ```
   GET HTTP/1.1 /transfer/414d51204d554e474f4e474f4d55474d512474f4e4ca74f2
   Host: example.com
   User-Agent: mozilla
   ```

If you receive an HTTP response with a status code other than 200, see the HTTP Response Codes topic
for more information.

Configuring the database logger for use with the Web Gateway

The following example shows the result of requesting the status of a transfer when the database logger is
not correctly configured.

About this task

1. This HTTP request submits a transfer query:

   ```
   GET HTTP/1.1 /transfer/414d51204d554e474f2afed834435bc6edaf323520204cee
   Host: example.com
   User-Agent: mozilla
   ```

2. The Web Gateway returns an HTTP response with the following format:

   ```
   HTTP/1.1 500 Internal Server Error
   Server: WAS/6.0
   Content-length: 93
   ```

```
Content-type: text/plain

BFGWI0018E: The request could not be completed due to an internal
web application server error.
```

To configure the database logger so that the request is processed correctly, perform the following steps:

Procedure

1. Install the WebSphere MQ Managed File Transfer database logger. For more information on how to install and configure the database logger, see "Configuring a WebSphere MQ Managed File Transfer logger" on page 133.
2. If you already have the WebSphere MQ Managed File Transfer database logger installed, ensure that your database tables are up to date. Use the SQL files provided in the following directories to update your database tables:
   - On distributed platforms: *MQ_INSTALLATION_PATH*/mqft/sql

Related tasks:

"Installing the WebSphere MQ Managed File Transfer JEE database logger" on page 152
Follow these instructions to install and configure the JEE database logger.

"Installing the WebSphere MQ Managed File Transfer stand-alone database logger" on page 141
Complete these steps to install and configure the stand-alone database logger.

Invalid requests for viewing transfer status

When you are submitting a request through the Web Gateway to view the status of a file transfer, you might receive an HTTP error code and a WebSphere MQ Managed File Transfer error message. The following example shows the result of requesting the status of an invalid transfer ID.

1. This HTTP request submits a transfer ID which has been truncated:

```
GET HTTP/1.1 /transfer/414d51204d554e474f2
Host: example.com
User-Agent: mozilla
```

2. The Web Gateway returns an HTTP response with the following format:

```
HTTP/1.1 400 Bad Request
Server: WAS/6.0
Content-length: 64
Content-type: text/plain

BFGWI0022E: The supplied transfer ID did not have a length of 48 characters.
This is not a valid transfer ID.
```

If you receive an HTTP response with a status code other than 200, see the HTTP Response Codes topic for more information.

Problems with uploading files

When you are submitting a request through the Web Gateway to upload a file, you might receive an HTTP error code and a WebSphere MQ Managed File Transfer error message. The following examples show some possible causes of errors received when requesting a file upload.

Failing to specify an MQMD user ID

If you request a file upload using the Web Gateway and there is no WebSphere MQ Message Descriptor (MQMD) user ID defined, the transfer fails with an HTTP response code of 403. For more information about the HTTP response codes returned by the Web Gateway, see the topic "HTTP response codes" on page 379. If you have enabled trace for the application server hosting the Web Gateway, the following information is written to the trace file:

```
BFGWI0056E: User fte-user is not permitted to access the system due to an MQMD
user identifier not being available.
```

In this example, *fte-user* is the user submitting the file upload request. For instructions on configuring trace in your application server, see "Enabling trace for the Web Gateway" on page 383.

To successfully submit file transfer requests through the Web Gateway, you must define the MQMD user ID to use for the transfer. You can either define a specific MQMD user ID for each user, or define a default MQMD user ID.

To define a set of mappings between web user ID and MQMD user ID, use the Web Gateway administration API. For more details, see the topics "Example: Mapping web user IDs to MQMD user IDs" on page 312 and "XML format for mapping web user ID to an MQMD user ID" on page 892. If a user who does not have an MQMD user ID defined submits a file upload request, the value of the `defaultMQMDUserID` parameter is used. For instructions on setting this parameter, see the topics "Preparing to deploy the Web Gateway with WebSphere Application Server Community Edition" on page 167 and "Deploying the Web Gateway with WebSphere Application Server Version 7.0" on page 184.

Failing to specify a destination agent

1. This HTTP request submits a request to upload a file without specifying a destination agent:

```
POST HTTP/1.1 /file/agent/
Host: example.com
User-Agent: mozilla
Content-Type: multi-part/form-data; boundary=Aa6b74
x-fte-checksum: MD5


--Aa6b74
Content-Disposition: form-data; name="files"; filename="myfile.txt"
Content-Type: text/plain

Account No, Balance
123456, 100.00
234567, 1022.00
345678, 2801.00
456789, 16.75
--Aa6b74
```

2. The Web Gateway returns an HTTP response with the following format:

```
HTTP/1.1 400 Bad Request
Server: WAS/6.0
Content-length: 62
Content-type: text/plain

BFGWI0002E: URI is incomplete: missing destination agent name.
```

To make the request valid specify the destination agent name in the URI of the request, as shown in the following example:

1.
```
POST HTTP/1.1 /file/agent/ACCOUNTS
Host: example.com
User-Agent: mozilla
Content-Type: multi-part/form-data; boundary=Aa6b74
x-fte-checksum: MD5


--Aa6b74
Content-Disposition: form-data; name="files"; filename="myfile.txt"
Content-Type: text/plain

Account No, Balance
123456, 100.00
234567, 1022.00
345678, 2801.00
456789, 16.75
--Aa6b74
```

If you receive an HTTP response with a status code other than 200, see the HTTP Response Codes topic for more information.

Attempting to create a file space without the required authority

To create a file space through the WebSphere MQ Managed File Transfer Web Gateway, your user ID must be associated with the appropriate MQMFT security roles. If you attempt to create a file space without the correct authority, you receive an HTTP error code and a WebSphere MQ Managed File Transfer error message. The following example shows a user who does not have the appropriate authority attempting to create a file space.

1. This HTTP request follows the required format for creating a file space. The user submitting the request is `jill`, who is a member of the group `employees`. The `employees` group is defined in the application server environment that hosts the Web Gateway. The group `employees` is not associated with either the `wmqfte-filespace-create` role or the `wmqfte-admin` role. The user `jill` is attempting to create a file space named `kevin`, into which the users `jill` and `lakshmi` can transfer files.

```
POST HTTP/1.1 /admin/filespace/kevin
Host: example.com
User-Agent: mozilla
Content-Type: application/xml

<?xml version="1.0" encoding="UTF-8"?>
<filespaces>
  <filespace>
    <quota bytes="1048576"/>
    <writers>
      <authorized action="add">
        <agent-user>jill</agent-user>
        <agent-user>lakshmi</agent-user>
      </authorized>
      <unauthorized action="add">
        <agent-user>mary</agent-user>
      </unauthorized>
    </writers>
  </filespace>
</filespaces>
```

2. The Web Gateway returns an HTTP response with the following format:

```
HTTP/1.1 401 Unauthorized
Server: Apache-Coyote/1.1
Content-Type: text/plain;charset=ISO-8859-1

BFGWI0014E: User not authorized to perform the request.
```

To make the request valid, the user `jill` must be added to an application server group that is associated with one of the MQMFT roles `wmqfte-admin` or `wmqfte-filespace-create`. The example deployment plan provided with the Web Gateway shows a sample security configuration for WebSphere Application Server Community Edition. This plan associates the `wmqfte-admin` role with the `administrators` group and the `wmqfte-filespace-create` role with the `managers` and `administrators` groups. The user `jill` does not belong to either of these groups and so cannot create a file space.

For more information about configuring security permissions in your application server, see the topics "Preparing to deploy the Web Gateway with WebSphere Application Server Community Edition" on page 167 and "Deploying the Web Gateway with WebSphere Application Server Version 7.0" on page 184.

For more information about the error codes returned by the Web Gateway administration API, see the HTTP Response Codes topic.

Related concepts:

"Securing the Web Gateway" on page 111
There are a number of ways that you can secure the Web Gateway. You must perform some of these security steps before you can use the Web Gateway. The other steps are optional and can increase the security of your Web Gateway and WebSphere MQ Managed File Transfer network, but they are not required for you to use the Web Gateway.

Related reference:

"User roles for the Web Gateway" on page 113
WebSphere MQ Managed File Transfer has defined several different roles that control the actions a user can take.

Attempting to create a file space that already exists

File spaces that you create through the WebSphere MQ Managed File Transfer Web Gateway must have unique names. If you attempt to create a file space with a name that is already in use, this will be treated as an attempt to modify the file space. If you do not have permission to modify the file space, you receive an HTTP error code and a WebSphere MQ Managed File Transfer error message.

1. This HTTP request submits a request to create a file space called `murray`. In this example, the file space `murray` already exists and the user submitting the request does not have permission to modify this file space.

   ```
   POST HTTP/1.1 /admin/filespace/murray
   Host: example.com
   User-Agent: mozilla
   Content-Type: application/xml
   Content-Length: 266

   <?xml version="1.0" encoding="UTF-8"?>
   <filespaces>
     <filespace>
       <quota bytes="1048576"/>
       <writers>
         <authorized>
           <agent-user>neerav</agent-user>
           <agent-user>SYS.ADMIN.*</agent-user>
         </authorized>
         <unauthorized>
           <agent-user>olivia</agent-user>
         </unauthorized>
       </writers>
     </filespace>
   </filespaces>
   ```

2. The Web Gateway returns an HTTP response with the following format:

   ```
   HTTP/1.1 400 Bad Request
   Server: Apache-Coyote/1.1
   Content-Type: text/plain;charset=ISO-8859-1

   BFGWI0014E: User not authorized to perform the request.
   ```

To make the request valid, specify a file space name that is not already in use. For information about listing the file spaces in your WebSphere MQ Managed File Transfer environment, see the topics "Example: Listing all file spaces" on page 307 and "Web Gateway administration API reference" on page 885.

For more information about the error codes returned by the Web Gateway administration API, see the HTTP Response Codes topic.

Web agent fails to start

If you receive an error from the **fteStartAgent** command, and you are attempting to start a web agent, check that the SYSTEM.FTE.WEB.*gateway_name* queue exists.

Example error

When you run the **fteCreateWebAgent** command, several WebSphere MQ queues are created. When you run the **fteStartAgent** command with a web agent, the agent can only start if these queues exist. If one of these queues is missing, the agent fails to start and a message is written to the agent log:

```
The agent received MQI reason code 2085 when opening queue 'SYSTEM.FTE.WEB.WG1_GTWY' on local queue manager 'QM1'.
The agent cannot continue and will end.
00000001 AgentRuntime E BFGAG0061E: The agent ended abnormally
```

If you see this error, check that both the SYSTEM.FTE.WEB.RESP.*agent_name* and SYSTEM.FTE.WEB.*gateway_name* queues exist. The SYSTEM.FTE.WEB.*gateway_name* queue is shared between all web agents associated with that Web Gateway and so is not deleted when you run the **fteDeleteAgent** command, in case another web agent is still running. Users must manually delete this queue, so another user of the Web Gateway might have deleted the queue without realizing that another web agent had been created.

Timeout when sending a file to a file space

When sending a file from a source agent to a destination file space, you might see the return code 58 and the following message: BFGFS0008E: Failed to look up a file space '*file_space_name*' for user '*user_name*' due to a timeout. This problem occurs only when the Web Gateway is deployed on WebSphere Application Server Version 7.0.

This problem might be caused by **Support distributed two phase commit protocol** not being selected in the application server. To enable this behavior perform the following steps:

1. Select **Resources** > **JMS** > **Queue connection factories** from the WebSphere Application Server Version 7.0 administration console navigation.
2. On the **Queue connections factories** panel, select the resource named jms/WMQFTEWebAgentConnectionFactory.
3. In the **Advanced** section, ensure that the **Support distributed two phase commit protocol** check box is selected.

Request fails because of an encoding problem

If the WebSphere Application Server Version 7.0 is running on a machine where either the default encoding is not UTF-8 or the default encoding does not map to UTF-8 (for example, cp1252), the Web Gateway cannot complete the request.

About this task

The request fails with the following error:

```
BFGWI0018E:(WEBGATEWAY) The request could not be completed due to an internal web
application server error. Caused by: Invalid byte 2 of 4-byte UTF-8 sequence.
```

To resolve this problem, set the Java file.encoding system property on the JVM by completing the following steps:

Procedure

1. Open the WebSphere Application Server administration console and navigate to: **Application servers** > *server name where the Web Gateway is located* > **Process definition** > **Java Virtual Machine**.
2. Add the following argument to the **Generic JVM arguments**:
   ```
   -Dfile.encoding=UTF8
   ```
3. Shut down and restart WebSphere Application Server to refresh the configuration.

HTTP response codes

Status codes are returned in HTTP responses to requests made to the .

The header of a response returned by the Web Gateway contains an HTTP response code. The HTTP header in the following example contains the HTTP response code 200 OK:

```
HTTP/1.1 200 OK
Server: WAS/6.0
Content-length: 0
```

The following table describes the possible values for the HTTP response code and an example of an associated WebSphere MQ Managed File Transfer error code that can be returned. For more information about the WebSphere MQ Managed File Transfer error codes, see Diagnostic messages.

Table 14. HTTP response codes

HTTP response code	Example WebSphere MQ Managed File Transfer error code	Example description
200 OK	None	A valid request has been handled correctly and optionally a response has been provided to the user.
202 Accepted	None	A valid request has been handled correctly but WebSphere MQ Managed File Transfer does not guarantee that the requested action has completed. For example, a file upload transfer request has been handled and submitted to a WebSphere MQ Managed File Transfer agent but the transfer has not yet taken place.
400 Bad Request	BFGWI0001	The URI is not valid because it is missing a resource type.
403 Forbidden	BFGWI0056	There is no WebSphere MQ Message Descriptor (MQMD) user identifier defined for the user.
404 Not Found	BFGWI0015	The requested resource cannot be found.
405 Method Not Allowed	BFGWI0016	The requested resource does not support the HTTP verb that has been used in the request. For example, a GET has been used against a resource that only allows POST or DELETE.
410 Resource Gone	BFGWI0031	The requested resource is no longer available. For example, the requested file has been deleted from the file space.
413 Request Entity Too Large	BFGWI0026	The request contains a file that is too large to be handled by the server.
415 Unsupported Media Type	BFGWI0017	A request has been received with a media type, specified by the `Content-type` HTTP header, that is not supported.

Table 14. HTTP response codes (continued)

HTTP response code	Example WebSphere MQ Managed File Transfer error code	Example description
500 Internal Server Error	BFGWI0018	An internal error has been encountered when handling the request. An FFDC or ABEND file has been produced.
502 Bad Gateway	BFGWI0019	The request cannot be completed because an error occurred outside WebSphere MQ Managed File Transfer. For example, a WebSphere MQ queue manager is not available.
503 Service Unavailable	BFGWI0020	The destination is temporarily unavailable. For example, a WebSphere MQ queue is full.
504 Gateway Timeout	BFGWI0021	An attempt to complete the request has timed out because of time limits imposed by WebSphere MQ Managed File Transfer, or because of time limits imposed by the HTTP client.

Troubleshooting the Connect:Direct bridge

Use the following reference information and examples to help you diagnose errors returned from the Connect:Direct bridge.

- "Tracing the Connect:Direct bridge"
- "Log information for the Connect:Direct bridge" on page 393
- "Solving permissions issues with Connect:Direct nodes" on page 393
- "What to do if text transfers to or from Connect:Direct nodes are not converting the data correctly" on page 394
- "Increasing the number of concurrent transfers for the Connect:Direct bridge" on page 395
- "Debugging a Connect:Direct process that is called by a file transfer" on page 396

Tracing the Connect:Direct bridge

You can capture trace from the Connect:Direct node that is part of the Connect:Direct bridge to help with problem determination.

About this task

To enable trace, complete the following steps:

Procedure

1. Stop the Connect:Direct bridge agent.
2. Edit the Connect:Direct bridge agent properties file to include the line:

   `cdTrace=true`
3. Start the Connect:Direct bridge agent.

Results

The trace information is written to the `output0.log` file in the Connect:Direct bridge agent configuration directory.

"The agent.properties file" on page 548
Each agent has its own properties file, `agent.properties`, that must contain the information that an agent uses to connect to its queue manager. The `agent.properties` file can also contain properties that alter the behavior of the agent.

Log information for the Connect:Direct bridge

You can use a Connect:Direct bridge agent to transfer files between MQMFT agents and Connect:Direct nodes. Log information about the Connect:Direct nodes and processes involved in these transfers is displayed in the WebSphere MQ Explorer plug-in and is stored in your log database.

The Connect:Direct bridge agent must be WebSphere MQ Managed File Transfer V7.0.4 or later. The other agent involved in the transfer can be any version of WebSphere MQ Managed File Transfer. However, for information about Connect:Direct nodes and processes to be logged, all MQMFT agents involved in the transfer must be V7.0.4 or later. For this information to be displayed in the WebSphere MQ Explorer plug-in, the plug-in must be V7.0.4 or later. For this information to be stored in the log database, the database logger and database schema must be V7.0.4 or later.

Log information about the Connect:Direct nodes and Connect:Direct processes involved in a file transfer is included in the log messages that are published to the SYSTEM.FTE topic on the coordination queue manager. For more information, see "File transfer log message formats" on page 626.

The following information is included in the published message:
• Connect:Direct bridge node name
• Primary node (PNODE) name
• Secondary node (SNODE) name
• Process name
• Process ID number

The Connect:Direct bridge node is the same node as either the primary node or the secondary node.

The value of the Connect:Direct bridge node name is the name that the bridge node is known to the MQMFT Connect:Direct bridge agent by. The primary and secondary node names are the names that are used to refer to the nodes in the network map of the Connect:Direct bridge node.

Related reference:
"Connect:Direct bridge transfer message examples" on page 647
The `destinationAgent` or `sourceAgent` element contains additional attributes when the destination agent or source agent is a Connect:Direct bridge agent. The Started log message contains only a subset of the information about the Connect:Direct transfer. The Progress and Completed log messages contain full information about the Connect:Direct transfer.

Solving permissions issues with Connect:Direct nodes

Use the information in this topic if your transfers between IBM WebSphere MQ Managed File Transfer and Connect:Direct fail with an error about insufficient permissions.

For transfers involving the Connect:Direct bridge, the user ID that connects to the Connect:Direct node is determined by which WebSphere MQ Message Descriptor (MQMD) user ID is associated with the transfer request. You can map specific MQMD user IDs to specific Connect:Direct user IDs. For more information, see "Mapping credentials for Connect:Direct" on page 193.

You might see transfers failing with one of the following errors:
• `BFGCD0001E: This task was rejected by the Connect:Direct API with the`
  `following error message: Connect:Direct Node detected error.`
  `LCCA000I The user has no functional authority to issue the selp command`

- BFGCD0026I: Connect:Direct messages: The submit of the process
  succeeded. Process number 1092 (name F35079AE, SNODE MYNODE)
  executing. User fteuser does not have permission to override SNODEID.
  User fteuser does not have permission to override SNODEID. User
  fteuser does not have permission to override SNODEID.

If you see either of these errors, determine which Connect:Direct user ID is associated with the MQMD user ID that was used for the transfer request. This Connect:Direct user ID must have authority to perform the Connect:Direct operations required by the Connect:Direct bridge. For the list of functional authorities needed, and guidance on how to grant these authorities, see "Mapping credentials for Connect:Direct by using the ConnectDirectCredentials.xml file" on page 193.

What to do if text transfers to or from Connect:Direct nodes are not converting the data correctly

When you transfer files in text mode between an MQMFT agent and a Connect:Direct node, code page and end-of-line character conversion is performed. The transfer uses the operating system information in the network map of the Connect:Direct bridge node to determine the end-of-line characters of a remote node. If the information in the network map is incorrect, the end-of-line character conversion might be performed incorrectly.

Ensure that the network map of the Connect:Direct bridge node and any Connect:Direct nodes that are used as a transfer destination include the correct platform description.

- If your Connect:Direct bridge node is on a Windows system, ensure that for each remote node in your network map you select the correct value from the **Operating System** list.
  - If the remote node is on a Windows system, select Windows.
  - If the remote node is on a UNIX or Linux system, select UNIX.

  Transfers to remote nodes on other operating systems are not supported by the Connect:Direct bridge.

- Ensure that for each remote node you transfer a file to or from, you specify the operating system type of the remote Connect:Direct node in the ConnectDirectNodeProperties.xml file in the Connect:Direct bridge agent configuration directory. For more information, see "Configure the ConnectDirectNodeProperties.xml file to include information about the remote Connect:Direct nodes" on page 192 and "Connect:Direct node properties file format" on page 582.

Related reference:

"Transferring text files between Connect:Direct and WebSphere MQ Managed File Transfer" on page 681 Text transfer involves converting the code page of a file from one code page to another. Text file transfer also involves converting CRLF (carriage return line feed) characters between systems. This topic summarizes text file transfer behavior in transfers between an MQMFT agent and a Connect:Direct node.

Connect:Direct file paths specified with a double forward slash

If, as part of a file transfer, you specify a file located on a Connect:Direct node by using a file path that starts with a double forward slash (//), the file is treated as a data set.

Sources and destinations on a Connect:Direct node are specified in the format *cd_node_name*:*file_path*. If the *file_path* starts with a double forward slash (//), the source or destination is treated as a data set. This is the case even when the Connect:Direct node is not on z/OS. This can cause transfer failures if the file path is accidentally specified with a double forward slash (//) at the start and the file is not a data set.

Ensure that you do not specify a *file_path* that starts with a double forward slash (//) if you do not want the file that you specify to be treated as a data set.

Related concepts:

"Troubleshooting the Connect:Direct bridge" on page 392
Use the following reference information and examples to help you diagnose errors returned from the Connect:Direct bridge.

Increasing the number of concurrent transfers for the Connect:Direct bridge

To increase the number of concurrent transfers that the Connect:Direct bridge agent can process, you must change three agent properties. You must also increase the maximum number of connections that the Connect:Direct node accepts.

The maximum number of concurrent transfers that a Connect:Direct bridge agent can process depends on the values of certain agent properties. The **maxSourceTransfers** and **maxDestinationTransfers** agent properties have a default value of five transfers for a Connect:Direct bridge agent. This default value is lower than the default of 25 transfers for other types of agent. A Connect:Direct bridge, where the agent is configured with the default values of **maxSourceTransfers** and **maxDestinationTransfers**, can process a maximum of 10 transfers at any one time: five transfers where the agent is the source, and five transfers where the agent is the destination.

These default values ensure that the Connect:Direct bridge agent does not exceed the maximum number of API connections to the Connect:Direct node. A Connect:Direct bridge agent with the default configuration uses a maximum of 10 API connections to the Connect:Direct node. The maximum number of connections accepted by a Connect:Direct node on UNIX is controlled by the **api.max.connects** Connect:Direct parameter. For a Connect:Direct node on Windows, the equivalent parameter is **max.api.connects**.

If the rate at which your Connect:Direct bridge carries out large numbers of file transfers is not sufficient, you can increase the number of concurrent transfers that the Connect:Direct bridge agent processes. Change the following agent properties for the Connect:Direct bridge agent:

maxSourceTransfers
> Set this property to a value that is larger than 5, but smaller than or equal to 25. If you choose a value that is larger than 25, the agent might run out of memory unless you increase the amount of memory that is available to the JVM used by the agent.

maxDestinationTransfers
> Set this property to a value that is larger than 5, but smaller than or equal to 25. If you choose a value that is larger than 25, the agent might run out of memory unless you increase the amount of memory that is available to the JVM used by the agent.

ioThreadPoolSize
> The default value of **ioThreadPoolSize** is 10. This property restricts the number of Connect:Direct node API connections for transfers where the Connect:Direct bridge agent is the source agent. These transfers are from Connect:Direct to WebSphere MQ Managed File Transfer. Use the following guidance to set the value of this property:
> - If the value of **maxSourceTransfers** is smaller than the value of **maxDestinationTransfers**, set **ioThreadPoolSize** to double the value of **maxSourceTransfers** or 10, whichever is the larger
> - If the value of **maxSourceTransfers** is larger than the value of **maxDestinationTransfers**, set **ioThreadPoolSize** to the sum of **maxSourceTransfers** and **maxDestinationTransfers**

In addition to these agent properties, you must also change the maximum number of concurrent API connections for the Connect:Direct node that is part of the Connect:Direct bridge. The Connect:Direct parameter that controls this number is **api.max.connects** if your node is on UNIX, or **max.api.connects** if your node is on Windows. Make the following changes to the appropriate parameter:

api.max.connects (if the node in your Connect:Direct bridge is on UNIX)

> Set this parameter to a value larger than the sum of **maxSourceTransfers** and **maxDestinationTransfers**. The default value of the **api.max.connects** parameter is 16. For more information about how to set this parameter, see the Connect:Direct documentation.

max.api.connects (if the node in your Connect:Direct bridge is on Windows)

> Set this parameter to a value larger than the sum of **maxSourceTransfers** and **maxDestinationTransfers**. The default value of the **max.api.connects** parameter is 10. For more information about how to set this parameter, see the Connect:Direct documentation.

Related tasks:

"Configuring the Connect:Direct bridge" on page 190
Configure the Connect:Direct bridge to transfer files between a WebSphere MQ Managed File Transfer network and a Connect:Direct network. The components of the Connect:Direct bridge are a Connect:Direct node and a WebSphere MQ Managed File Transfer agent that is dedicated to communicating with that node. This agent is referred to as the Connect:Direct bridge agent.

Related reference:

"The agent.properties file" on page 548
Each agent has its own properties file, `agent.properties`, that must contain the information that an agent uses to connect to its queue manager. The `agent.properties` file can also contain properties that alter the behavior of the agent.

Debugging a Connect:Direct process that is called by a file transfer

You can configure the Connect:Direct bridge agent to write log information about the Connect:Direct process that is called by a file transfer to the `output0.log` file in the Connect:Direct bridge agent configuration directory.

About this task

To configure logging of the Connect:Direct processes, complete the following steps:

Procedure

1. Stop the Connect:Direct bridge agent.
2. Edit the `agent.properties` file in the *MQ_DATA_PATH*/mqft/config/*coordination_queue_manager*/agents/*bridge_agent_name* directory to include the property `logCDProcess`. The `logCDProcess` property can have one of the following values:
   - None - No information is logged. This is the default.
   - Failures - Information about failed Connect:Direct processes is logged.
   - All - Information about all Connect:Direct processes is logged.
3. Start the Connect:Direct bridge agent.

Results

Information about Connect:Direct processes is logged to the Connect:Direct bridge agent's `output0.log` file. The information that is logged comprises:
- MQMFT transfer ID
- Connect:Direct process name
- Connect:Direct process number
- Generated process definition
- File name of the process template, if the Connect:Direct process is user-defined

Related concepts:

"Troubleshooting the Connect:Direct bridge" on page 392
Use the following reference information and examples to help you diagnose errors returned from the Connect:Direct bridge.

Related reference:

"The agent.properties file" on page 548
Each agent has its own properties file, `agent.properties`, that must contain the information that an agent uses to connect to its queue manager. The `agent.properties` file can also contain properties that alter the behavior of the agent.

Reference

Product overview

How does WebSphere MQ Managed File Transfer work?

WebSphere MQ Managed File Transfer interacts in a number of ways with WebSphere MQ. This topic describes how the two products interact.

- WebSphere MQ Managed File Transfer transfers files between agent processes by dividing each file into one or more messages and transmitting the messages through your WebSphere MQ network.

- The agent processes move file data by using nonpersistent messages to minimize the impact on your WebSphere MQ logs. By communicating with one another the agent processes regulate the flow of messages containing file data. This prevents messages containing file data building up on WebSphere MQ transmission queues and ensures that if any of the nonpersistent messages are not delivered, the file data is sent again.

- WebSphere MQ Managed File Transfer agents use a number of WebSphere MQ queues. For more information, see .

- Although some of these queues are strictly for internal use, an agent can accept requests in the form of specially formatted command messages sent to a specific queue that the agent reads from. Both the command-line commands and the WebSphere MQ Explorer plug-in send WebSphere MQ messages to the agent to instruct the agent to perform the wanted action. You can write WebSphere MQ applications that interact with the agent in this way. For more information, see "Controlling WebSphere MQ Managed File Transfer by putting messages on the agent command queue" on page 342.

- WebSphere MQ Managed File Transfer agents send information about their state and the progress and outcome of transfers to a WebSphere MQ queue manager that has been designated as the coordination queue manager. This information is published by the coordination queue manager and can be subscribed to by applications that want to monitor transfer progress or keep records of the transfers that have occurred. Both the command-line commands and the WebSphere MQ Explorer plug-in can use the information that is published. You can write WebSphere MQ applications that use this information. For more information about the topic that the information is published to, see "The SYSTEM.FTE topic" on page 610.

- Key components of WebSphere MQ Managed File Transfer take advantage of the capability of WebSphere MQ queue managers to store and forward messages. This means that if you suffer an outage, unaffected parts of your infrastructure can continue to transfer files. This extends to the coordination queue manager, where a combination of store and forward and durable subscriptions allow the coordination queue manager to tolerate becoming unavailable without losing key information about the file transfers that have taken place.

How to read railroad (syntax) diagrams

Each railroad diagram begins with a double right arrow and ends with a right and left arrow pair. Lines beginning with a single right arrow are continuation lines. You read a railroad diagram from left to right and from top to bottom, following the direction of the arrows.

Other conventions used in railroad diagrams are:

Table 15. How to read railroad diagrams
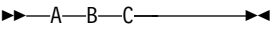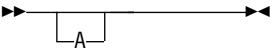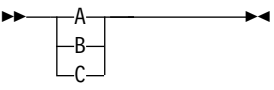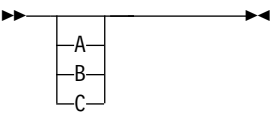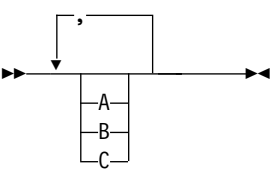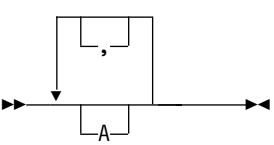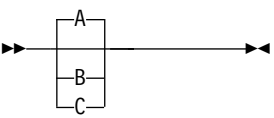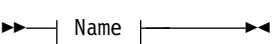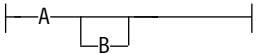
Convention	Meaning
►►—A—B—C————►◄	You must specify values A, B, and C. Required values are shown on the main line of a railroad diagram.
►►———————►◄ └A┘	You may specify value A. Optional values are shown below the main line of a railroad diagram.

►►——A———►◄
  ├B┤
  └C┘
	Values A, B, and C are alternatives, one of which you must specify.

►►————————►◄
  ├A┤
  ├B┤
  └C┘
	Values A, B, and C are alternatives, one of which you might specify.

┌—,←—┐
►►—┴—A—┴—►◄
  ├B┤
  └C┘
	You might specify one or more of the values A, B, and C. Any required separator for multiple or repeated values (in this example, the comma (,)) is shown on the arrow.

┌—┬—,—┬—┐ ►►—┴———┴—A—►◄	You might specify value A multiple times. The separator in this example is optional.

┌A┐
►►—┼——┼—►◄
  ├B┤
  └C┘
	Values A, B, and C are alternatives, one of which you might specify. If you specify none of the values shown, the default A (the value shown above the main line) is used.

►►—┤ Name ├—►◄ **Name:** ├—A———┤ └B┘	The railroad fragment Name is shown separately from the main railroad diagram.

Table 15. How to read railroad diagrams (continued)

Convention	Meaning
Punctuation and uppercase values	Specify exactly as shown.
Lowercase values (for example, *name*)	Supply your own text in place of the *name* variable.

Installing

WebSphere MQ Managed File Transfer hardware and software prerequisites

Before you install WebSphere MQ Managed File Transfer, check that your system meets both the hardware and software requirements of the product. For all platforms, you must have one WebSphere MQ Version 7.0 or higher queue manager available in your WebSphere MQ Managed File Transfer network to use as the coordination queue manager.

See WebSphere MQ System Requirements for hardware and software prerequisites.

Installed command sets

The following table shows which commands are installed with each component.

Table 16. WebSphere MQ Managed File Transfer commands available in each command set

Command	Agent command set	Service command set	Tools command set	Logger command set
fteAnt			X	
fteCancelTransfer			X	
fteChangeDefaultConfigurationOptions	X	X	X	X (Windows only)
fteCleanAgent	X	X		
fteCommon	X		X	
fteCreateAgent	X	X (Windows only)		
fteCreateBridgeAgent		X		
fteCreateCDAgent	X	X		
fteCreateLogger				X
fteCreateMonitor			X	
fteCreateTemplate			X	
fteCreateTransfer			X	
fteCreateWebAgent		X		
fteDeleteAgent	X	X		
fteDeleteLogger				X
fteDeleteMonitor			X	
fteDeleteScheduledTransfer			X	
fteDeleteTemplates			X	
fteDisplayVersion	X	X	X	X (Windows only)
fteListAgents	X	X	X	X (Windows only)
fteListMonitors			X	
fteListScheduledTransfers			X	
fteListTemplates			X	

Table 16. WebSphere MQ Managed File Transfer commands available in each command set (continued)

Command	Agent command set	Service command set	Tools command set	Logger command set
fteMigrateAgent	X	X		
fteMigrateConfigurationOptions	X	X	X	X (Windows only)
fteMigrateLogger				X
fteModifyAgent	X (Windows only)	X (Windows only)		
fteModifyLogger				X
fteObfuscate	X	X	X	X (Windows only)
ftePingAgent			X	
fteSetAgentTraceLevel	X	X		
fteSetLoggerTraceLevel				X
fteSetupCommands	X	X	X	X
fteSetupCoordination	X	X	X	X (Windows only)
fteShowAgentDetails	X	X	X	X (Windows only)
fteShowLoggerDetails				X
fteStartAgent	X	X		
fteStartLogger				X
fteStopAgent	X	X		
fteStopLogger				X

► V 7.5.0.2 ◄

Table 17. WebSphere MQ Managed File Transfer commands available in each command set

Command	Agent command set	Service command set	Tools command set	Logger command set
fteAnt			X	
fteBundleConfiguration			X (Distributed platforms only)	
fteCancelTransfer			X	
fteChangeDefaultConfigurationOptions	X	X	X	X
fteCleanAgent	X	X		
fteCreateAgent	X	X		
fteCreateBridgeAgent		X		
fteCreateCDAgent	X (Distributed platforms only)	X (Distributed platforms only)		
fteCreateLogger				X
fteCreateMonitor			X	
fteCreateTemplate			X	
fteCreateTransfer			X	
fteCreateWebAgent		X (Distributed platforms only)		
fteDefine			X (Distributed platforms only)	
fteDelete			X (Distributed platforms only)	
fteDeleteAgent	X	X		
fteDeleteLogger				X
fteDeleteMonitor			X	
fteDeleteScheduledTransfer			X	

Table 17. WebSphere MQ Managed File Transfer commands available in each command set (continued)

Command	Agent command set	Service command set	Tools command set	Logger command set
fteDeleteTemplates			X	
fteDisplayVersion	X	X		X
fteListAgents	X	X	X	X
fteListMonitors			X	
fteListScheduledTransfers			X	
fteListTemplates			X	
fteMigrateAgent	X	X		
fteMigrateConfigurationOptions	X	X	X	X
fteMigrateLogger				X
fteModifyAgent	X (Windows only)	X (Windows only)		
fteModifyLogger				X (Windows only)
fteObfuscate	X	X		X
ftePingAgent			X	
fteSetAgentTraceLevel	X	X		
fteSetLoggerTraceLevel				X
fteSetupCommands	X	X	X	X
fteSetupCoordination	X	X	X	X
fteShowAgentDetails	X	X	X	X
fteShowLoggerDetails				X
fteStartAgent	X	X		
fteStartLogger				X
fteStopAgent	X	X		
fteStopLogger				X

Security

File system permissions for WebSphere MQ Managed File Transfer in WebSphere MQ

When you install and configure the WebSphere MQ Managed File Transfer component of WebSphere MQ, the `configuration`, `installations`, and `logs` directories are created with the following permissions.

UNIX and Linux

Table 18. Summary of permissions for directories on UNIX and Linux

Directory	Permissions
`/var/mqm/mqft/config`	• Writable by the mqm group • World readable Users in the mqm group have write access to these directories and files
`/var/mqm/mqft/installations`	• Writable by the mqm group • World readable
`/var/mqm/mqft/logs`	World readable and writable

Windows

Table 19. Summary of permissions for directories on Windows

Directory	Permissions
`MQ_DATA_PATH\mqft\config`	The following users have full read and write access: • Administrators • System account • mqm group Other users have read access
`MQ_DATA_PATH\mqft\installations`	The following users have full read and write access: • Administrators • System account • mqm group Other users have read access
`MQ_DATA_PATH\mqft\logs`	The following users have full read and write access: • Administrators • System account • mqm group Other users have read and write access

Authorities for resources specific to WebSphere MQ Managed File Transfer

For any file transfer request, the agent processes require some level of access to their local file systems. In addition, both the user identifier associated with the agent process, and the user identifiers associated with users performing file transfer operations must have the authority to use certain WebSphere MQ objects.

Commands are issued by users, who might be in an operational role where they typically start a file transfer. Alternatively, they might be in an administrative role where they can additionally control when agents are created, started, deleted, or cleaned (that is, when messages from all agent system queues are removed). Messages containing command requests are placed on an agent's SYSTEM.FTE.COMMAND queue when a user issues a command. The agent process retrieves messages containing command requests from the SYSTEM.FTE.COMMAND queue. The agent process also uses four other system queues, which are as follows:

* SYSTEM.FTE.DATA.*agent_name*
* SYSTEM.FTE.EVENT.*agent_name*
* SYSTEM.FTE.REPLY.*agent_name*
* SYSTEM.FTE.STATE.*agent_name*

If an agent is a web agent it has two additional queues. These queues have the following names:

* SYSTEM.FTE.WEB.RESP.*agent_name*
* SYSTEM.FTE.WEB.*gateway_name*

Because users issuing commands use the queues listed previously in different ways to the agent process, assign different WebSphere MQ authorities to the user identifiers or user groups associated with each. See "Group authorities for resources specific to WebSphere MQ Managed File Transfer" on page 405 for more information.

The agent has additional queues that can be used to grant users the authority to perform certain actions. See "User authorities on WebSphere MQ Managed File Transfer actions" on page 409 for information about how to use the authority queues. The agent does not put or get messages on these queues. However, you must ensure that the queues are assigned the correct WebSphere MQ authorities both for the user identifier used to run the agent process as well as the user identifiers associated with users who are being authorized to perform certain actions. The authority queues are as follows:

- SYSTEM.FTE.AUTHADM1.*agent_name*
- SYSTEM.FTE.AUTHAGT1.*agent_name*
- SYSTEM.FTE.AUTHMON1.*agent_name*
- SYSTEM.FTE.AUTHOPS1.*agent_name*
- SYSTEM.FTE.AUTHSCH1.*agent_name*
- SYSTEM.FTE.AUTHTRN1.*agent_name*

If you are migrating from a version of WebSphere MQ File Transfer Edition earlier than V7.0.2 to IBM WebSphere MQ V7.5, and are keeping existing agent configurations, you will need to create the authority queues manually. Use the following MQSC command to create the queues:

```
DEFINE QLOCAL(authority_queue_name) DEFPRTY(0) DEFSOPT(SHARED) GET(ENABLED) MAXDEPTH(0) +
 MAXMSGL(0) MSGDLVSQ(PRIORITY) PUT(ENABLED) RETINTVL(999999999) SHARE  NOTRIGGER +
 USAGE(NORMAL) REPLACE
```

The agent process also publishes messages to the SYSTEM.FTE topic on the coordination queue manager using the SYSTEM.FTE queue. Depending on whether the agent process is in the role of the source agent or destination agent, the agent process might require authority to read, write, update, and delete files.

You can create and modify authority records for IBM WebSphere MQ objects using the IBM WebSphere MQ Explorer. Right-click the object and then click **Object Authorities** > **Manage Authority Records**. You can also create authority records using the **setmqaut** command, which is described at setmqaut (grant or revoke authority) command.

Related reference:

"Group authorities for resources specific to WebSphere MQ Managed File Transfer"
Instead of granting authority to individual users for all of the various objects that might be involved, configure two security groups for the purposes of administering WebSphere MQ Managed File Transfer access control: FTEUSER and FTEAGENT. It is the responsibility of the WebSphere MQ administrator to create and populate these groups. The administrator can choose to extend or modify the proposed configuration described here.

"User authorities on WebSphere MQ Managed File Transfer actions" on page 409
In addition to using groups to manage access to resources, you can enable an additional level of security to restrict the agent actions that a user can take. Grant authorities on an agent authority queue to a user to give the user permission to perform specific agent actions.

"Authorities for the logger" on page 412
The operating system user who runs the logger requires certain WebSphere MQ authorities on the logger queues and the SYSTEM.FTE topic.

Group authorities for resources specific to WebSphere MQ Managed File Transfer

Instead of granting authority to individual users for all of the various objects that might be involved, configure two security groups for the purposes of administering WebSphere MQ Managed File Transfer access control: FTEUSER and FTEAGENT. It is the responsibility of the WebSphere MQ administrator to create and populate these groups. The administrator can choose to extend or modify the proposed configuration described here.

Authority to connect to queue managers

Commands that are run by operational users, administrative users, and the WebSphere MQ Explorer need to be able to connect to the command queue manager and coordination queue manager. The agent process and commands that are run to create, alter, or delete the agent need to be able to connect to the agent queue manager.

- Grant the FTEUSER group connect authority for the command queue manager and coordination queue manager. For example:

For UNIX, Linux, and Windows systems:
```
setmqaut -m command_queue_manager -t qmgr -g FTEUSER +connect
setmqaut -m coordination_queue_manager -t qmgr -g FTEUSER +connect
```

- Grant the FTEAGENT group connect and inquire authority to the agent queue manager. For example:

For UNIX, Linux, and Windows systems:
```
setmqaut -m agent_queue_manager -t qmgr -g FTEAGENT +connect +inq +setid
```

For information about which command directly connects to which queue manager, see "Which WebSphere MQ Managed File Transfer command connects to which queue manager" on page 419

Authority to put a message on the COMMAND queue that belongs to the agent

The agent command queue must be available to any user who is authorized to request that the agent performs an action. To satisfy this requirement,

- Grant the FTEUSER group only put access to the SYSTEM.FTE.COMMAND.*agent_name* queue. For example:

For UNIX, Linux, and Windows systems:
```
setmqaut -m QM1 -n SYSTEM.FTE.COMMAND.agent_name -t queue -g FTEUSER +put
```

- Grant the FTEAGENT group put, get, and setid access to the SYSTEM.FTE.COMMAND.*agent_name* queue. For example:

For UNIX, Linux, and Windows systems:
```
setmqaut -m QM1 -n SYSTEM.FTE.COMMAND.agent_name -t queue -g FTEAGENT +browse +put +get +setid
```

Authority to put messages on the DATA, STATE, EVENT, and REPLY queues that belong to the agent

Only WebSphere MQ Managed File Transfer agents need to be able to use these system queues, therefore grant the group FTEAGENT put, get and inquire access. The names of these system queues are as follows:

- DATA - SYSTEM.FTE.DATA.*agent_name*
- STATE - SYSTEM.FTE.STATE.*agent_name*
- EVENT - SYSTEM.FTE.EVENT.*agent_name*
- REPLY - SYSTEM.FTE.REPLY.*agent_name*

For example, for the SYSTEM.FTE.DATA.*agent_name* queue, use a command like the following:

For UNIX, Linux, and Windows systems:
```
setmqaut -m QM1 -n SYSTEM.FTE.DATA.agent_name -t queue -g FTEAGENT +put +get +inq
```

Authority that the agent process runs under

The authority that the agent process runs under affects the files the agent can read and write from the file system, and the queues and topics the agent can access. How the authority is configured is system-dependent. Add the user ID that the agent process runs under to the FTEAGENT group. For more information about adding a user ID to a group, see Setting up security and navigate to the information for your operating system.

Authority that the commands and WebSphere MQ Explorer run under

Administrative commands, for example the **fteStartAgent** command, and the WebSphere MQ Managed File Transfer plug-in for the WebSphere MQ Explorer need to be able to put messages to the SYSTEM.FTE.COMMAND.*agent_name* queue and retrieve published information from that queue. Add the user IDs that are authorized to run the commands or the WebSphere MQ Explorer to the FTEUSER group. This originator user ID is recorded in the transfer log. For more information about adding a user ID to a group, see Setting up security and navigate to the information for your operating system.

Authority to put messages on the SYSTEM.FTE queue and SYSTEM.FTE topic

Only the agent process needs to be able to place messages on the SYSTEM.FTE queue and SYSTEM.FTE topic. Grant put, get and inquire authority to the FTEAGENT group on the SYSTEM.FTE queue, and grant publish and subscribe authority to the FTEAGENT group on the SYSTEM.FTE topic. For example:

For UNIX, Linux, and Windows systems:
```
setmqaut -m QM1 -n SYSTEM.FTE -t queue -g FTEAGENT +put +get +inq
setmqaut -m QM1 -n SYSTEM.FTE -t topic -g FTEAGENT +pub +sub +resume
```

For a message to get published to the SYSTEM.FTE topic, the authority records of the SYSTEM.FTE topic must allow publication by the user ID contained in the message descriptor structure (MQMD) of the message. This is described in Authority to publish log and status messages.

Authority to receive publications on the SYSTEM.FTE topic

Transfer log messages, progress messages, and status messages are intended for general use, so grant the FTEUSER group authority to subscribe to the SYSTEM.FTE topic. For example:

For UNIX, Linux, and Windows systems:
```
setmqaut -m QM1 -n SYSTEM.FTE -t topic -g FTEUSER +sub
```

Authority to connect to remote queue managers using transmission queues

In a topology of multiple queue managers, the agent requires put authority on the transmission queues used to connect to the remote queue managers.

Authority to create a temporary reply queue for file transfers

File transfer requests wait for the transfer to complete and rely on a temporary reply queue being created and populated. Grant the FTEUSER group DISPLAY, PUT, GET, and BROWSE authorities on the temporary model queue definition. For example:

For UNIX, Linux, and Windows systems:
```
setmqaut -m QM1 -n SYSTEM.DEFAULT.MODEL.QUEUE -t queue -g FTEUSER +dsp +put +get +browse
```

By default this is SYSTEM.DEFAULT.MODEL.QUEUE, but you can configure the name by setting values for the properties 'modelQueueName' and 'dynamicQueuePrefix' in the `command.properties` file.

The following table summarizes the access control configuration for FTEUSER and FTEAGENT in the security scheme described:

Table 20. Summary of access control configuration for FTEUSER and FTEAGENT

Object	Object type	FTEUSER	FTEAGENT
Agent queue manager	Queue manager		CONNECT, INQ, and SETID. ALT_USER is also required to enable user authority checking.
Coordination queue manager	Queue manager		
Command queue manager	Queue manager	CONNECT	CONNECT
SYSTEM.FTE	Local queue		GET and PUT
SYSTEM.FTE.COMMAND.*agent_name*	Local queue	PUT	BROWSE, GET, PUT, and SETID
SYSTEM.FTE.DATA.*agent_name*	Local queue		GET and PUT
SYSTEM.FTE.EVENT.*agent_name*	Local queue		BROWSE, GET and PUT
SYSTEM.FTE.REPLY.*agent_name*	Local queue		GET and PUT
SYSTEM.FTE.STATE.*agent_name*	Local queue		BROWSE, GET, INQ, and PUT
SYSTEM.FTE.WEB.*gateway_name*	Local queue		PUT
SYSTEM.FTE.WEB.RESP.*agent_name*	Local queue		GET
SYSTEM.FTE	Local topic	SUBSCRIBE	PUBLISH and SUBSCRIBE
SYSTEM.DEFAULT.MODEL.QUEUE (or the model queue defined in WebSphere MQ Managed File Transfer that is used to create a temporary reply queue.)	Model queue	BROWSE, DISPLAY, GET, and PUT	BROWSE, DISPLAY, GET, and PUT
Transmission queues to communicate with remote queue managers	Local queue		PUT

Authority to manage transfers through IBM WebSphere MQ Explorer

In addition to granting IBM WebSphere MQ Managed File Transfer authorities to users in situations that are already mentioned on this page, further authorities need to be granted to the IBM WebSphere MQ Managed File Transfer agent user who administers and performs all IBM WebSphere MQ Managed File Transfer operations through IBM WebSphere MQ Explorer. To issue commands such as create, cancel, schedule file transfer, create, delete resource monitors, and create transfer templates, the IBM WebSphere MQ Explorer user must have authority as follows:

- Coordination queue manager: connect, inquire, display
- Command queue manager: connect, inquire, display
- SYSTEM.FTE topic: publish, subscribe
- SYSTEM.MQEXPLORER.REPLY.MODEL: display, inquire, get, browse
- SYSTEM.ADMIN.COMMAND.QUEUE: inquire, put, display
- SYSTEM.DEFAULT.MODEL.QUEUE: get, put, inquire, display, browse

For information about which command directly connects to which queue manager, see "Which WebSphere MQ Managed File Transfer command connects to which queue manager" on page 419.

Related reference:

"User authorities on WebSphere MQ Managed File Transfer actions"
In addition to using groups to manage access to resources, you can enable an additional level of security to restrict the agent actions that a user can take. Grant authorities on an agent authority queue to a user to give the user permission to perform specific agent actions.

"Authorities for the logger" on page 412
The operating system user who runs the logger requires certain WebSphere MQ authorities on the logger queues and the SYSTEM.FTE topic.

User authorities on WebSphere MQ Managed File Transfer actions

In addition to using groups to manage access to resources, you can enable an additional level of security to restrict the agent actions that a user can take. Grant authorities on an agent authority queue to a user to give the user permission to perform specific agent actions.

Enabling user authority management

To turn on user authority checking on agent actions, complete the following steps:

1. In the `agent.properties` file, set the `authorityChecking` value to `true`.

2. Ensure that the user who runs the agent has the WebSphere MQ ALT_USER authority to the agent queue manager.

Both agents involved in a transfer must have the same level of security enabled, that is, `authorityChecking` must be set to the same value in the property files of both agents. Transfers between agents that have different values for the `authorityChecking` property will fail.

Agent authority queues

The agent has authority queues that are used to manage which users have the authority to perform certain agent actions. The agent does not put or get messages to these queues. The agent authority queues are as follows:

- SYSTEM.FTE.AUTHADM1.*agent_name*
- SYSTEM.FTE.AUTHAGT1.*agent_name*
- SYSTEM.FTE.AUTHMON1.*agent_name*
- SYSTEM.FTE.AUTHOPS1.*agent_name*
- SYSTEM.FTE.AUTHSCH1.*agent_name*
- SYSTEM.FTE.AUTHTRN1.*agent_name*

The authorities that a user has on the agent authority queues specify the actions that the user is authorized to take.

Table 21. The level of WebSphere MQ access authority that a user or group requires on an agent authority queue to perform specific actions.

User action	WebSphere MQ Managed File Transfer access authority	Authority queues	WebSphere MQ access authority (Distributed platforms)
Shut down the agent, using the `-m` option on **fteStopAgent** command.	Administration	SYSTEM.FTE.AUTHADM1.*agent_name*	BROWSE

Table 21. The level of WebSphere MQ access authority that a user or group requires on an agent authority queue to perform specific actions. (continued)

User action	WebSphere MQ Managed File Transfer access authority	Authority queues	WebSphere MQ access authority (Distributed platforms)
Start a transfer of files from this agent	Transfer source	SYSTEM.FTE.AUTHTRN1.*source_agent_name*	BROWSE
Cancel a transfer of files from this agent started by the same user			
Start a transfer of files to this agent	Transfer destination	SYSTEM.FTE.AUTHTRN1.*destination_agent_name*	PUT
Cancel a transfer of files to this agent started by the same user			
Create a resource monitor	Monitor	SYSTEM.FTE.AUTHMON1.*monitor_agent_name*	BROWSE
Delete a resource monitor created by the same user			
Delete a resource monitor created by any user	Monitor operations	SYSTEM.FTE.AUTHOPS1.*agent_name*	SET
Create a schedule	Schedule	SYSTEM.FTE.AUTHSCH1.*source_agent_name*	BROWSE
Delete a schedule created by the same user			
Delete a schedule created by any user or group	Schedule operations	SYSTEM.FTE.AUTHOPS1.*agent_name*	PUT
Cancel a transfer created by any user or group	Transfer operations	SYSTEM.FTE.AUTHOPS1.*source_agent_name* SYSTEM.FTE.AUTHOPS1.*destination_agent_name*	BROWSE

Note: To give a user or group permission to set up a resource monitor or schedule that starts a transfer the user needs both the **Monitor** or **Schedule** authority and **Transfer source** and **Transfer destination** authorities.

A user can start one agent and want it to interact with another agent. How the two agents can interact depends on the level of access authority that the user has on the other agent authority queue.

Table 22. The level of WebSphere MQ access authority that the user that starts an agent requires on another agent authority queue so that files can be transferred between the agents.

Agent action	WebSphere MQ Managed File Transfer access authority	Authority queues	WebSphere MQ access authority (Distributed platforms)
Receive a transfer from *<source_agent>*	Agent source	SYSTEM.FTE.AUTHAGT1.*source_agent_name*	BROWSE
Send a transfer to *<destination_agent>*	Agent destination	SYSTEM.FTE.AUTHAGT1.*destination_agent_name*	PUT

Configuring user authority management

To authorize a user to be able to perform an action on an agent, grant the user the appropriate authority on the relevant authority queue. To grant authorities to a user, complete the following steps:

1. Create a user on the system where the agent queue manager is located that has the same name as the user you want to give authority to perform agent actions. This user does not have to be active.

2. Grant the user the appropriate authority on the relevant authority queue. If you are using Linux, UNIX, or Windows, you can use the setmqaut command.

3. Refresh the security configuration of the queue manager. You can use the REFRESH SECURITY MQSC command.

Example

A user, who is a member of the group `requestor_group`, wants to set up a resource monitor on AGENT1 that transfers a file from AGENT1, which is running under the user `user1`, who is a member of the group `user1_group`, to AGENT2, which is running under the user `user2`, who is a member of the group `user2_group`. AGENT1 connects to QM1; AGENT2 connects to QM2. Both agents have authority checking enabled. To make this possible take the following steps:

1. `requestor` must have **Monitor** authority on AGENT1. Set this authority by running the following command on the system where QM1 is running:

   `setmqaut -m QM1 -t queue -n SYSTEM.FTE.AUTHMON1.AGENT1 -g requestor_group +browse`

2. `requestor` must have **Transfer source** authority on AGENT1. Set this authority by running the following command on the system where QM1 is running:

   `setmqaut -m QM1 -t queue -n SYSTEM.FTE.AUTHTRN1.AGENT1 -g requestor_group +browse`

3. `requestor` must have **Transfer destination** authority on AGENT2. Set this authority by running the following command On the system where QM2 is running:

   `setmqaut -m QM2 -t queue -n SYSTEM.FTE.AUTHTRN1.AGENT2 -g requestor_group +put`

4. `user2` must have **Agent source** authority on AGENT1. Set this authority by running the following command on the system where QM1 is running:

   `setmqaut -m QM1 -t queue -n SYSTEM.FTE.AUTHAGT1.AGENT1 -g user2_group +browse`

5. `user1` must have **Agent destination** authority on AGENT2. Set this authority by running the following command on the system where QM2 is running:

   `setmqaut -m QM2 -t queue -n SYSTEM.FTE.AUTHAGT1.AGENT2 -g user1_group +put`

Logging

If user authority checking is enabled, failed authority checks cause a not authorized log message to be published to the coordination queue manager. See "Message formats for security" on page 833 for more information.

Messages about user authority can be written to the agent event log. You can configure the amount of information written to the agent event log by setting the `logAuthorityChecks` property in the agent property file. By default the level of authority check logging is `None`. You can also set the value of `logAuthorityChecks` to `Failures`, which specifies that only failed authorization checks are reported, or `All` which specifies that failed and successful authorization checks are reported.

See "The agent.properties file" on page 548 for more information.

Related reference:

"Group authorities for resources specific to WebSphere MQ Managed File Transfer" on page 405
Instead of granting authority to individual users for all of the various objects that might be involved, configure two security groups for the purposes of administering WebSphere MQ Managed File Transfer access control: FTEUSER and FTEAGENT. It is the responsibility of the WebSphere MQ administrator to create and populate these groups. The administrator can choose to extend or modify the proposed configuration described here.

"Authorities for the logger"
The operating system user who runs the logger requires certain WebSphere MQ authorities on the logger queues and the SYSTEM.FTE topic.

"fteStopAgent (stop a WebSphere MQ Managed File Transfer agent)" on page 532
Use the `fteStopAgent` command to either stop a WebSphere MQ Managed File Transfer agent in a controlled way or to stop an agent immediately if necessary using the `-i` parameter.

Authorities for the logger

The operating system user who runs the logger requires certain WebSphere MQ authorities on the logger queues and the SYSTEM.FTE topic.

The operating system user who runs the logger requires the following WebSphere MQ authorities:
* CONNECT and INQUIRE on the coordination queue manager.
* SUBSCRIBE permission on the SYSTEM.FTE topic.
* PUT permission on the SYSTEM.FTE.LOG.RJCT.*logger_name* queue.
* GET permission on the SYSTEM.FTE.LOG.CMD.*logger_name* queue.

Related reference:

"Group authorities for resources specific to WebSphere MQ Managed File Transfer" on page 405
Instead of granting authority to individual users for all of the various objects that might be involved, configure two security groups for the purposes of administering WebSphere MQ Managed File Transfer access control: FTEUSER and FTEAGENT. It is the responsibility of the WebSphere MQ administrator to create and populate these groups. The administrator can choose to extend or modify the proposed configuration described here.

"User authorities on WebSphere MQ Managed File Transfer actions" on page 409
In addition to using groups to manage access to resources, you can enable an additional level of security to restrict the agent actions that a user can take. Grant authorities on an agent authority queue to a user to give the user permission to perform specific agent actions.

Permissions for configuration files containing sensitive information

Any file used to store sensitive configuration information, meaning any file referenced from the WebSphere MQ configuration tree, must not have system-wide read, write, or (where applicable), delete permissions. These restrictions are also be applied to truststore and keystore files.

If a WebSphere MQ Managed File Transfer process detects a condition that a configuration file contains sensitive information, is a keystore or truststore file, and has system-wide read, write, or delete permissions, the process takes one of the following actions:
* Fails to start, if the condition is detected at startup time.
* Generates a warning message and ignores the contents of the configuration file if the condition was detected at runtime. This is relevant to the protocol bridge and the Connect:Direct bridge, which reload a configuration if it changes while the process is running.

On systems with a UNIX type file system

The criteria for determining that a file has unacceptable system-wide permissions are:
* The others class has been granted read permission on the file
* The others class has been granted write permission on the file

- The others class has been granted write permission on the directory containing the file

On Windows systems

The criteria for determining that a file has unacceptable system-wide permissions are:
- Any of the Everyone, Guests, or Users groups have any of the following permissions:
  - Read data permission on the file
  - Append data permission on the file
  - Write data permission on the file
- Any of the Everyone, Guests, or Users groups has Create files permission on the folder containing the file and they also have any of the following permissions:
  - Delete subfolders and files permission on the folder containing the file
  - Delete permission on the file

Authority to publish log and status messages

Agents issue various log, progress, and status messages that are published on the coordination queue manager. The publication of these messages is subject to the WebSphere MQ security model, and in some cases you might have to perform further configuration to enable publication.

For more information about WebSphere MQ security, see the information starting with Security.

WebSphere MQ Managed File Transfer agents flow messages for publication to the SYSTEM.FTE queue on the coordination queue manager. Each message carries a user ID in its message descriptor (MQMD). Messages are published using a topic object that is also called SYSTEM.FTE. For the publication of a given message to take place, the authority records of the SYSTEM.FTE topic must permit publication by the user ID contained in the MQMD of the message.

The user ID initially contained in the message depends on how the agent is connected to its own queue manager. Messages from bindings-connected agents contain the user ID that the agent is running under. Messages from client-connected agents contain an internal WebSphere MQ user ID.

You can change the user ID in a message. For both client- and bindings-connected agents, you can use the property publicationMDUser (in the `agent.properties` file) to specify a user ID, which is used in all log and status messages from that agent. The agent must be given permission by its own queue manager to use this alternative user ID; give this permission by granting setid authority to the user ID that the agent runs under.

You can also change the user ID contained in all messages from a client-connected agent using the MCAUSER property on the channel that the agent uses to connect to its queue manager.

You can change the user ID in messages using a channel exit, for example on the receiver channel bringing messages into the coordination queue manager.

Depending on the WebSphere MQ topology and policies, there are a number of ways an WebSphere MQ administrator can use the information in this topic to ensure that the publication of status and log messages takes place. Two examples are:
- Determine all the user IDs used by agents in the network. Explicitly grant an authority record for each of these IDs.
- Create one or more common user names to publish log and status messages. Create authority records for these user names on the coordination queue manager. Set the publicationMDUser property for each agent to a common user name. On each agent queue manager, grant setid authority to the user ID that the agent runs under to allow it to accept the publicationMDUser property.

Authorities to access file systems

For any file transfer request, the agent processes require some level of access to their local file systems.

- To transfer from a source file, the user ID that the source agent runs under must have read access to the source file. Additionally, you might need to give the source agent delete or write authority depending on the source disposition attribute.
- To transfer to a file or directory, the user ID that the destination agent runs under must have write authority to the specified path. Additionally, you might need to give the destination agent update authority, depending on the destination exists attribute.
- In addition to the file access authority that you grant to the agent process, you can also use sandboxing to specify and enforce a restricted file path area. For more information, see "Sandboxes" on page 103.
- If the files that you want to transfer to or from are not in a location accessible to the agent, for example in a location that is restricted by the sandboxing capability, you can use WebSphere MQ Managed File Transfer user exits to move the file to or from a location that can be accessed by the agent. For more information, see "Customizing WebSphere MQ Managed File Transfer with user exit routines" on page 330.

The commandPath property

Use the commandPath property to restrict the locations that WebSphere MQ Managed File Transfer can run commands from.

You can specify a command to be run on the system where the agent is running from the managed transfer and managed call functions of WebSphere MQ Managed File Transfer. See Program invocation for information. However, commands must be on paths referenced by the commandPath agent property.

If the command specified is not fully qualified, WebSphere MQ Managed File Transfer attempts to find a matching command on the command path. If there is more than one matching command on the command path, the first match is used.

By default, the commandPath property is empty so that the agent cannot call any commands. Take extreme care when you set this property because any command in one of the specified commandPaths can effectively be called from a remote client system that is able to send commands to the agent. For this reason, by default, when you specify a commandPath, sandboxing is configured so that all commandPath directories are automatically denied access for a transfer. You can set the sandboxRoot property to override this default behavior, but you are not recommended to do so, because this effectively enables a client to transfer any command to the agent's system and call that command.

Specify the commandPath agent property as follows:

`commandPath=`*command_directory_name separator...command_directory_name*

where:

- *command_directory_name* is a directory path for commands that can be run.
- *separator* is the platform-specific separator.

For example, on a UNIX system if you want to run commands that are located in the directories /home/user/cmds1 and /home/user/cmds2, set the commandPath agent property as follows:

`commandPath=/home/user/cmds1:/home/user/cmds2`

For example, on a Windows system if you want to run commands that are located in the directories `C:\File Transfer\commands` and `C:\File Transfer\agent commands`, set the commandPath agent property as follows:

`commandPath=C:\\File Transfer\\commands;C:\\File Transfer\\agent commands`

On a Windows system the separator character, backslash (\), must be escaped and be entered as a double backslash (\\). The backslash character (\) can also be replaced with a forward slash (/).

For example, on an IBM 4690 system the separator character is a semicolon (;). To run commands that are located in the directories f:/fteuser/cmds and mqftcmds:/public where mqftcmds is an IBM 4690 logical name defined to a directory that contains the commands, set the commandPath agent property as follows:

```
commandPath=f:/fteuser/cmds;mqftcmds:/public
```

The commandPath property is described in Table 34 on page 550.

Summary of the WebSphere MQ Managed File Transfer commands

All WebSphere MQ Managed File Transfer commands are listed with links to their detailed descriptions.

Table 23. WebSphere MQ Managed File Transfer commands and their purpose

Command name	Purpose
Commands for migration:	
fteMigrateAgent	Migrate an agent and its configuration from WebSphere MQ Managed File Transfer V7.0 to IBM WebSphere MQ V7.5
fteMigrateLogger	Migrate the configuration of a stand-alone database logger from WebSphere MQ Managed File Transfer V7.0.1 or later to IBM WebSphere MQ V7.5
Commands for configuration:	
fteChangeDefaultConfigurationOptions	Change the default configuration options that you want WebSphere MQ Managed File Transfer to use
fteCreateAgent	Create a WebSphere MQ Managed File Transfer agent
fteCreateWebAgent	Create a WebSphere MQ Managed File Transfer web agent
fteCreateBridgeAgent	Create a WebSphere MQ Managed File Transfer protocol bridge agent
fteCreateCDAgent	Create a WebSphere MQ Managed File Transfer Connect:Direct bridge agent
`V 7.5.0.2` fteDefine	Generate the configuration scripts necessary to define the specified objects.
`V 7.5.0.2` fteDelete	Generate the configuration scripts necessary to remove the specified objects.
fteModifyAgent	Windows only. Modify an agent, web agent, Connect:Direct bridge agent, or protocol bridge agent to run as a Windows service.
fteDeleteAgent	Delete a particular WebSphere MQ Managed File Transfer agent
fteSetupCommands	Specify the details of the queue manager that connects to the WebSphere MQ network when you issue commands
fteSetupCoordination	Configure a WebSphere MQ Managed File Transfer coordination queue manager
fteCreateLogger	Create a WebSphere MQ Managed File Transfer logger
fteModifyLogger	Windows only. Modify the logger to run as a Windows service.
fteDeleteLogger	Delete a WebSphere MQ Managed File Transfer logger
Commands for administration:	
fteAnt	Run an Ant script in an environment with file transfer Ant tasks available.
fteCancelTransfer	Cancel a file transfer
fteCleanAgent	Clean up the queues used by an agent
fteCreateMonitor	Create and start a new resource monitor
fteCreateTemplate	Create a transfer template for future use

Table 23. WebSphere MQ Managed File Transfer commands and their purpose (continued)

Command name	Purpose
fteCreateTransfer	Create and start a new file transfer
fteDeleteMonitor	Stop and remove an existing resource monitor
fteDeleteTemplates	Delete existing file transfer templates
fteDeleteScheduledTransfer	Delete a particular file transfer that you have previously scheduled
fteListAgents	List all of the agents registered against a particular coordination queue manager
fteListMonitors	List all of the resource monitors registered against a particular coordination queue manager
fteListScheduledTransfers	List all of the WebSphere MQ Managed File Transfer transfers that you previously created using the command line or the WebSphere MQ Explorer.
fteListTemplates	List all the file transfer templates for a coordination queue manager
ftePingAgent	Pings an agent to determine whether the agent is active and able to process transfers.
fteShowAgentDetails	Display the details of a particular agent
fteStartAgent	Start a particular agent before using it to transfer files
fteStartLogger	Start logger
fteStopAgent	Stop a particular agent
fteStopLogger	Stop logger
Command for security:	
fteObfuscate	Encrypt sensitive data in credentials files.
Commands for troubleshooting:	
fteSetAgentTraceLevel	Set the level of agent trace to run
fteSetLoggerTraceLevel	Set the level of logger trace to run
fteDisplayVersion	Display the product version

See "Installed command sets" on page 401 for a table showing which commands are installed with which WebSphere MQ Managed File Transfer offering.

The syntax for each command and its parameters is presented in the form of a syntax diagram called a railroad diagram. For information about how to interpret railroad diagrams, see How to read railroad diagrams.

Authority to use WebSphere MQ Managed File Transfer commands

Your user ID must be a member of the mqm group if you want to issue WebSphere MQ Managed File Transfer commands, unless you have already configured WebSphere MQ to allow users who are not in the mqm group to issue commands.

For more information about authorization, see Authority to administer WebSphere MQ.

A subset of the WebSphere MQ Managed File Transfer commands can be issued using the WebSphere MQ Explorer.

Issuing commands from Windows and UNIX systems

Note the following environment-specific information for issuing commands:

WebSphere MQ Managed File Transfer for Windows

All commands can be issued from a command line. Command names are not case-sensitive: You can enter them in uppercase, lowercase, or a combination of uppercase and lowercase. However, arguments to control commands (such as queue names) and parameters (such as **-m** for queue manager name) are case-sensitive.

In the syntax descriptions, the hyphen (-) is used as a flag indicator.

WebSphere MQ Managed File Transfer for UNIX systems

All WebSphere MQ Managed File Transfer commands can be issued from a shell. All commands are case-sensitive.

Issuing commands from IBM 4690 systems

The WebSphere MQ Managed File Transfer commands detailed in these reference topics are not applicable to the IBM 4690 environment. For more information on using WebSphere MQ Managed File Transfer in the IBM 4690 environment, see "Using WebSphere MQ Managed File Transfer in a retail environment" on page 35

Related reference:

"Return codes for WebSphere MQ Managed File Transfer" on page 373
WebSphere MQ Managed File Transfer commands, Ant tasks, and log messages provide return codes to indicate whether functions have successfully completed.

Installed command sets

The following table shows which commands are installed with each component.

Table 24. WebSphere MQ Managed File Transfer commands available in each command set

Command	Agent command set	Service command set	Tools command set	Logger command set
fteAnt			X	
fteCancelTransfer			X	
fteChangeDefaultConfigurationOptions	X	X	X	X (Windows only)
fteCleanAgent	X	X		
fteCommon	X		X	
fteCreateAgent	X	X (Windows only)		
fteCreateBridgeAgent		X		
fteCreateCDAgent	X	X		
fteCreateLogger				X
fteCreateMonitor			X	
fteCreateTemplate			X	
fteCreateTransfer			X	
fteCreateWebAgent		X		
fteDeleteAgent	X	X		
fteDeleteLogger				X
fteDeleteMonitor			X	
fteDeleteScheduledTransfer			X	
fteDeleteTemplates			X	
fteDisplayVersion	X	X	X	X (Windows only)
fteListAgents	X	X	X	X (Windows only)
fteListMonitors			X	
fteListScheduledTransfers			X	
fteListTemplates			X	

Table 24. WebSphere MQ Managed File Transfer commands available in each command set (continued)

Command	Agent command set	Service command set	Tools command set	Logger command set
fteMigrateAgent	X	X		
fteMigrateConfigurationOptions	X	X	X	X (Windows only)
fteMigrateLogger				X
fteModifyAgent	X (Windows only)	X (Windows only)		
fteModifyLogger				X
fteObfuscate	X	X	X	X (Windows only)
ftePingAgent			X	
fteSetAgentTraceLevel	X	X		
fteSetLoggerTraceLevel				X
fteSetupCommands	X	X	X	X
fteSetupCoordination	X	X	X	X (Windows only)
fteShowAgentDetails	X	X	X	X (Windows only)
fteShowLoggerDetails				X
fteStartAgent	X	X		
fteStartLogger				X
fteStopAgent	X	X		
fteStopLogger				X

▶ V 7.5.0.2

Table 25. WebSphere MQ Managed File Transfer commands available in each command set

Command	Agent command set	Service command set	Tools command set	Logger command set
fteAnt			X	
fteBundleConfiguration			X (Distributed platforms only)	
fteCancelTransfer			X	
fteChangeDefaultConfigurationOptions	X	X	X	X
fteCleanAgent	X	X		
fteCreateAgent	X	X		
fteCreateBridgeAgent		X		
fteCreateCDAgent	X (Distributed platforms only)	X (Distributed platforms only)		
fteCreateLogger				X
fteCreateMonitor			X	
fteCreateTemplate			X	
fteCreateTransfer			X	
fteCreateWebAgent		X (Distributed platforms only)		
fteDefine			X (Distributed platforms only)	
fteDelete			X (Distributed platforms only)	
fteDeleteAgent	X	X		
fteDeleteLogger				X
fteDeleteMonitor			X	
fteDeleteScheduledTransfer			X	

Table 25. WebSphere MQ Managed File Transfer commands available in each command set (continued)

Command	Agent command set	Service command set	Tools command set	Logger command set
fteDeleteTemplates			X	
fteDisplayVersion	X	X		X
fteListAgents	X	X	X	X
fteListMonitors			X	
fteListScheduledTransfers			X	
fteListTemplates			X	
fteMigrateAgent	X	X		
fteMigrateConfigurationOptions	X	X	X	X
fteMigrateLogger				X
fteModifyAgent	X (Windows only)	X (Windows only)		
fteModifyLogger				X (Windows only)
fteObfuscate	X	X		X
ftePingAgent			X	
fteSetAgentTraceLevel	X	X		
fteSetLoggerTraceLevel				X
fteSetupCommands	X	X	X	X
fteSetupCoordination	X	X	X	X
fteShowAgentDetails	X	X	X	X
fteShowLoggerDetails				X
fteStartAgent	X	X		
fteStartLogger				X
fteStopAgent	X	X		
fteStopLogger				X

Which WebSphere MQ Managed File Transfer command connects to which queue manager

Different WebSphere MQ Managed File Transfer commands connect to different queue managers when you run the command.

The following table summarizes which queue manager each WebSphere MQ Managed File Transfer command directly connects to when the command is run.

If there are no X characters for a command, the command does not connect to any queue manager when it is run.

Table 26. Summary of which WebSphere MQ Managed File Transfer commands connect to which queue manager

Command name	Agent queue manager	Command queue manager	Coordination queue manager
fteAnt			
fteCancelTransfer		X	
fteChangeDefaultConfigurationOptions			
fteCleanAgent	X		
fteCreateAgent	X		
fteCreateBridgeAgent	X		
fteCreateCDAgent	X		

Table 26. Summary of which WebSphere MQ Managed File Transfer commands connect to which queue manager (continued)

Command name	Agent queue manager	Command queue manager	Coordination queue manager
fteCreateWebAgent	X		
fteCreateMonitor		X	
fteCreateTemplate			X
fteCreateTransfer		X	
fteDeleteAgent	X		
fteDeleteLogger			
fteDeleteMonitor		X	
fteDeleteScheduledTransfer		X	
fteDeleteTemplates			X
fteDisplayVersion			
fteListAgents			X
fteListMonitors			X
fteListScheduledTransfers			X
fteListTemplates			X
fteMigrateAgent			
fteMigrateConfigurationOptions			
fteMigrateLogger			
fteModifyAgent			
fteModifyLogger			
fteObfuscate			
ftePingAgent		X	
fteSetAgentTraceLevel			
fteSetLoggerTraceLevel			
fteSetupCommands			
fteSetupCoordination			
fteShowAgentDetails			X
fteShowLoggerDetails			
fteStartAgent	X		
fteStartLogger			
fteStopAgent		X	
fteStopLogger			X
WebSphere MQ Managed File Transfer plug-in for WebSphere MQ Explorer		X	X
WebSphere MQ Managed File Transfer logger			X

fteAnt (run Ant tasks in a WebSphere MQ Managed File Transfer environment)

The **fteAnt** command runs Ant scripts in an environment that has WebSphere MQ Managed File Transfer Ant tasks available.

Purpose

Use the **fteAnt** command to run an Ant script in an environment with WebSphere MQ Managed File Transfer. Unlike the standard **ant** command, **fteAnt** requires that you define a script file.

The **fteAnt** command cannot be run directly on an IBM 4690 system. However, an IBM 4690 system can be referred to by an Ant script. For more information about using WebSphere MQ Managed File Transfer in the IBM 4690 environment, see "Using WebSphere MQ Managed File Transfer in a retail environment" on page 35

Syntax

fteAnt

```
►►──fteAnt────────────────────────────────────────────────────────────────────────────►
           └──-d─────┘   └──-q─────┘   └──-v──────┘   └──-k──────────┘
           └──-debug─┘   └──-quiet─┘   └──-verbose─┘   └──-keep-going─┘

►──────────────────────────────────────────────┬──-f────(Ant script)────────┬──────────────────►◄
   └──-D──property=value─┘   └──-propertyfile──(name)─┘  ├──-file──(Ant script)────────┤   └──(targets)─┘
                                                          └──-buildfile──(Ant script)──┘
```

Parameters

-debug or -d
    Optional. Generate debugging output.

-quiet or -q
    Optional. Generate minimal output.

-verbose or -v
    Optional. Generate verbose output.

-keep-going or -k
    Optional. Execute all targets that do not depend on failed targets.

-D *property=value*
    Optional. Use *value* for a given *property*. Properties that are set with **-D** take precedence over those set in a properties file.

    Use the property **com.ibm.wmqfte.propertyset** to specify the set of configuration options that are used for Ant tasks. Use the name of a non-default coordination queue manager as the value for this property. Ant tasks then use the set of configuration options that are associated with this non-default coordination queue manager. If you do not specify this property, the default set of configuration options that are based on the default coordination queue manager is used. If you specify the **cmdqm** attribute for an Ant task, this attribute takes precedence over the set of configuration options that are specified for the **fteAnt** command. This behavior applies regardless of whether you are using the default set of configuration options or specifying a set with the **com.ibm.wmqfte.propertyset** property.

-propertyfile (*name***)**
　　Optional. Load all properties from a file with **-D** properties taking precedence.

-f (*Ant script***), -file (***Ant script***), or -buildfile (***Ant script***)**
　　Required. Specifies the name of the Ant script to run.

targets
　　Optional. The name of one or more targets to run from the Ant script. If you do not specify a value
　　for this parameter, the default target for the script is run.

-version
　　Optional. Displays the WebSphere MQ Managed File Transfer command and Ant versions.

-? or -h
　　Optional. Displays command syntax.

Example

In this example, the target **copy** in Ant script `fte_script.xml` is run and the command writes debugging
output to standard out.

```
fteAnt -d -f fte_script.xml copy
```

Return codes

0　　　　Command completed successfully.

1　　　　Command ended unsuccessfully.

Other status return codes can also be specified from Ant scripts, for example by using the Ant fail task.

fteCommon and ftePlatform scripts

The fteCommon and ftePlatform are scripts that are provided by WebSphere MQ Managed File Transfer
in the *MQ_INSTALLATION_PATH*/bin directory as helper scripts. Not all of these scripts are present on every
platform.

fteCommon

fteCommon is a helper script started by the other WebSphere MQ Managed File Transfer command
scripts to perform common setup processing before starting Java.

ftePlatform

ftePlatform is a helper script started by the fteCommon script to perform platform-specific setup
processing.

fteCancelTransfer (cancel a WebSphere MQ Managed File Transfer transfer)

Use the **fteCancelTransfer** command to cancel a WebSphere MQ Managed File Transfer transfer. You can
issue this command against either the source or destination agent for the transfer.

Purpose

If you issue the **fteCancelTransfer** command while that transfer is currently in progress, any files
already transferred as part of that transfer remain on the destination system and are not deleted. Any
files partially transferred as part of that transfer are deleted from the destination system. The destination
side of the transfer logs that transfer as "cancelled".

If a transfer to a Connect:Direct node is canceled, any files partially transferred as part of the canceled transfer remain on the destination system and are not deleted.

You can run the **fteCancelTransfer** command from any system that can connect to the WebSphere MQ network and then route to the agent queue manager. Specifically for the command to run, you must have installed WebSphere MQ Managed File Transfer on this system and you must have configured WebSphere MQ Managed File Transfer on this system to communicate with the WebSphere MQ network. If no connectivity details are available, the agent queue manager details are used for connection instead, provided these details are available.

Specify the optional **-p** parameter for this command only if you want to use a set of configuration options different from your default set. See Configuration options for more information.

Syntax

fteCancelTransfer

```
►►─fteCancelTransfer──────────────────────────────────────────────────────────────────────────────────────────►
                     └─m──(agent_qmgr_name)─┘  └─p──(configuration_options)─┘  ─a──agent_name──────►

►──transfer_ID─────────────────────────────────────────────────────────────────────────────────────────────►◄
```

Parameters

-m (_agent_qmgr_name_**)**
> Optional. The name of the agent queue manager. This agent must be either the source or destination agent for the transfer you want to cancel. If you do not specify this parameter, the cancel request is sent to the queue manager identified by the set of configuration options you are using.

-p (_configuration_options_**)**
> Optional. This parameter determines the set of configuration options to use to cancel the transfer. By convention use the name of a non-default coordination queue manager as the input for this parameter. The command then uses the set of properties files associated with this non-default coordination queue manager.
>
> If you do not specify this parameter, the set of configuration options based on the default coordination queue manager is used.

-a (_agent_name_**)**
> Required. The name of either the source or destination agent of the transfer that you want to cancel.

_transfer_ID_
> Required. The ID of the transfer you want to cancel. The transfer ID (also known as the request ID) is displayed at the command line after you issue the **fteCreateTransfer** command. Transfer IDs are also included in file transfer log messages or are displayed in the WebSphere MQ Explorer Transfer Log panel.

-? or -h
> Optional. Displays command syntax.

Example

In this example AGENT1 is the source agent for the transfer to be canceled.
```
fteCancelTransfer -a AGENT1 414d5120514d5f4c4d343336303920201159c54820027102
```

Return codes

0   Command completed successfully.

1 Command ended unsuccessfully.

Related reference:

"**fteCreateTransfer** (create new file transfer)" on page 461
The **fteCreateTransfer** command creates and starts a new file transfer from the command line. This command can start a file transfer immediately, schedule a file transfer for a future time and date, repeat a scheduled transfer one or more times, and trigger a file transfer based on certain conditions.

fteChangeDefaultConfigurationOptions (change the default configuration options)

Use the **fteChangeDefaultConfigurationOptions** command to change the default configuration options that you want WebSphere MQ Managed File Transfer to use. The value of the configuration options defines the group of properties files that WebSphere MQ Managed File Transfer uses.

Important: Only users who are WebSphere MQ administrators (and members of the mqm group) can run this command. If you try to run this command as a user who is not a WebSphere MQ administrator, you will receive an error message and the command will not run.

Purpose

Your default WebSphere MQ Managed File Transfer configuration options are established during installation and are based on your default coordination queue manager. By using the **fteChangeDefaultConfigurationOptions** command you can change the default coordination queue manager that is defined in the `installation.properties` file. If you change this coordination queue manager, WebSphere MQ Managed File Transfer uses the configuration options given by the structured set of directories and property files contained it the directory you used as input for *configuration_options* by default. This directory name is the same as the coordination queue manager used by agents under this configuration.

See "Configuration options" on page 124 for more information about the `installation.properties` file.

Syntax

fteChangeDefaultConfigurationOptions

►►—fteChangeDefaultConfigurationOptions—*configuration_options*—————————————►◄

Parameters

configuration_options
    Required. This parameter specifies the default configuration options that you want to change to. Use the name of a non-default coordination queue manager as the input for this parameter.

-? or -h
    Optional. Displays command syntax.

Example

In this example, the default configuration options are changed to QM_COORD2:
```
fteChangeDefaultConfigurationOptions QM_COORD2
```

Return codes

0   Command completed successfully.

1   Command ended unsuccessfully.

Related concepts:

"Configuration options" on page 124
WebSphere MQ Managed File Transfer provides a set of properties files that contain key information
about your setup and are required for operation. These properties files are in the configuration directory
that you defined when you installed the product.

fteCleanAgent (cleans up a WebSphere MQ Managed File Transfer agent)

Use the **fteCleanAgent** command to clean up the queues that a WebSphere MQ Managed File Transfer
agent uses, by deleting messages from the persistent and non-persistent queues used by the agent. Use
the **fteCleanAgent** command if you are having problems starting an agent, which might be caused by
information remaining on the queues used by the agent.

Purpose

Use the **fteCleanAgent** command to delete messages from the persistent and non-persistent queues used
by the agent. Specifically, this command can carry out the following actions:

- Remove any transfers that were in progress to this agent or from this agent before the transfer was
  stopped. These transfers are *not* resumed when the agent restarts
- Remove any commands that have already been submitted to the agent, but have not yet been carried
  out
- Delete all resource monitors stored on the agent
- Delete all scheduled transfers stored on the agent
- Delete all invalid messages stored on the agent

If the agent is a Connect:Direct bridge agent, the **-ms**, **-ss**, and **-ims** parameters are not valid. For
Connect:Direct bridge agents the command also carries out the following actions:

- Deletes all files from the directory where the Connect:Direct bridge agent temporarily stores files while
  they are being transferred. The location of this directory is defined by the **cdTmpDir** parameter
- Displays information about the Connect:Direct processes that are associated with any ongoing transfers

Before IBM WebSphere MQ Version 7.5.0, Fix Pack 8, if you run the **fteCleanAgent** command with just
the **agent_name** parameter, by default the command runs as if the **-all** parameter has been supplied,
which results in all the scheduled transfers, the resource monitor and scheduled transfer definitions on
the agent being cleared.

From IBM WebSphere MQ Version 7.5.0, Fix Pack 8, you must, by default, specify which WebSphere MQ
Managed File Transfer state to clear by passing the appropriate parameters to the **fteCleanAgent**
command, as well as providing an agent name. This means that, by default, **fteCleanAgent** does not clear
all in-progress and pending transfers, resource monitor definitions and scheduled transfer definitions for
the agent specified. You can enable or disable this behavior by setting the
failCleanAgentWithNoArguments property in the command.properties file to the appropriate value:

- By default, the value of failCleanAgentWithNoArguments is true, which means that the **fteCleanAgent**
  command fails to run if only the agent name parameter is specified.
- If failCleanAgentWithNoArguments is set to false and only the agent name parameter is specified,
  fteCleanAgent behaves in the same way as it does when you specify the **-all** parameter.

You must run the **fteCleanAgent** command on an agent that has been stopped. If you try to run the
fteCleanAgent command on an agent that is currently running, you receive an error. This command does
not start the agent. The **fteCleanAgent** command cleans up an agent on the system where you issue the
command. You cannot clean up an agent on a remote system. To run the **fteCleanAgent** command you
must have write access to the agent lock file, which is located at *MQ_DATA_PATH*\mqft\logs\
coordination_QMgr_name\agents*agent_name*\agent.lck

The FTEAGENT group must have BROWSE authority on the following queues to run **fteCleanAgent** successfully:

- SYSTEM.FTE.COMMAND.*agent_name*
- SYSTEM.FTE.EVENT.*agent_name*
- SYSTEM.FTE.STATE.*agent_name*

If you are running the **fteCleanAgent** command on an agent that is connected to its queue manager in bindings mode, and the agent has recently stopped running, the **fteCleanAgent** command might report messaging problem: MQRC 2042. This MQRC occurs because a queue handle for the agent still exists in the queue manager. After a short delay the queue manager removes this handle, and you can reissue **fteCleanAgent**.

Specify the optional **-p** parameter for this command only if you want to use a set of configuration options different from your default set. See Configuration options for more information.

Note: When cleaning a Connect:Direct bridge agent, the user ID used to run the **fteCleanAgent** command must have read and write access to the Connect:Direct bridge agent temporary directory.

Syntax

fteCleanAgent
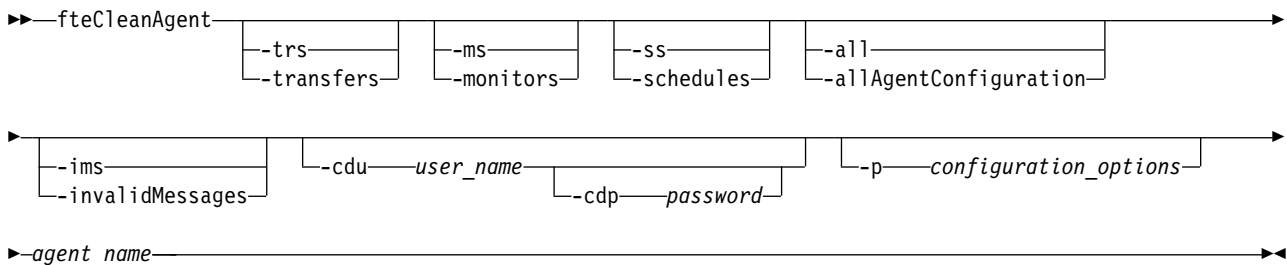
Parameters

You can use the **fteCleanAgent** command to delete specific artifacts. For example, you can specify the **-trs** command to delete pending transfers but not change any resource monitors and scheduled transfers.

-trs or -transfers
   Optional. Specifies that in-progress and pending transfers are to be deleted from the agent. You cannot specify this parameter with **-all** or **-ims** parameters.

-ms or -monitors
   Optional. Specifies that all resource monitor definitions are to be deleted from the agent. You cannot specify this parameter with **-all** or **-ims** parameters.

-ss or -schedules
   Optional. Specifies that all scheduled transfer definitions are to be deleted from the agent. You cannot specify this parameter with the **-all** or **-ims** parameters.

-all or -allAgentConfiguration
   Optional. Specifies that all transfers, resource monitor definitions and scheduled transfer definitions are to be deleted from the agent. You cannot specify this parameter with the **-trs,-ss**, **-ms**, or **-ims** parameters.

-ims or -invalidMessages
   Optional. Specifies that all invalid messages are to be deleted from the agent. You cannot specify this parameter with the **-trs,-ss,-ms**, or **-all** parameters.

-cdu *user_name*

Optional. Only valid if the agent being cleaned is a Connect:Direct bridge agent. If this parameter is specified, the command uses the user name provided to make a connection to the Connect:Direct bridge node and retrieve additional information about existing Connect:Direct processes. If you do not specify this parameter, the agent is cleaned but information about Connect:Direct processes is not displayed.

-cdp *password*

Optional. Valid only if the agent being cleaned is a Connect:Direct bridge agent and you have specified the **-cdu** parameter. If you specify the **-cdp** parameter, the command uses the password provided to make a connection to the Connect:Direct bridge node and retrieve additional information about existing Connect:Direct processes. If you do not specify this parameter, and the **-cdu** parameter has been specified, you are asked to provide the password interactively.

-p (*configuration_options*)

Optional. This parameter determines the set of configuration options that is used to clean up an agent. By convention use the name of a non-default coordination queue manager as the input for this parameter. The command then uses the set of properties files associated with this non-default coordination queue manager.

If you do not specify this parameter, the set of configuration options based on the default coordination queue manager is used.

agent_name

Required. The name of the WebSphere MQ Managed File Transfer agent that you want to clean up.

-? or -h

Optional. Displays command syntax.

Examples

In this basic example, all the queues used by AGENT2 are cleaned up:

```
 C:\Documents and Settings\Administrator>fteCleanAgent -all AGENT2
5724-H72 Copyright IBM Corp.  2008, 2011.  ALL RIGHTS RESERVED

All messages will be deleted from all queues

State Queue Entries:

Transfer Identifier:            414d5120716d31202020202020202020202786de4d20485b03
Source Agent Name:              AGENT2
Destination Agent Name:         AGENT3

Transfer Identifier:            414d5120716d31202020202020202020202786de4d20487203
Source Agent Name:              AGENT2
Destination Agent Name:         AGENT3


Command Queue New Transfer Entries:


Scheduler Queue Schedule Entries:

Directory Monitor Configuration for "MONITOR1" has been cleared from the Agent.


Schedule Identifier:        1
Source Agent Name:          AGENT2
Destination Agent Name:     AGENT3

BFGCL0149I: The agent 'AGENT2' has been cleaned.
```

In this example, the invalid messages queue used by AGENT2 are cleaned up:

```
C:\Documents and Settings\Administrator>fteCleanAgent -ims AGENT2
5724-H72 Copyright IBM Corp.  2008, 2011.  ALL RIGHTS RESERVED

Invalid messages will be deleted from all queues

State Queue Entries:

Warning - Invalid message found on the queue


Command Queue New Transfer Entries:

Warning - Invalid message found on the queue


Scheduler Queue Schedule Entries:

Warning - Invalid message found on the queue

BFGCL0149I: The agent 'AGENT2' has been cleaned.
```

In this example, the transfers queue used by the Connect:Direct bridge agent, AGENT_CD_BRIDGE, is cleaned up:

```
C:\Documents and Settings\Administrator>fteCleanAgent -trs -cdu USER1 AGENT_CD_BRIDGE
5724-H72 Copyright IBM Corp.  2008, 2011.  ALL RIGHTS RESERVED
Enter Connect:Direct password:

All messages will be deleted from the state and command queues

State Queue Entries:

Transfer Identifier:                 414d5120716d31202020202020202020202786de4d2048a703
Source Agent Name:                   AGENT2
Destination Agent Name:              AGENT_CD_BRIDGE
Connect:Direct PNODE Name:           CDNODE1
Connect:Direct SNODE Name:           CDNODE2
Connect:Direct Current Processes:  Name=FA34F8, Number=139



Command Queue New Transfer Entries:

BFGCL0149I: The agent 'AGENT_CD_BRIDGE' has been cleaned.
```

Return codes

0   Command completed successfully.

1   Command ended unsuccessfully.

Related reference:
"fteStopAgent (stop a WebSphere MQ Managed File Transfer agent)" on page 532
Use the **fteStopAgent** command to either stop a WebSphere MQ Managed File Transfer agent in a controlled way or to stop an agent immediately if necessary using the **-i** parameter.

"fteDeleteAgent (delete a WebSphere MQ Managed File Transfer agent)" on page 485
The **fteDeleteAgent** command deletes a WebSphere MQ Managed File Transfer agent and its configuration. If the agent is protocol a bridge agent, the user credentials file is left on the file system.

fteCommon and ftePlatform scripts

The fteCommon and ftePlatform are scripts that are provided by WebSphere MQ Managed File Transfer in the *MQ_INSTALLATION_PATH*/bin directory as helper scripts. Not all of these scripts are present on every platform.

fteCommon

fteCommon is a helper script started by the other WebSphere MQ Managed File Transfer command scripts to perform common setup processing before starting Java.

ftePlatform

ftePlatform is a helper script started by the fteCommon script to perform platform-specific setup processing.

fteCreateAgent (create a WebSphere MQ Managed File Transfer agent)

The **fteCreateAgent** command creates an agent and its associated configuration.

Important: Only users who are WebSphere MQ administrators (and members of the mqm group) can run this command. If you try to run this command as a user who is not a WebSphere MQ administrator, you will receive an error message and the command will not run.

Purpose

Use the **fteCreateAgent** command to create an agent. This command provides you with the MQSC commands that you must run against your agent queue manager to create the following agent queues:
- SYSTEM.FTE.AUTHADM1.*agent_name*
- SYSTEM.FTE.AUTHAGT1.*agent_name*
- SYSTEM.FTE.AUTHMON1.*agent_name*
- SYSTEM.FTE.AUTHOPS1.*agent_name*
- SYSTEM.FTE.AUTHSCH1.*agent_name*
- SYSTEM.FTE.AUTHTRN1.*agent_name*
- SYSTEM.FTE.COMMAND.*agent_name*
- SYSTEM.FTE.DATA.*agent_name*
- SYSTEM.FTE.EVENT.*agent_name*
- SYSTEM.FTE.REPLY.*agent_name*
- SYSTEM.FTE.STATE.*agent_name*

These queues are internal system queues that you must not modify, delete, or read messages from unless you are deleting the agent. The MQSC commands to run are also supplied in a file in the following location: *MQ_DATA_PATH*\mqft\config*coordination_qmgr_name*\agents*agent_name*\ *agent_name*_create.mqsc.

If you later want to delete the agent, this command also provides you with the MQSC commands you must run to clear then delete the queues used by the agent. The MQSC commands are in a file in the following location: *MQ_DATA_PATH*\mqft\config*coordination_qmgr_name*\agents*agent_name*\ *agent_name*_delete.mqsc.

WebSphere MQ Managed File Transfer provides advanced agent properties that help you configure agents. These properties are described in The agent.properties file.

You might need to create a MQMFTCredentials.xml credentials file in order to work with your agent. A sample of this file is located in *MQ_INSTALLATION_PATH*/mqft/samples/credentials/. For more information and examples, see "MQMFT credentials file format" on page 835.

Syntax

fteCreateAgent
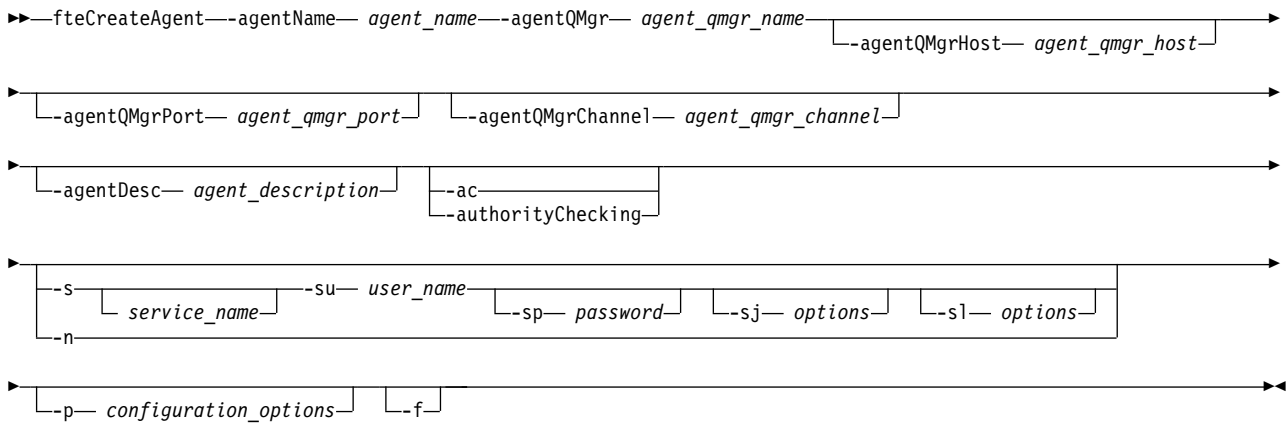
```
►►──fteCreateAgent──-agentName─ agent_name ──-agentQMgr─ agent_qmgr_name ─────────────────►
                                                       └─-agentQMgrHost─ agent_qmgr_host ─┘

►─────────────────────────────────────────────────────────────────────────────────────────►
    └─-agentQMgrPort─ agent_qmgr_port ─┘   └─-agentQMgrChannel─ agent_qmgr_channel ─┘

►─────────────────────────────────────────────────────────────────────────────────────────►
    └─-agentDesc─ agent_description ─┘   ┌─-ac──────────────┐
                                        └─-authorityChecking─┘

►─────────────────────────────────────────────────────────────────────────────────────────►
   ┌─-s──────────────────────┬─-su─ user_name ──────────────────────────────────────────┐
   │    └─ service_name ─┘    │              └─-sp─ password ─┘  └─-sj─ options ─┘  └─-sl─ options ─┘
   └─-n──────────────────────┘

►─────────────────────────────────────────────────────────────────────────────────────────►◄
    └─-p─ configuration_options ─┘   └─-f─┘
```

Parameters

-agentName (*agent_name*)
Required. The name of the agent you want to create. The agent name must be unique to its coordination queue manager.

For more information about naming agents, see Object naming conventions.

-agentQMgr (*agent_qmgr_name*)
Required. The name of the agent queue manager.

-agentQMgrHost (*agent_qmgr_host*)
Optional. The host name or IP address of the agent queue manager.

-agentQMgrPort (*agent_qmgr_port*)
Optional. The port number used for client connections to the agent queue manager.

-agentQMgrChannel (*agent_qmgr_channel*)
Optional. The channel name used to connect to the agent queue manager.

-agentDesc (*agent_description*)
Optional. A description of the agent, which is displayed in WebSphere MQ Explorer.

-ac or -authorityChecking
Optional. This parameter enables authority checking. If you specify this parameter, the agent checks that users who are submitting requests are authorized to perform the requested action. For more information, see "User authorities on WebSphere MQ Managed File Transfer actions" on page 409.

-s (*service_name*)
Optional (Windows only). Indicates that the agent is to run as a Windows service, the command must be run from a Windows administrator user ID. If you do not specify *service_name*, the service is named mqmftAgent<AGENT><QMGR>, where <AGENT> is the agent name and <QMGR> is your agent queue manager name.

The display name for the service, which is shown in the Windows **Services** window in the **Name** column, is always **WebSphere MQ Managed File Transfer agent <AGENT>@<QMGR>**.

-su (*user_name*)
Optional (Windows only). When the agent is to run as a Windows service, this parameter specifies the name of the account under which the service runs. To run the agent using a Windows domain user account specify the value in the form DomainName\UserName. To run the service using an account from the local built-in domain specify the value in the form UserName.

The Windows user account that you specify using the **-su** parameter must have the **Log on as a service** right. For information about how to grant this right, see "Guidance for running an agent or logger as a Windows service" on page 366.

Required when **-s** specified. Equivalent to **-serviceUser**.

-sp (*password*)
Optional (Windows only). Password for the user account set by **-su** or **-serviceUser** parameter.

This parameter is only valid when **-s** is specified. Equivalent to **-servicePassword**. If you do not specify this parameter when you specify the **-s** parameter, a warning message is produced. This message warns you that you must set the password using the Windows Services tool before the service starts successfully.

-sj (*options*)
Optional (Windows only). When the agent is started as a Windows service, defines a list of options in the form of -D or -X that are passed to the JVM. The options are separated using a number sign (#) or semicolon (;) character. If you must embed any # or semicolon (;) characters, put them inside single quotation marks.

This parameter is only valid when **-s** is specified. Equivalent to **-serviceJVMOptions**.

-sl (*options*)
Optional (Windows only). Sets the Windows service log level. Valid options are: error, info, warn, debug. The default is info. This option can be useful if you are having problems with the Windows service. Setting it to debug gives more detailed information in the service log file.

This parameter is only valid when **-s** is specified. Equivalent to **-serviceLogLevel**.

-n Optional (Windows only). Indicates that the agent is to be run as a normal process. This is mutually exclusive with the **-s** option. If neither one of the **-s** parameter and the **-n** parameter is specified, then the agent is configured as a normal Windows process.

Equivalent to **-normal**.

-p (*configuration_options*)
Optional. This parameter determines the set of configuration options that is used to create an agent. By convention use the name of a non-default coordination queue manager as the input for this parameter. The **fteCreateAgent** command then uses the set of properties files associated with this non-default coordination queue manager.

Specify the optional **-p** parameter only if you want to use configuration options different from your defaults. If you do not specify this parameter, the set of configuration options based on the default coordination queue manager is used.

-f Optional. Forces the command to overwrite non-matching existing parameters. Specifying this parameter does not force the replacement of an existing Windows service agent.

-? or -h
Optional. Displays command syntax.

Example

In this example, AGENT3 is created with an agent queue manager QM_NEPTUNE and uses the default coordination queue manager:

```
fteCreateAgent -agentName AGENT3 -agentQMgr QM_NEPTUNE
 -agentQMgrHost myhost.ibm.com -agentQMgrPort 1415 -agentQMgrChannel CHANNEL1
```

Return codes

0   Command completed successfully.

1   Command ended unsuccessfully.

fteCreateBridgeAgent (create and configure WebSphere MQ Managed File Transfer protocol bridge agent)

The **fteCreateBridgeAgent** command creates a protocol bridge agent and its associated configuration. Create a protocol bridge agent for each file server that you want to send files to and receive files from.

Important: Only users who are WebSphere MQ administrators (and members of the mqm group) can run this command. If you try to run this command as a user who is not a WebSphere MQ administrator, you will receive an error message and the command will not run.

Purpose

Use the **fteCreateBridgeAgent** command to create a protocol bridge agent. For an overview of how to use the protocol bridge, see "The protocol bridge" on page 248. This **fteCreateBridgeAgent** command provides you with the MQSC commands that you must run against your agent queue manager to create the following agent queues:

- SYSTEM.FTE.AUTHADM1.*agent_name*
- SYSTEM.FTE.AUTHAGT1.*agent_name*
- SYSTEM.FTE.AUTHMON1.*agent_name*
- SYSTEM.FTE.AUTHOPS1.*agent_name*
- SYSTEM.FTE.AUTHSCH1.*agent_name*
- SYSTEM.FTE.AUTHTRN1.*agent_name*
- SYSTEM.FTE.COMMAND.*agent_name*
- SYSTEM.FTE.DATA.*agent_name*
- SYSTEM.FTE.EVENT.*agent_name*
- SYSTEM.FTE.REPLY.*agent_name*
- SYSTEM.FTE.STATE.*agent_name*

These queues are internal system queues that you must not modify, delete, or read messages from unless you are deleting the agent. The MQSC commands to run are also supplied in a file in the following location: `MQ_DATA_PATH\mqft\config\`*coordination_qmgr_name*`\agents\`*agent_name*`\` *agent_name*`_create.mqsc`

If you later want to delete the agent, this command also provides you with the MQSC commands you must run to clear then delete the queues use by the agent. The MQSC commands are in a file in the following location: `MQ_DATA_PATH\mqft\config\`*coordination_qmgr_name*`\agents\`*agent_name*`\` *agent_name*`_delete.mqsc`.

The **fteCreateBridgeAgent** command creates a `ProtocolBridgeProperties.xml` XML file in the following directory: `MQ_DATA_PATH\mqft\config\`*coordination_qmgr_name*`\agents\`*agent_name*. The user must manually create a `ProtocolBridgeCredentials.xml` file. The `ProtocolBridgeCredentials.xml` file allows you to define user names and credential information that the protocol bridge agent uses to authorize itself with the protocol server and the `ProtocolBridgeProperties.xml` file allows you to define multiple protocol file servers so you can transfer to multiple endpoints. There is a sample `ProtocolBridgeCredentials.xml` in the `MQ_INSTALLATION_PATH`/mqft/samples/credentials/ directory. For more information, see "Protocol bridge credentials file format" on page 567 and "Protocol bridge properties file format" on page 571. If you run the **fteCreateBridgeAgent** command and specify a default protocol file server, this default server is contained in the `ProtocolBridgeProperties.xml` file and its hostname is used for the server name. If you do not specify a default server, there are no entries in the `ProtocolBridgeProperties.xml` file; you must add at least one server manually before transfers can take place.

WebSphere MQ Managed File Transfer provides advanced agent properties that help you configure protocol bridge agents. The properties that relate to the protocol bridge start with `protocol`. These

properties are described in The agent.properties file. If you see unexpected behavior in the protocol bridge, review these protocol properties and ensure that you have set these properties correctly for your system.

If you see the following output from the **fteCreateBridgeAgent** command:

```
BFGMQ1007I: The coordination queue manager cannot be contacted or has refused a connection attempt.
The WebSphere MQ reason code was 2058. The agent's presence will not be published.
```

it indicates that the coordination queue manager can not be contacted and provides the WebSphere MQ reason code for why. This information message can indicate that the coordination queue manager is currently unavailable or that you have defined the configuration incorrectly.

Syntax

fteCreateBridgeAgent

```
▶▶──fteCreateBridgeAgent──-agentName── ──agent_name──-agentQMgr── ──agent_qmgr_name─────────────▶

▶─────┬──────────────────────────────────┬──┬─────────────────────────┬───────────────────────▶
      └─-bt── ──protocol_file_server_type─┘  └─-bh── ──server_host_name─┘

▶─────┬───────────────────────────────┬──┬─────────────────────────────┬─────────────────────────▶
      └─-btz── ──server_time_zone─────┘  └─-bm── ──server_platform_type─┘

▶─────┬───────────────────────────────────────────────────────┬──-bts── ──truststore_file────────▶
      └─-bsl── ──server_locale─┘ └─-bfe── ──server_file_encoding─┘

▶─────┬─────────────────────────────┬──┬──────┬──┬──────────────────────────┬─────────────────────▶
      └─-bp── ──server_port_number─┘  └─-blw─┘  └─-blf── ──server listing format─┘

▶─────┬───────────────────────────────────────┬──┬──────────────────────────────────────┬─────────▶
      └─-agentQMgrHost── ──agent_qmgr_host_name─┘  └─-agentQMgrPort── ──agent_qmgr_port────┘

▶─────┬───────────────────────────────────────────┬──┬───────────────────────────────────┬────────▶
      └─-agentQMgrChannel── ──agent_qmgr_channel──┘  └─-agentDesc── ──agent_description────┘

▶─────┬──────────────────────┬───────────────────────────────────────────────────────────────────▶
      ├─-ac────────────────┤
      └─-authorityChecking─┘

▶─────┬──-s──┬────────────────┬──-su──user_name──┬──────────────────┬──┬─────────────┬──┬─────────────┬─▶
      │      └─service_name──┘                    └─-sp──password────┘  └─-sj──options─┘  └─-sl──options─┘
      └─-n──────────────────────────────────────────────────────────────────────────────────────────┘

▶─────┬───────────────────────────────────┬──┬────┬──────────────────────────────────────────────◀◀
      └─-p── ──configuration-options──────┘  └─-f─┘
```

Parameters

-agentName (*agent_name*)
Required. The name of the agent you want to create. The agent name must be unique in its administrative domain.

For more information about naming agents, see Object naming conventions.

-agentQMgr (*agent_qmgr_name*)
Required. The name of the agent queue manager.

-bt (*protocol_file_server_type*)
Optional. Specifies that you want to define a default protocol file server. Specify one of the following options:

FTP   Standard FTP server

SFTP   SSH FTP server

FTPS   FTP server secured using SSL or TLS

If you do not specify this parameter, no default protocol server is defined.

-bh (*server_host_name***)**
Required only if you also specify a default protocol file server using the **-bt** parameter. The IP host name or IP address of the protocol file server.

-btz (*server_time_zone***)**
Required only if you also specify the **-bt** parameter (FTP and FTPS servers only). The time zone of the protocol file server. Specify the time zone in the following format: Area/Location. For example: Europe/London.

You can use the **-htz** parameter to list the possible values for **-btz**. For example:
`fteCreateBridgeAgent -htz`

-bm (*server_platform***)**
Required only if you also specify a default protocol file server using the **-bt** parameter. The platform type of the protocol file server. Specify one of the following options:

UNIX   Generic UNIX platform

WINDOWS
        Generic Windows platform

-bsl (*server_locale***)**
Required only if you also specify the **-bt** parameter (FTP and FTPS servers only). The locale of the protocol file server. Specify the locale in the following format: *xx_XX*. For example: en_GB.
- *xx* is the ISO Language Code. For a list of valid values, see Codes for the Representation of Names of Languages
- *XX* is the ISO Country Code. For a list of valid values, see Country names and code elements

-bfe (*server_file_encoding***)**
Required only if you also specify a default protocol file server using the **-bt** parameter. The character encoding format of the files stored on the protocol file server. For example: UTF-8.

You can use the **-hcs** parameter to list the possible values for **-bfe**. For example:
`fteCreateBridgeAgent -hcs`

-bts (*truststore_file***)**
Required when you specify the **-bt** parameter (FTPS servers only). Specifies the path to a truststore that is used to validate the certificate presented by the FTPS server.

You can specify the **-bts** parameter only if you have also specified the FTPS option on the **-bt** parameter.

-bp (*server_port***)**
Optional. The IP port that the protocol file server is connected to. Specify this parameter only if your protocol file server does not use the default port for that protocol. If you do not specify this parameter, WebSphere MQ Managed File Transfer uses the default port for the protocol type of file server.

-blw
Optional. Defines the protocol file server as having limited write abilities. By default, a protocol bridge agent expects the protocol file server to permit file deletion, file renaming, and file opening for append writing. Specify this parameter to indicate that the protocol file server does not permit these file actions. Instead the file server permits read from and write to file only. If you specify this parameter, any transfers might not be recoverable if they are interrupted and might result in a failure for the file currently being transferred.

-blf (*server listing format***)**

Optional and for FTP and FTPS servers only. Defines the server listing format of the listed file information returned from the default protocol file server. The options are as follows:

UNIX   Generic UNIX platform

WINDOWS

Generic Windows platform

To identify which format to select, use a FTP client program and perform a listing of a directory and select which format is the best fit. For example,

UNIX displays the following type of listing:

```
-rwxr-xr-x 2 userid groupId 4096 2009-07-23 09:36 filename
```

Windows displays the following type of listing:

```
437,909 filename
```

The default is UNIX, which is the format used by most servers.

-agentQMgrHost (*agent_qmgr_host***)**

Optional. The host name or IP address of the agent queue manager.

-agentQMgrPort (*agent_qmgr_port***)**

Optional. The port number used for client connections to the agent queue manager.

-agentQMgrChannel (*agent_qmgr_channel***)**

Optional. The channel name used to connect to the agent queue manager.

-agentDesc (*agent_description***)**

Optional. A description of the agent, which is displayed in the WebSphere MQ Explorer.

-ac or -authorityChecking

Optional. This parameter enables authority checking. If you specify this parameter, the agent checks that users who are submitting requests are authorized to perform the requested action. For more information, see "User authorities on WebSphere MQ Managed File Transfer actions" on page 409.

-s (*service_name***)**

Optional (Windows only). Indicates that the agent is to run as a Windows service. If you do not specify *service_name*, the service is named mqmftAgent*<AGENT><QMGR>*, where *<AGENT>* is the agent name and *<QMGR>* is your agent queue manager name.

The display name for the service, which is shown in the Windows **Services** window in the **Name** column, is always **WebSphere MQ Managed File Transfer agent** *<AGENT>@<QMGR>*.

-su (*user_name***)**

Optional (Windows only). When the agent is to run as a Windows service, this parameter specifies the name of the account under which the service runs. To run the agent using a Windows domain user account specify the value in the form DomainName\UserName. To run the service using an account from the local built-in domain specify the value in the form UserName.

The Windows user account that you specify using the **-su** parameter must have the **Log on as a service** right. For information about how to grant this right, see "Guidance for running an agent or logger as a Windows service" on page 366.

Required when **-s** specified. Equivalent to **-serviceUser**.

-sp (*password***)**

Optional (Windows only). Password for the user account set by **-su** or **-serviceUser** parameter.

This parameter is only valid when **-s** is specified. Equivalent to **-servicePassword**. If you do not specify this parameter when you specify the **-s** parameter, a warning message is produced. This message warns you that you must set the password using the Windows Services tool before the service starts successfully.

-sj (_options_**)**

Optional (Windows only). When the agent is started as a Windows service, defines a list of options in the form of -D or -X that are passed to the JVM. The options are separated using a number sign (#) or semicolon (;) character. If you must embed any # or semicolon (;) characters, put them inside single quotation marks.

This parameter is only valid when **-s** is specified. Equivalent to **-serviceJVMOptions**.

-sl (_options_**)**

Optional (Windows only). Sets the Windows service log level. Valid options are: error, info, warn, debug. The default is info. This option can be useful if you are having problems with the Windows service. Setting it to debug gives more detailed information in the service log file.

This parameter is only valid when **-s** is specified. Equivalent to **-serviceLogLevel**.

-n Optional (Windows only). Indicates that the agent is to be run as a normal process. This is mutually exclusive with the **-s** option. If neither one of the **-s** parameter and the **-n** parameter is specified, then the agent is configured as a normal Windows process.

Equivalent to **-normal**.

-p (_configuration-options_**)**

Optional. This parameter determines the set of configuration options that is used to create an agent. By convention use the name of a non-default coordination queue manager as the input for this parameter. The **fteCreateBridgeAgent** command then uses the set of properties files associated with this non-default coordination queue manager.

Specify the optional **-p** parameter only if you want to use configuration options different from your defaults. If you do not specify **-p**, the configuration options defined in the installation.properties file are used. See "Configuration options" on page 124 for more information.

-f Optional. Forces the command to overwrite the existing configuration.

-htz

Optional. Displays a list of supported time zones that you can use as input for the **-btz** parameter.

-hcs

Optional. Displays a list of supported character sets that you can use as input for the **-bfe** parameter.

Run the **fteCreateBridgeAgent -hcs** command to list the known code pages for the JVM. This information is not available from an external source because the known code pages vary between JVMs.

-? or -h

Optional. Displays command syntax.

Deprecated parameters

The following parameters have been deprecated and are not supported on IBM WebSphere MQ V7.5 or on WebSphere MQ Managed File Transfer V7.0.2, or later.

-brd (_reconnect_delay_**)**

Deprecated. Optional. Specifies in seconds the delay period between attempts to re-establish a lost connection with the protocol file server. The default value is 10 seconds.

-brr (_reconnect_retries_**)**

Deprecated. Optional. Specifies the maximum number of times to try again when attempting to re-establish a lost connection with the default protocol file server. When this maximum number is reached, the current file transfer is classed as failed. The default value is 2.

Example

In this example, a new protocol bridge agent ACCOUNTS1 is created with an agent queue manager QM_ACCOUNTS and uses the default coordination queue manager. ACCOUNTS1 connects to the FTP server accountshost.ibm.com. This FTP server runs on Windows using a time zone of Europe/Berlin, a locale of de_DE, and a file encoding of UTF-8. The number of reconnect retries is 4:

```
fteCreateBridgeAgent -agentName ACCOUNTS1 -agentQMgr QM_ACCOUNTS -bt FTP
 -bh accountshost.ibm.com -bm WINDOWS -btz Europe/Berlin -bsl de_DE -bfe UTF8
 -agentQMgrHost myhost.ibm.com -agentQMgrPort 1415 -agentQMgrChannel CHANNEL1
```

In this example, a new protocol bridge agent ACCOUNTS2 is created with an agent queue manager QM_ACCOUNTS and uses the default coordination manager. ACCOUNTS2 is created without a default protocol file server.

```
fteCreateBridgeAgent -agentName ACCOUNTS2 -agentQMgr QM_ACCOUNTS
```

Return codes

0   Command completed successfully.

1   Command ended unsuccessfully.

Related concepts:

"The protocol bridge" on page 248
The protocol bridge enables your WebSphere MQ Managed File Transfer (MQMFT) network to access files stored on a file server outside your MQMFT network, either in your local domain or a remote location. This file server can use the FTP, FTPS, or SFTP network protocols. Each file server needs at least one dedicated agent. The dedicated agent is known as the protocol bridge agent. A bridge agent can interact with multiple file servers.

fteCreateCDAgent (create a Connect:Direct bridge agent)

The fteCreateCDAgent command creates a WebSphere MQ Managed File Transfer agent and its associated configuration for use with the Connect:Direct bridge.

Important: Only users who are WebSphere MQ administrators (and members of the mqm group) can run this command. If you try to run this command as a user who is not a WebSphere MQ administrator, you will receive an error message and the command will not run.

Purpose

Use the **fteCreateCDAgent** command to create a Connect:Direct bridge agent. This type of agent is dedicated to transferring files to and from Connect:Direct nodes. For more information, see "The Connect:Direct bridge" on page 263. For details of the supported operating system versions for the Connect:Direct bridge, see the web page WebSphere MQ System Requirements.

This command provides you with the MQSC commands that you must run against your agent queue manager to create the following agent queues:

- SYSTEM.FTE.AUTHADM1.*agent_name*
- SYSTEM.FTE.AUTHAGT1.*agent_name*
- SYSTEM.FTE.AUTHMON1.*agent_name*
- SYSTEM.FTE.AUTHOPS1.*agent_name*
- SYSTEM.FTE.AUTHSCH1.*agent_name*
- SYSTEM.FTE.AUTHTRN1.*agent_name*
- SYSTEM.FTE.COMMAND.*agent_name*
- SYSTEM.FTE.DATA.*agent_name*
- SYSTEM.FTE.EVENT.*agent_name*

- SYSTEM.FTE.REPLY.*agent_name*
- SYSTEM.FTE.STATE.*agent_name*

These queues are internal system queues that you must not modify, delete, or read messages from unless you are deleting the agent. The MQSC commands to run are also supplied in a file in the following location: *MQ_DATA_PATH*\mqft\config*coordination_qmgr_name*\agents*agent_name*\ *agent_name*_create.mqsc.

If you later want to delete the agent, this command also provides you with the MQSC commands you must run to clear then delete the queues belonging to the agent. The MQSC commands are in a file in the following location: *MQ_DATA_PATH*\mqft\config*coordination_qmgr_name*\agents*agent_name*\ *agent_name*_delete.mqsc.

WebSphere MQ Managed File Transfer provides advanced agent properties that help you configure agents. These properties are described in Properties files for WebSphere MQ File Transfer Edition.

The **fteCreateCDAgent** command creates two XML files in the agent properties directory. ConnectDirectNodeProperties.xml, which is used to define information about the remote nodes in a transfer, and ConnectDirectProcessDefinitions.xml, which is used to specify which user-defined Connect:Direct processes are started by transfers.

To define user names and passwords that the Connect:Direct bridge agent uses to connect to Connect:Direct nodes, you must manually create a ConnectDirectCredentials.xml file. Sample XML files are located in *MQ_INSTALLATION_PATH*/mqft/samples/credentials/. For more information and examples, see "Connect:Direct credentials file format" on page 579.

fteCreateCDAgent
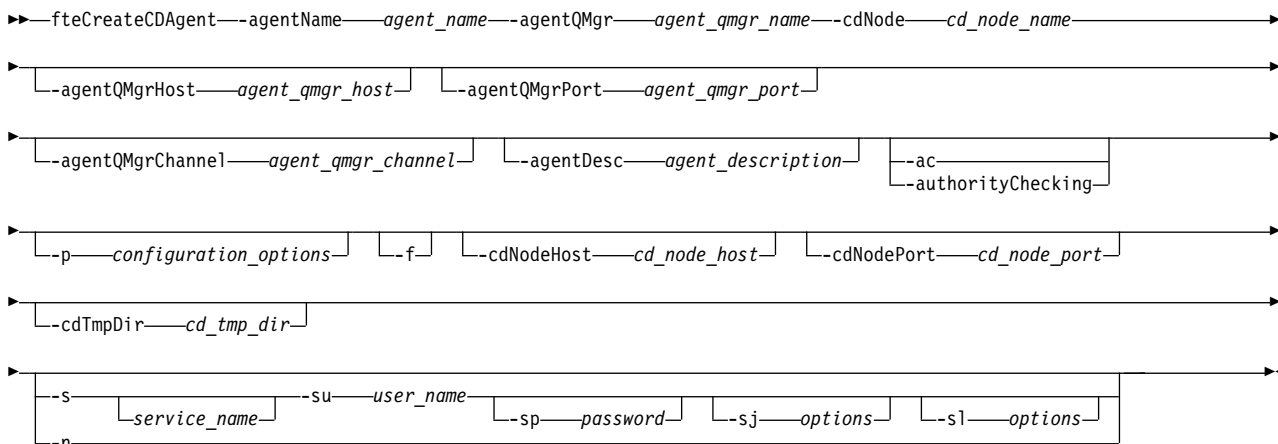
```
►►──fteCreateCDAgent──-agentName────agent_name──-agentQMgr────agent_qmgr_name──-cdNode────cd_node_name────────►

►──┬─────────────────────────────────────────┬──┬──────────────────────────────────┬──────────────────────►
   └─-agentQMgrHost────agent_qmgr_host─┘          └─-agentQMgrPort────agent_qmgr_port─┘

►──┬──────────────────────────────────────────────┬──┬──────────────────────────────────┬──┬──────────────┬──►
   └─-agentQMgrChannel────agent_qmgr_channel─┘        └─-agentDesc────agent_description─┘   ├─-ac─────────┤
                                                                                            └─-authorityChecking─┘

►──┬──────────────────────────────────┬──┬────┬──┬─────────────────────────────┬──┬───────────────────────────┬──►
   └─-p────configuration_options─┘        └─-f─┘  └─-cdNodeHost────cd_node_host─┘   └─-cdNodePort────cd_node_port─┘

►──┬──────────────────────────────┬────────────────────────────────────────────────────────────────────────►
   └─-cdTmpDir────cd_tmp_dir─┘

►──┬─-s─┬──────────────────────┬──-su────user_name──┬──────────────────────┬──┬──────────────┬──┬─────────────┬──►◄
   │    └─service_name─┘                              └─-sp────password─┘      └─-sj────options─┘  └─-sl────options─┘
   └─-n─┘
```

Parameters

-agentName (*agent_name***)**
Required. The name of the agent you want to create. The agent name must be unique to its coordination queue manager.

For more information about naming agents, see Object naming conventions.

-agentQMgr (*agent_qmgr_name***)**
Required. The name of the agent queue manager.

-cdNode *cd_node_name*
Required. The name of the Connect:Direct node to use to transfer messages from this agent to destination Connect:Direct nodes. The value of this parameter is used for logging and not to specify

to the Connect:Direct bridge agent which node to connect to. The values of the **-cdNodeHost** and **-cdNodePort** specify the Connect:Direct node that is part of the Connect:Direct bridge.

-agentQMgrHost (_agent_qmgr_host_**)**
　　Optional. The host name or IP address of the agent queue manager.

-agentQMgrPort (_agent_qmgr_port_**)**
　　Optional. The port number used for client connections to the agent queue manager.

-agentQMgrChannel (_agent_qmgr_channel_**)**
　　Optional. The channel name used to connect to the agent queue manager.

-agentDesc (_agent_description_**)**
　　Optional. A description of the agent, which is displayed in WebSphere MQ Explorer.

-ac or -authorityChecking
　　Optional. This parameter enables authority checking. If you specify this parameter, the agent checks that users who are submitting requests are authorized to perform the requested action. For more information, see "User authorities on WebSphere MQ Managed File Transfer actions" on page 409.

-p (_configuration_options_**)**
　　Optional. This parameter determines the set of configuration options that is used to create an agent. By convention use the name of a non-default coordination queue manager as the input for this parameter. The **fteCreateCDAgent** command then uses the set of properties files associated with this non-default coordination queue manager.

　　Specify the optional **-p** parameter only if you want to use configuration options different from your defaults. If you do not specify this parameter, the set of configuration options based on the default coordination queue manager is used.

-f　Optional. Forces the command to overwrite non-matching existing parameters. Specifying this parameter does not force the replacement of an existing Windows service agent.

-cdNodeHost _cd_node_host_name_
　　Optional. The host name or IP address of the system where the Connect:Direct node, specified by the **-cdNode** parameter, is located. If you do not specify the **-cdNodeHost** parameter, a default of the host name or IP address of the local system is used.

　　In most cases, the Connect:Direct node is on the same system as the Connect:Direct bridge agent. In these cases, the default value of this property, which is the IP address of the local system, is correct. If your system has multiple IP addresses, or your Connect:Direct node is on a different system to your Connect:Direct bridge agent and their systems share a file system, use this property to specify the correct host name for the Connect:Direct node.

-cdNodePort _cd_node_port_name_
　　Optional. The port number of the Connect:Direct node that client applications use to communicate with the node that is specified by the **-cdNode** parameter. In Connect:Direct product documentation, this port is referred to as the API port. If you do not specify the **-cdNodePort** parameter, a default port number of 1363 is assumed.

-cdTmpDir _cd_tmp_directory_
　　Optional. The directory to be used by this agent to store files temporarily before they are transferred to the destination Connect:Direct node. This parameter specifies the full path of the directory where files are temporarily stored. For example, if **cdTmpDir** is set to /tmp then the files are temporarily placed in the /tmp directory. If you do not specify the **-cdTmpDir** parameter, the files are stored temporarily in a directory named cdbridge-_agent_name_. This default directory is created in the location that is defined by the value of the java.io.tmpdir property.

　　The Connect:Direct bridge agent and the Connect:Direct bridge node must be able to access the directory specified by this parameter using the same path name. Consider this when planning the installation of your Connect:Direct bridge. If possible, create the agent on the system where the Connect:Direct node that is part of the Connect:Direct bridge is located. If your agent and node are

on separate systems, the directory must be on a shared file system and be accessible from both systems using the same path name. For more information about the supported configurations, see "The Connect:Direct bridge" on page 263.

> **Note:** If you run the **fteCleanAgent** command, all files in this directory are deleted.

-s (*service_name***)**
> Optional (Windows only). Indicates that the agent is to run as a Windows service, the command must be run from a Windows administrator user ID. If you do not specify *service_name*, the service is named mqmftAgent*<AGENT><QMGR>*, where *<AGENT>* is the agent name and *<QMGR>* is your agent queue manager name.
>
> The display name for the service, which is shown in the Windows **Services** window in the **Name** column, is always **WebSphere MQ Managed File Transfer agent *<AGENT>@<QMGR>***.

-su (*user_name***)**
> Optional (Windows only). When the agent is to run as a Windows service, this parameter specifies the name of the account under which the service runs. To run the agent using a Windows domain user account specify the value in the form DomainName\UserName. To run the service using an account from the local built-in domain specify the value in the form UserName.
>
> The Windows user account that you specify using the **-su** parameter must have the **Log on as a service** right. For information about how to grant this right, see "Guidance for running an agent or logger as a Windows service" on page 366.
>
> Required when **-s** specified. Equivalent to **-serviceUser**.

-sp (*password***)**
> Optional (Windows only). Password for the user account set by **-su** or **-serviceUser** parameter.
>
> This parameter is only valid when **-s** is specified. Equivalent to **-servicePassword**. If you do not specify this parameter when you specify the **-s** parameter, a warning message is produced. This message warns you that you must set the password using the Windows Services tool before the service starts successfully.

-sj (*options***)**
> Optional (Windows only). When the agent is started as a Windows service, defines a list of options in the form of -D or -X that are passed to the JVM. The options are separated using a number sign (#) or semicolon (;) character. If you must embed any # or semicolon (;) characters, put them inside single quotation marks.
>
> This parameter is only valid when **-s** is specified. Equivalent to **-serviceJVMOptions**.

-sl (*options***)**
> Optional (Windows only). Sets the Windows service log level. Valid options are: error, info, warn, debug. The default is info. This option can be useful if you are having problems with the Windows service. Setting it to debug gives more detailed information in the service log file.
>
> This parameter is only valid when **-s** is specified. Equivalent to **-serviceLogLevel**.

-n Optional (Windows only). Indicates that the agent is to be run as a normal process. This is mutually exclusive with the **-s** option. If neither one of the **-s** parameter and the **-n** parameter is specified, then the agent is configured as a normal Windows process.

> Equivalent to **-normal**.

Example

In this example, a new Connect:Direct bridge agent CD_BRIDGE is created with an agent queue manager QM_NEPTUNE. The agent uses the Connect:Direct node BRIDGE_NODE to transfer files to other

Connect:Direct nodes. The BRIDGE_NODE node is located on the same system as the agent and uses the default port for client connections. Files that are transferred to or from Connect:Direct are temporarily stored in the directory /tmp/cd-bridge.

```
fteCreateCDAgent -agentName CD_BRIDGE -agentQMgr QM_NEPTUNE
                 -cdNode BRIDGE_NODE -cdTmpDir /tmp/cd-bridge
```

Return codes

0   Command completed successfully.

1   Command ended unsuccessfully.

fteCreateLogger (create a WebSphere MQ Managed File Transfer logger)

Use the **fteCreateLogger** command to create a file or database logger.

Important: Only users who are WebSphere MQ administrators (and members of the mqm group) can run this command. If you try to run this command as a user who is not a WebSphere MQ administrator, you will receive an error message and the command will not run.

Purpose

The **fteCreateLogger** command is supported on WebSphere MQ Managed File Transfer Version 7.5 and later.

The **fteCreateLogger** command provides you with the MQSC commands that you must run against your logger command queue manager to create the following logger queues:
- SYSTEM.FTE.LOG.CMD.*logger_name*
- SYSTEM.FTE.LOG.RJCT.*logger_name*

These queues are internal system queues that you must not modify, delete, or read messages from unless you are deleting the logger. The MQSC commands to run are also supplied in a file in the following location: *MQ_DATA_PATH*\mqft\config*coordination_qmgr*\loggers*logger_name**logger_name*_create.mqsc

If you later want to delete the logger, use the **fteDeleteLogger** command.

WebSphere MQ Managed File Transfer provides advanced logger properties that help you configure loggers. See, "Logger configuration properties for WebSphere MQ Managed File Transfer" on page 144

If the logger you are creating is a database logger and it is not connecting to a local Db2 database, you will need to manually create a MQMFTCredentials.xml file. The new wmqfte.database.credentials.file property is used to specify the location of the credentials file if one is required. A sample of this credentials file is located in *MQ_INSTALLATION_PATH*/mqft/samples/credentials/.

Syntax

fteCreateLogger

```
►►──fteCreateLogger── -loggerType──┬──DATABASE──┤ DATABASE options ├──┬──────────────────►
                                   └──FILE──┤ FILE options ├──────────┘


►──┬─────────────────────────────┬──┬─────────────────────────┬──────────────────────────►
   └─ -loggerQMgr──── qmgr_name──┘  └─┤ Windows systems options ├─┘
```
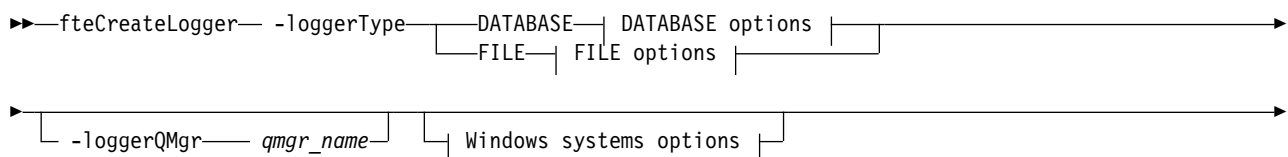
```
►►─┤ Parameters for MQ security: ├─┬─────────────────────────────┬─┬─────┬─ logger_name ──────────────►◄
                                   └─ -p─configuration_ options ─┘ └─ -f ─┘
```

DATABASE options:

```
├──── -dbType──── database_type── -dbName──── database_ name── -dbDriver──── driver ───────────────────►

►─┬────────────────────────┬─┤
  └─ -dbLib──── path ─┘
```

FILE options:

```
├──── -fileLoggerMode──────┬─CIRCULAR─┬── -fileSize──── size── -fileCount──── number ──────────────────┤
                           └─LINEAR───┘
```

Windows system options:

```
├──── -s ──── service_name── -su──── user_name ─┬─────────────────────┬─┬──────────────┬─┬─────────────┬─┤
                                                └─ -sp──── password ──┘ └─ -sj──── options ─┘ └─ -sl──── options ─┘
```

Parameters

-loggerType (*type*)

Required. Specifies where managed file transfer information will be logged. The options for type are either DATABASE, if transfer information will be logged to a database, or FILE, if the information will be logged to a file.

-loggerQMgr (*qmgr_name*)

Optional. Determines the queue manager to connect to in order to receive messages containing information about managed file transfers. The queue manager must be on the same system as the logger. If you do not specify the **–loggerQMgr** parameter then the coordination queue manager that is associated with the configuration options set for this logger is used as the default.

-dbType (*database_type*)

Required when -loggerType is DATABASE. Specifies the type of database management system in use for storing managed file transfer information. The options are db2 or oracle.

-dbName (*database_name*)

Required when -loggerType is DATABASE. The name of the database where managed file transfer information is stored. The database must be configured with the WebSphere MQ Managed File Transfer log tables.

-dbDriver (*driver*)

Required when -loggerType is DATABASE. The location of the JDBC driver classes for the database. This is typically the path and file name of a JAR file.

-dbLib (*path*)

Optional when -loggerType is DATABASE. The location of any native libraries needed by your chosen database driver.

-fileLoggerMode (*mode*)

Required when -loggerType is FILE. Specifies the type of file system in use for storing managed file transfer information. The options are LINEAR or CIRCULAR.

Option LINEAR means the file logger will write information to a file until that file reaches its maximum size as defined by -filesize. When the maximum size is reached the file logger will start a new file. Previously written files will not be deleted which allows them to be kept as a historical

record of log messages. Files are not deleted when running in this mode, so the -fileCount will be ignored as there is no upper limit to the number of files that can be created. As there is no upper limit when running in this mode it will be necessary to track the amount of disk space used by the log files in order to avoid running low on disk space.

Option CIRCULAR means the file logger will write information to a file until that file reaches its maximum size as defined by -fileSize. When the maximum size is reached the file logger will start a new file. The maximum number of files written in this mode is controlled by the value defined using the -fileCount. When this maximum number of files is reached the file logger will delete the first file and re-create it for use as the currently active file. If the value defined in the -fileSize is a fixed size byte unit, the upper limit on the disk space used in this mode will equal fileSize x fileCount. If the values defined in -fileSize are a time unit, the maximum size will depend on the throughput of log message in your system over these time periods.

For more information, see "Logger configuration properties for WebSphere MQ Managed File Transfer" on page 144

-fileSize (*size***)**
Required when -loggerType is FILE. The maximum size that a log file is allowed to grow to. The value is a positive integer, greater than zero, followed by one of the following units: KB, MB, GB, m (minutes), h (hours), d (days), w (weeks). For example: -fileSize 5MB (specifies a maximum size of 5MB) , -fileSize 2d (specifies a maximum of 2 days worth of data).

-fileCount (*number***)**
Required when -loggerType is FILE and -fileLoggerMode is CIRCULAR. The maximum number of log files to create. When the amount of data exceeds the maximum amount that can be stored in this number of files, the oldest file is deleted so that the number of log files never exceeds the value specified in this parameter.

-s (*service_name***)**
Optional (Windows systems only). Indicates that the logger is to run as a Windows service. If you do not specify *service_name*, the service is named mqmftLogger<*LOGGER*><*QMGR*>, where <*LOGGER*> is the logger name and <*QMGR*> is your logger queue manager name.

The display name for the service, which is shown in the Windows **Services** window in the **Name** column, is always **WebSphere MQ Managed File Transfer logger <*LOGGER*>@<*QMGR*>**.

-su (*user_name***)**
Optional (Windows only). When the logger is to run as a Windows service, this parameter specifies the name of the account under which the service runs. To run the logger using a Windows domain user account specify the value in the form DomainName\UserName. To run the service using an account from the local built-in domain specify the value in the form UserName.

The Windows user account that you specify using the **-su** parameter must have the **Log on as a service** right. For information about how to grant this right, see "Guidance for running an agent or logger as a Windows service" on page 366.

Required when **-s** specified. Equivalent to **-serviceUser**.

-sp (*password***)**
Optional (Windows only). Password for the user account set by **-su** or **-serviceUser** parameter.

This parameter is only valid when **-s** is specified. Equivalent to **-servicePassword**. If you do not specify this parameter when you specify the **-s** parameter, a warning message is produced. This message warns you that you must set the password using the Windows Services tool before the service starts successfully.

-sj (*options***)**
Optional (Windows only). When the logger is started as a Windows service, defines a list of options in the form of -D or -X that are passed to the JVM. The options are separated using a number sign (#) or semicolon (;) character. If you must embed any (#) or semicolon (;) characters, put them inside single quotation marks.

This parameter is only valid when **-s** is specified. Equivalent to **-serviceJVMOptions**.

-sl (_options_**)**
Optional (Windows only). Sets the Windows service log level. Valid options are: error, info, warn, debug. The default is info. This option can be useful if you are having problems with the Windows service. Setting it to debug gives more detailed information in the service log file.

This parameter is only valid when **-s** is specified. Equivalent to **-serviceLogLevel**.

-p (_configuration options_**)**
Optional. Specifies the set of configuration options that is used to create the logger. By convention, this value is the name of a coordination queue manager. If you do not specify this parameter, the default set of configuration options is used.

-f Optional. Forces the command to overwrite the existing configuration.

(_logger_name_**)**
Required. Name of the logger to create. This is incorporated into WebSphere MQ Managed File Transfer queue names, and so must contain only letters, numbers, and the periods (.) and underscore characters (_). It is also limited to a maximum length of 28 characters.

-? or -h
Optional. Displays command syntax.

Example

In this example, a circular file logger is created called filelogger1. The file logger will create a maximum of 10 files, each file being 10MB in size, using a maximum of 100MB of disk space in total:

```
fteCreateLogger -loggerType FILE -fileLoggerMode CIRCULAR -fileSize 10MB -fileCount 10 filelogger1
```

In this example, a database logger is created called dblogger1. The database logger connects to a Db2 database called FTEDB:

```
fteCreateLogger -loggerType DATABASE -dbName FTEDB -dbType DB2 -dbDriver "C:\Program Files (x86)\IBM\SQLLIB\java\db2
```

Return codes

0   Command completed successfully.

1   Command ended unsuccessfully.

fteCreateMonitor (create new resource monitor)

The **fteCreateMonitor** command creates and starts a new resource monitor from the command line. You can monitor a resource (for example, the contents of a directory) using WebSphere MQ Managed File Transfer so that when a trigger condition is satisfied, a specified task, such as a file transfer, is started.

fteCreateMonitor
Purpose

Use the **fteCreateMonitor** command to create and then start a new resource monitor using a WebSphere MQ Managed File Transfer agent. For example, you can use a resource monitor in the following way: An external application puts one or more files in a known directory and when processing is complete, the external application places a trigger file in a monitored directory. The trigger file is then detected and a defined file transfer starts, which copies the files from the known directory to a destination agent.

You can use the **-ox** and **-ix** parameters to export and import a resource monitor configuration to an XML file. Importing this file with the **fteCreateMonitor** command creates a new resource monitor with the same parameters as the resource monitor given in the **fteCreateMonitor** command to export to the

XML file. You can also use the **fteListMonitor** command to export a resource monitor configuration to an XML file. Additionally, you can use the **-f** and **-c** parameters to overwrite a monitor configuration dynamically.

The **fteCreateMonitor** command is not supported on protocol bridge agents.

Syntax

fteCreateMonitor

```
►►──fteCreateMonitor──────────────────────────────────────────────────────────────►
                      ├─ -ix ── monitor_definition_file ─┤   └─ -mm ── monitoring_agent_qmgr_name ─┘
                      ├─ -ox ── monitor_definition_file ─┤
                      ├─ -ma ── monitoring_agent_name ───┤
                      └─ -mn ── monitor_name ────────────┘

►──┬───────┬──┬─ -md ── directory_path ─┬──┬─ -mt ── task_definition_file_name ─┬──►
   ├─ -f ─┤  └─ -mq ── queue_name ───────┘
   └─ -c ─┘

►──┬─ -rl ── number_of_recursion_levels ─┬──┬─ -pi   interval_period ─┬────────────►
                                            ├─ -pu   units ───────────┤

►──┬─ -tr── ── condition ──────────────┬──┬─ -x ── exclude_pattern ─┬──┬─ -jn ── job name ─┬──►
                └─ ,── pattern ─┘

►──┬─ -pt ── pattern_type ─┬──┬─ -bs ── matches_per_task ─┬──┬─ -dv ── default_variables ─┬──►

►──┬─ -p ── configuration_options ─┬──────────────────────────────────────────────►◄
```

Parameters

-ix (*xml_filename*)
   Optional. Imports the resource monitor configuration from an XML file.

-ox (*xml_filename*)
   Optional. This parameter must be specified with the -ma and -mn parameters. Exports the resource monitor configuration to an XML file.

-ma (*monitoring_agent_name*)
   Optional. The name of the agent to perform the resource monitoring. This monitoring agent must be the source agent for the monitor task that you want to trigger.

-mn (*monitor_name*)
   Optional. The name that you assign to this monitor. The monitor name must be unique to the monitoring agent. However, you can delete a monitor and then create a monitor with the same name.

   Resource monitor names are not case-sensitive. Resource monitor names entered in lowercase or mixed case are converted to uppercase. Resource monitor names must not contain asterisk (*), percent (%) or question mark (?) characters.

-mm (*monitoring_agent_qmgr_name*)
   Optional. The name of the queue manager that the monitoring agent is connected to. Because the monitoring agent and the source agent must be same, this queue manager is also your source agent queue manager.

-f Optional. Use this parameter to overwrite a resource monitor configuration. For example, when the resource monitor name you have chosen already exists on the resource monitoring agent and you want to update it rather than delete and re-create a monitor with the same name. Using this parameter causes the agent to restart the monitor process.

-c Optional. This parameter clears the history of an updated resource monitor, which causes the resource monitor to check the trigger conditions again. You can use this parameter with the **-f** parameter only.

-md (*directory_path***)**
Optional. The absolute name of the directory path that you want to monitor. Unless you are using the **-ix** or **-ox** parameters you must specify one of the **-md** or **-mq** parameters.

-mq (*queue_name***)**
Optional. The name of the queue that you want to monitor. This queue must be located on the monitoring agent queue manager. Unless you are using the **-ix** or **-ox** parameters you must specify one of the **-md** or **-mq** parameters.

-mt (*task_definition_file_name***)**
Optional. The name of the XML document that contains the task definition that you want to carry out when the trigger condition is satisfied. The path to the transfer definition XML document must be on the local file system that you run the **fteCreateMonitor** command from. Unless you are using the **-ix** or **-ox** parameters this will be a required parameter.

You can use the **-gt** parameter on the fteCreateTransfer command to generate a template XML document that contains your file transfer request. The monitor uses the transfer template as its task definition.

-rl (*number_of_recursion_levels***)**
Optional. The level of monitoring recursion of the root monitoring directory, that is how many levels of subdirectory to go down into. For example in a directory structure like the following example with `C:\wmqfte\monitor` set as the root monitoring directory:

```
C:\wmqfte\monitor
C:\wmqfte\monitor\reports
C:\wmqfte\monitor\reports\2009
C:\wmqfte\monitor\reports\2009\April
```

If you specify `-rl 2`, WebSphere MQ Managed File Transfer only searches as far down as the `C:\wmqfte\monitor\reports\2009` directory and its sibling directories. The `C:\wmqfte\monitor\reports\2009\April` directory is ignored. By default, recursion is set to none.

-pi (*interval_period***)**
Optional. The interval period between each monitor of a directory. The poll interval must be a positive integer value. The default value for **-pi** is 1.

-pu (*units***)**
Optional. The time units for the monitor poll interval. If you specify the **-pu** parameter, you must also specify the **-pi** parameter. The default value for **-pu** is `minutes`. Specify one of the following options:

seconds

minutes

hours

days

-tr
Optional. Specifies the trigger condition that must be satisfied for the defined task to take place. If the condition is not satisfied, according to the source agent, the monitor task (for example the file transfer) is not started. A trigger condition consists of two optional parts, condition and pattern, separated by a comma. Specify one of the following formats:
• *condition*,*pattern*

where *condition* is one of the following values:

match For each trigger that is satisfied, the defined task is performed. `match` is the default value.

For example, if the match is `*.go` and the files `LONDON.go` and `MANCHESTER.go` are present, the task is performed for `LONDON.go` and another task is performed for `MANCHESTER.go`.

If the same trigger file is present from a previous poll (that is, the file has not been modified), this file has a not satisfied trigger condition. That is, the match trigger file must be new and must have been modified since last the poll before the defined task is performed.

noMatch
No files in the monitored directory match the pattern. That is, if *any* of the files in the monitored directory do not exist, the condition is satisfied. If no files match the trigger condition at the time the monitor is created, the monitor starts instantly, but does not start again until a file match is found, and then removed.

noSizeChange=*n*
A minimum of one of the files in the directory matches the pattern and has a file size that does not change for *n* polling intervals. The value of *n* is a positive integer.

fileSize>=*size*
A minimum of one of the files in the directory matches the pattern and has a minimum file size greater or equal to *size*. The value *size* is a combination of an integer with an optional size unit of B, KB, MB, or GB. For example, `fileSize">"=10KB`. If you do not specify a size unit, the default size used is bytes. On all operating systems, you must enclose the greater than symbol (>) in double quotation marks when you specify the `fileSize` option on the command line, as shown in this example.

The pattern is a file pattern match sequence in wildcard or Java regular expression format. The default value for the pattern is `*`, or match any file, and the default format is wildcard format. Use the **-pt** to specify the format of the pattern.

For example, the following trigger condition is satisfied when a file exists in the monitored directory with the suffix `.go`.

`-tr match,*.go`

The following trigger condition is satisfied when there are no files in the monitored directory which have the suffix `.stop`.

`-tr noMatch,*.stop`

You can only specify *condition*,*pattern* if you have also specified the **-md** parameter.

- *condition*

where *condition* is one of the following values:

queueNotEmpty
The monitored queue is not empty. That is, if there are *any* WebSphere MQ messages on the monitored queue, the condition is satisfied. A single task is run for all of the messages on the queue.

completeGroups
There is a complete group on the monitored queue. That is, if *any* of the WebSphere MQ message groups on the monitored queue are complete, the condition is satisfied. An individual task is run for each complete group on the queue.

If a single message that is not in a group is put on the queue, it is treated as if it is a complete group and a task is run for the single message.

You can only specify *condition* if you have also specified the **-mq** parameter.

For each monitor that you create, you can specify the **-tr** parameter once only.

-x (*exclude_pattern***)**

Optional. Specifies files that are excluded from the trigger pattern match. The trigger pattern is specified by the **-tr** parameter.

The pattern is a file pattern match sequence in wildcard or Java regular expression format. The default format is wildcard format. Use the **-pt** parameter to specify the format of the pattern.

-jn (*job name***)**

Optional. Specifies a job name reference, which is a user-defined identifier for the request.

-pt (*pattern_type***)**

Optional. The type of pattern that is used by the **-tr** and **-x** parameters. Valid values are:

wildcard

The patterns are evaluated as wildcard patterns. An asterisk (*) matches zero or more characters and a question mark (?) matches exactly one character. This is the default.

regex The patterns are evaluated as Java regular expressions. For more information, see "Regular expressions used by WebSphere MQ Managed File Transfer" on page 685.

-bs (*matches_per_task***)**

Optional. The maximum number of trigger matches to include in a single task. For example, if a value of 5 is specified for *matches_per_task* and nine trigger matches occur in a single poll interval, two tasks are performed. The first task corresponds to triggers 1-5 inclusive, and the second task corresponds to triggers 6-9. The default value of *matches_per_task* is 1.

The **-bs** parameter is supported only when the task definition XML that you supply to the **-mt** parameter is a managedTransfer. A managedCall is not supported with the **-bs** parameter.

-dv (*default_variables***)**

Optional. A comma-separated list of default variables that can be used in variable substitution when monitoring a queue. The values are in the format of a key-value pair. For example:

```
-dv size=medium,color=blue
```

For more information about variable substitution, see "Customizing tasks with variable substitution" on page 225. You can only specify the **-dv** parameter if you have also specified the **-mq** parameter.

-? or -h

Optional. Displays command syntax.

-p (*configuration_options***)**

Optional. This parameter determines the set of configuration options to use to cancel the transfer. By convention use the name of a nondefault coordination queue manager as the input for this parameter. The command then uses the set of properties files associated with this nondefault coordination queue manager.

If you do not specify this parameter, the set of configuration options based on the default coordination queue manager is used.

Examples

In this example, a new resource monitor is created called MYMONITOR using the monitoring agent MYAGENT. Provided the trigger condition that a file larger than 5 MB is present in the directory `C:\wmqfte\monitors`, the file transfer defined in the file `C:\templates\transfer_reports.xml` is started. MYAGENT is also the source agent for the file transfer defined in `C:\templates\transfer_reports.xml`:

```
fteCreateMonitor -ma MYAGENT -md C:\wmqfte\monitors -mn MYMONITOR -mt C:\templates\transfer_reports.xml
 -tr fileSize">"=5MB,*.go
```

In this example, a resource monitor called MONITOR1 using the agent AGENT1 is created to transfer files greater than 5 MB and is exported to the XML file monitor.xml.

```
fteCreateMonitor -ox monitor.xml -ma AGENT1 -mn MONITOR1 -mt task.xml -tr "fileSize>=5MB,*.zip"
```

Then the XML file is imported and changed to exclude any files greater than 10MB.

```
fteCreateMonitor -ix monitor.xml -x "fileSize>=10MB,*.zip" -f
```

In this example, a new resource monitor is created called MYMONITOR using the agent MYAGENT.

```
fteCreateMonitor -ma MYAGENT -md c:\wmqfte -mn MYMONITOR -mt c:\templates\transfer_reports.xml -tr "fileSize>=5MB,*.go"
```

However the trigger is initially incorrectly set to monitor c:\wmqfte rather than c:\wmqfte\monitors. The **fteCreateMonitor** request is immediately re-issued with the monitor directory corrected and the **-f** (overwrite) and **-c** (clear history) parameters used to update the monitor.

```
fteCreateMonitor -ma MYAGENT -md c:\wmqfte\monitors -mn MYMONITOR -mt c:\templates\transfer_reports.xml
-tr "fileSize>=5MB,*.go" -f -c
```

Return codes

Return code	Description
0	Command completed successfully.
1	Command ended unsuccessfully.

fteCreateTemplate (create new file transfer template)

The **fteCreateTemplate** command creates a file transfer template that you can keep for future use. The only required parameter is the **-tn** (*template_name*) parameter. All other parameters are optional, although if you specify a source file specification, you must also provide a destination file. Similarly, if you specify a destination file, you must also specify a source file specification.

fteCreateTemplate
Purpose

Use the **fteCreateTemplate** command to create a file transfer template that stores your transfer details until you want to use them at a later date. Use transfer templates to store common file transfer settings for repeated or complex transfers. After you have created a transfer template, submit the template using the WebSphere MQ Explorer. You cannot submit a transfer template from the command line.

The transfer template that you create using the **fteCreateTemplate** command is not the same as the XML message that you create using the **-gt** parameter on the **fteCreateTransfer** command. You cannot use the two different types of template interchangeably.

You can run the **fteCreateTemplate** command from any system that can connect to the WebSphere MQ network and then route to the coordination queue manager. Specifically for the command to run, you must have installed WebSphere MQ Managed File Transfer on this system and you must have configured the WebSphere MQ Managed File Transfer component on this system to communicate with the WebSphere MQ network.

This command uses the `command.properties` file to connect to the WebSphere MQ network. If the `command.properties` file does not contain property information, a bindings mode connection is made to the default queue manager on the local system. If the `command.properties` file does not exist, an error is generated. For more information, see "The command.properties file" on page 545.

You can specify multiple source files for a file transfer but only one destination agent; transferring one file to multiple destination agents is not supported. However, you can transfer multiple source files to multiple destination files on a single destination agent.

For guidance about how to transfer files, see "Guidelines for transferring files" on page 668.

Special characters

Take care when you use parameters that contain special characters so that you avoid the command shell interpreting the characters in a way you do not expect. To avoid characters being interpreted by the command shell, enclose the entire parameter in double quotation marks as shown in the final two examples "Examples" on page 460, or escape the special characters using the escape sequence of the command shell.

Relative paths

The **fteCreateTemplate** command supports the use of relative file paths. On distributed systems by default paths are considered to be relative to the home directory of the user that the agent is running as. To change the directory that path names are evaluated relative to, set the transferRoot property in the `agent.properties` file. This file is located in the *MQ_DATA_PATH*/mqft/config/*coordination_qmgr*/agents/ *agent_name* directory. Add the following line to the file:

`transferRoot=`*directory_name*

You must escape Windows paths or write them in UNIX format. For example, specify `C:\TransferRoot` as `C\:\\TransferRoot` or `C:/TransferRoot`

Syntax

fteCreateTemplate

```
         ┌─────────┐              ┌───────────┐
►────────┤ -tb──┬─ADMIN──┬──────┤ -jn ──job_name─┤──────────────────────────►
              ├─SOURCE─┤
              └─UTC────┘
```

```
       ┌────────────────────────────────────────────────────────────────────┐
►──────┤ -ss──schedule_start_time──oi──┬─minutes─┬──-of──occurrence_frequency──┬──────────────────────────────┬──►
                                   ├─hours──┤                           ├─-oc──occurrence_count──┤
                                   ├─days───┤                           └─-es──schedule_end_time─┘
                                   ├─weeks──┤
                                   ├─months─┤
                                   └─years──┘
```

```
►──── -tn ────template_name─┬──────────┬──┬─ -sqdt ──text_delimiter──┬──────────────────────┬─┬──►
                         └─ -sqgi ──┘                            └─ -sqdp ──┬─prefix──┬─┘
                                                                         └─postfix─┘
```

```
       ┌──────────────────────────────────────────────────┐
►──────┤ -sqdb ──hexadecimal_delimiter─┬────────────────────────┬─┤ -sqwt ──wait_time──┤ -sq ─┤──►
                                    └─ -sqdp ──┬─prefix──┬─┘
                                             └─postfix─┘
```

```
►──────┬─────────────────────┬──────────────────────────────────────────────────────◄
       └─source_specification─┘
```

File splitting options:

```
├──┬─ -qs ──message_size───────────────────────────────────────────────┬──┤
   ├─ -dqdb ──hexadecimal_delimiter─┬───────┬──┬─ -dqdp ──┬─prefix──┬─┬─┤
   │                              └─ -qi ─┘            └─postfix─┘ │
   └─ -dqdt ──pattern_delimiter──┬───────┬──┬─ -dqdp ──┬─prefix──┬─┘
                               └─ -qi ─┘            └─postfix─┘
```

Parameters

-sa *source_agent_name*

Optional. The name of the agent that the source file is transferred from. If you do not specify this agent name when you create the template, you must specify the source agent name when you use the template.

-sm *source_agent_qmgr_name*

Optional. The name of the queue manager that the source agent is connected to.

If you do not specify the **-sm** parameter, the queue manager used is determined by the set of configuration options in use, based on the source agent name. If the queue manager name cannot be determined using these options, the transfer template creation fails. For example, the template creation fails if the agent.properties file for the source agent cannot be found.

-da *destination_agent_name*
Optional. The name of the agent that the file is transferred to. If you do not specify the destination agent name when you create the template, you must specify the destination agent name when you use the template.

-dm *destination_agent_qmgr_name*
Optional. The name of the queue manager that the destination agent is connected to.

If you do not specify the **-dm** parameter, the queue manager used is determined by the set of configuration options in use, based on the destination agent name. If the queue manager name cannot be determined using these options, the transfer template creation fails. For example, the template creation fails if the `agent.properties` file for the destination agent cannot be found.

-td *transfer_definition_file*

Optional. The name of the XML document that defines one or more source and destination file specifications for the transfer.

One of the **-td**, **-df**, **-dd**, **-ds**, **-dq**, **-du**, and **-dp** parameters is required. If you specify the **-td** parameter, you cannot specify source files, or specify the **-df**, **-dd**, **-ds**, **-dp**, **-dq**, **-du**, **-sd**, **-r**, **-de**, **-t**, or **-cs** parameters.

The **fteCreateTemplate** command locates the transfer definition file in relation to your current directory. If you cannot use relative path notation to specify the location of the transfer definition file, use the fully qualified path and file name of the transfer definition file instead.

For more information, see Using transfer definition files.

-df *destination_file*
Optional. The name of the destination file. Specify a file name that is valid on the system where the destination agent is running.

If the destination agent is a Connect:Direct bridge agent, the destination file is specified in the format *connect_direct_node_name:file_path*. The Connect:Direct bridge agent accepts only file paths that are specified in this format. If the destination agent is a Connect:Direct bridge agent and the destination is a PDS member, you must also specify the **-de** parameter with a value of overwrite.

One of the **-td**, **-df**, **-dd**, **-ds**, **-dq**, **-du**, and **-dp** parameters is required. If you specify the **-df** parameter, you cannot specify the **-td**, **-dd**, **-dp**, **-dq**, **-du**, or **-ds** parameters because these parameters are mutually exclusive.

-dd *destination_directory*
Optional. The name of the directory the file is transferred to. Specify a directory name that is valid on the system where the destination agent is running.

If the destination agent is a Connect:Direct bridge agent, the destination directory is specified in the format *connect_direct_node_name:directory_path*. If the destination agent is a Connect:Direct bridge agent and the destination is a PDS, you must also specify the **-de** parameter with a value of overwrite.

One of the **-td**, **-df**, **-dd**, **-ds**, **-dq**, **-du**, and **-dp** parameters is required. If you specify the **-dd** parameter, you cannot specify the **-td**, **-df**, **-dp**, **-dq**, **-du**, or **-ds** parameters because these parameters are mutually exclusive.

-du *destination_user*

Optional. The name of the user whose destination file space the files are transferred into. For more information about file spaces, see "File spaces" on page 313.

One of the **-td**, **-df**, **-dd**, **-ds**, **-dp**, **-du**, and **-dq** parameters is required. If you specify the **-du** parameter, you cannot specify the **-td**, **-dd**, **-df**, **-dp**, **-dq**, or **-ds** parameters because these parameters are mutually exclusive.

The **-du** parameter is not supported when the destination agent is a protocol bridge agent or a Connect:Direct bridge agent.

-dq *destination_queue*

Optional. The name of a destination queue that files are transferred onto. You can optionally include a queue manager name in this specification, using the format QUEUE@QUEUEMANAGER. If you do not specify a queue manager name, the destination agent queue manager name is used. You must specify a valid queue name that exists on the queue manager.

One of the **-td**, **-df**, **-dd**, **-ds**, **-dp**, **-du**, and **-dq** parameters is required. If you specify the **-dq** parameter, you cannot specify the **-td**, **-dd**, **-df**, **-dp**, **-du**, or **-ds** parameters because these parameters are mutually exclusive.

The **-dq** parameter is not supported when the destination agent is a protocol bridge agent or a Connect:Direct bridge agent, or when the source specification is a queue.

-dqp *persistent*

Optional. Specifies whether messages written to the destination queue are persistent. The valid options are as follows:

true   Writes persistent messages to the destination queue. This is the default value.

false   Writes non-persistent messages to the destination queue.

qdef   The persistence value is take from the DefPersistence attribute of the destination queue.

You can only specify the **-dqp** parameter if you have also specified the **-dq** parameter.

-qmp *boolean*

Optional. Specifies whether the first message written to the destination queue by the transfer has WebSphere MQ message properties set. The valid options are as follows:

true   Sets message properties on the first message created by the transfer.

false   Does not set message properties on the first message created by the transfer. This is the default value.

You can only specify the **-qmp** parameter if you have also specified the **-dq** parameter. For more information, see "WebSphere MQ message properties set on messages written to destination queues" on page 701

-qs *message_size*

Optional. Specifies whether to split the file into multiple fixed-length messages. All the messages have the same WebSphere MQ group ID; the last message in the group has the WebSphere MQ LAST_MSG_IN_GROUP flag set. The size of the messages is specified by the value of *message_size*. The format of *message_size* is *<length><units>*, where *length* is a positive integer value and *units* is one of the following values:

B   Bytes. The minimum value allowed is two times the maximum bytes-per-character value of the code page of the destination messages.

K   This is equivalent to 1024 bytes.

M   This is equivalent to 1048576 bytes.

If you specify the value `text` for the **-t** parameter and the file is in a double byte character set or multibyte character set, the file is split into messages on the closest character boundary to the specified message size.

You can only specify the **-qs** parameter if you have also specified the **-dq** parameter. You can only specify one of the **-qs**, **-dqdb**, and **-dqdt** parameters.

-dqdb *hexadecimal_delimiter*

Optional. Specifies the hexadecimal delimiter to use when splitting a binary file into multiple messages. All the messages have the same WebSphere MQ group ID; the last message in the group has the WebSphere MQ LAST_MSG_IN_GROUP flag set. The format for specifying a hexadecimal byte as a delimiter is xNN, where N is a character in the range 0-9 or a-f. You can specify a sequence of hexadecimal bytes as a delimiter by specifying a comma-separated list of hexadecimal bytes, for example: x3e,x20,x20,xbf.

You can only specify the **-dqdb** parameter if you have also specified the **-dq** parameter and the transfer is in binary mode. You can only specify one of the **-qs**, **-dqdb**, and **-dqdt** parameters.

-dqdt *pattern*

Optional. Specifies the regular expression to use when splitting a text file into multiple messages. All the messages have the same WebSphere MQ group ID; the last message in the group has the WebSphere MQ LAST_MSG_IN_GROUP flag set. The format for specifying a regular expression as a delimiter is a regular expression enclosed in parentheses, (*regular_expression*). The value of this parameter is evaluated as aJava regular expression. For more information, see "Regular expressions used by WebSphere MQ Managed File Transfer" on page 685.

By default, the length of the string that the regular expression can match is limited by the destination agent to five characters. You can change this behavior using the `maxDelimiterMatchLength` agent property. For more information, see "Advanced agent properties" on page 549.

You can only specify the **-dqdt** parameter if you have also specified the **-dq** parameter and the value text for the **-t** parameter. You can specify only one of the **-qs**, **-dqdb**, and **-dqdt** parameters.

-dqdp

Optional. Specifies the expected position of destination text and binary delimiters when splitting files. You can only specify the **-dqdp** parameter if you have also specified one of the **-dqdt** and **-dqdb** parameters.

Specify one of the following options:

prefix The delimiters are expected at the beginning of each line.

postfix
 The delimiters are expected at the end of each line. This is the default option.

-qi

Optional. Specifies whether to include the delimiter that is used to split the file into multiple messages in the messages. If **-qi** is specified, the delimiter is included at the end of the message that contains the file data preceding the delimiter. By default the delimiter is not included in the messages.

You can only specify the **-qi** parameter if you have also specified one of the **-dqdt** and **-dqdb** parameters.

-de *destination_file_behavior*
Optional. Specifies the action that is taken if a destination file exists on the destination system. The valid options are as follows:

error Reports an error and the file is not transferred. This is the default value.

overwrite
 Overwrites the existing destination file.

If you specify the **-de** parameter, you cannot specify the **-td** parameter because these parameters are mutually exclusive.

-sd *source_file_disposition*

Optional. Specifies the action that is taken on a source file when that source file has successfully been transferred to its destination. The valid options are as follows:

leave The source files are left unchanged. This is the default value.

delete The source file is deleted from the source system after the source file is successfully transferred.

If the source is a queue and you specify the `leave` option, the command returns an error and a transfer is not requested.

If the source agent is a Connect:Direct bridge agent and you specify the `delete` option, the behavior is different to the usual source disposition behavior. One of the following cases occurs:

- If Connect:Direct uses a process that is generated by WebSphere MQ Managed File Transfer to move the file or data set from the source, specifying the `delete` option causes the transfer to fail. To specify that the source file is deleted, you must submit a user-defined Connect:Direct process. For more information, see "Submitting a user-defined Connect:Direct process from a file transfer request" on page 269.
- If Connect:Direct uses a user-defined process to move the file or data set from the source, this parameter is passed to the process through the **%FTEFDISP** intrinsic symbolic variable. The user-defined process determines whether the source is deleted. The result that the transfer returns depends on the result that is returned by the user-defined process.

If you specify the **-sd** parameter, you cannot specify the **-td** parameter because these parameters are mutually exclusive. However, you can specify source disposition behavior in the transfer definition file.

-pr *transfer_priority*

Optional. Specifies the priority level of the transfer. Priority is a value in the range 0-9, where 0 is the lowest priority. The default priority level is 0 and by default the transfer uses the priority level of the source agent.

This value matches the message priority value used by WebSphere MQ, see Getting messages from a queue: priority for more information. Message traffic for file transfer data defaults to a priority level of 0, which allows your WebSphere MQ message traffic to take priority.

-p *configuration_options*

Optional. This parameter determines the set of configuration options that is used to create the transfer template. Use the name of a non-default coordination queue manager as the input for this parameter. The command then uses the set of properties files associated with this non-default coordination queue manager.

If you do not specify this parameter, the set of configuration options based on the default coordination queue manager is used.

-r

Optional. Recursively transfer files in subdirectories when *source_specification* contains wildcard characters. When WebSphere MQ Managed File Transfer is presented with a wildcard character as a *source_specification*, any directories that match the wildcard character are transferred only if you have specified the **-r** parameter. When *source_specification* matches a subdirectory, all files in that directory and its subdirectories (including hidden files) are always transferred.

For more information about how WebSphere MQ Managed File Transfer handles wildcard characters, see Using wildcard characters

If you specify the **-r** parameter, you cannot specify the **-td** parameter because these parameters are mutually exclusive. However, you can specify recursive behavior in the transfer definition file.

-t Optional. Specifies the type of file transfer: binary mode or text mode.

binary The data in the file is transferred without any conversion. This is the default value.

text   The code page and end-of-line characters of the file are converted. The exact conversions performed depend on the operating systems of the source agent and destination agent.

If you specify the **-t** parameter, you cannot specify the **-td** parameter because these parameters are mutually exclusive. However, you can specify transfer mode behavior in the transfer definition file.

-cs

Optional. Specifies whether a checksum algorithm is run on the file transfer data to check the integrity of the transferred files. Specify one of the following options:

MD5   Computes an MD5 checksum for the data. The resulting checksum for the source and destination files is written to the transfer log for validation purposes. By default, WebSphere MQ Managed File Transfer computes MD5 checksums for all file transfers.

none   No MD5 checksum is computed for the file transfer data. The transfer log records that checksum was set to none and the value for the checksum is blank. For example:

```
<checksum method="none"></checksum>
```

If you use the `none` option, you might improve file transfer performance, depending on your environment. However, selecting this option means that there is no validation of the source or destination files.

If you specify the **-cs** parameter, you cannot specify the **-td** parameter because these parameters are mutually exclusive. However, you can specify checksum behavior in the transfer definition file.

-tr

Optional. Specifies a condition that must be true for this file transfer to take place. If the condition is not true, according to the source agent, the file transfer is discarded and no transfer takes place. Specify the following format:

condition,namelist

where *condition* is one of the following values:

file=exist
A minimum of one of the files in the namelist exists. That is, if *any* of the files in the namelist exists, the condition is true.

file!=exist
A minimum of one of the files in the namelist does not exist. That is, if *any* of the files in the namelist do not exist, the condition is true.

filesize>=_size_
A minimum of one of the files in the namelist exists and has a minimum size as specified by *size*. The value of *size* is an integer with an optional size unit of KB, MB, or GB. For example, `filesize">"=10KB`. If you do not specify a size unit, the size is assumed to be bytes. On all operating systems, you must enclose the greater than symbol (>) in double quotation marks when you specify the `filesize` option on the command line, as shown in this example.

And where *namelist* is a comma-separated list of file names located on the source system. Depending on your operating system, if you want to use path names or file names in a namelist that contain spaces, you might have to enclose the path names and file names in double quotation marks. You can specify more than one trigger condition by using the **-tr** parameter more than once. However in that case, every separate trigger condition must be true for the file transfer to take place.

Note: To continually monitor a resource for a trigger condition to be true, you are recommended to use resource monitoring. You can create a resource monitor using the fteCreateMonitor command.

In the following example, the file `file1.doc` is transferred from AGENT1 to AGENT2, on condition that either file `A.txt`, or file `B.txt`, or both files exist on AGENT1 *and* that either file `A.txt`, or file `B.txt`, or both files are equal to or larger than 1 GB:

```
fteCreateTemplate -tn JUPITER_AGENT_TRIGGER_TEST_TEMPLATE -sa AGENT1 -sm QM_JUPITER -da AGENT2 -dm QM_NEPTUNE
-tr file=exist,C:\export\A.txt,C:\export\B.txt
-tr filesize">"=1GB,C:\export\A.txt,C:\export\B.txt
-df C:\import\file1.doc C:\export\file1.doc
```

You can combine triggering parameters with scheduling parameters. If you do specify both types of parameters, the trigger conditions are applied to the file transfer created by the scheduling parameters.

-tl

Optional. Specifies whether trigger failures are logged. Specify one of the following options:

yes   Log entries are created for failed triggered transfers. This is the default behavior even if you do not specify the **-tl** parameter.

no   No log entries are created for failed triggered transfers.

-md

Optional. Specifies the user-defined metadata that is passed to the exit points of the agent. The **-md** parameter can take one or more name-value pairs separated by commas. Each name pair consists of <name>=<value>. You can use the **-md** parameter more than once in a command.

-tb

Optional. Specifies the time base you want to use for the scheduled file transfer. That is, whether you want to use a system time or Coordinated Universal Time (UTC). You must use this parameter with the **-ss** parameter only. Specify one of the following options:

admin   The start and end times used for the scheduled transfer are based on the time and date of the system used by the administrator. This is the default value.

source   The start and end times used for the scheduled transfer are based on the time and date of the system where the source agent is located.

UTC   The start and end times used for the scheduled transfer are based on Coordinated Universal Time (UTC).

-jn *job_name*

Optional. A user-defined job name identifier that is added to the log message when the transfer has started.

-ss *schedule_start_time*

Optional. Specifies the time and date that you want the scheduled transfer to take place. Use one of the following formats to specify the time and date. Specify the time using the 24-hour clock:

`yyyy-MM-ddThh:mm`

`hh:mm`

Scheduled file transfers start within a minute of the schedule start time, if there are no problems that might affect the transfer. For example, there might be issues with your network or agent that prevent the scheduled transfer starting.

-oi

Optional. Specifies the interval that the scheduled transfer occurs at. You must use this parameter with the **-ss** parameter only. Specify one of the following options:

minutes

hours

days

weeks

months

years

-of *occurrence_frequency*

   Optional. Specifies the frequency that the scheduled transfer occurs at. For example, every **5** weeks or every **2** months. You must specify this parameter with the **-oi** and **-ss** parameters only. If you do not specify this parameter, a default value of 1 is used.

-oc *occurrence_count*

   Optional. Specifies how many times you want this scheduled transfer to occur. After the occurrence count has been met, the scheduled transfer is deleted.

   Specify this parameter with the **-oi** and **-ss** parameters only.

   If you specify the **-oc** parameter, you cannot specify the **-es** parameter because these parameters are mutually exclusive.

   You can omit both the **-oc** and **-es** parameters to create a transfer that repeats indefinitely.

-es *schedule_end_time*

   Optional. The time and date that a repeating scheduled transfer ends.

   You must specify this parameter with the **-oi** and **-ss** parameters only.

   If you specify the **-es** parameter, you cannot specify the **-oc** parameter because these parameters are mutually exclusive.

   You can omit both the **-es** and **-oc** parameters to create a transfer that repeats indefinitely.

   Use one of the following formats to specify the end time and date. Specify the time using the 24-hour clock:

   `yyyy-MM-ddThh:mm`

   `hh:mm`

-tn *template_name*

   Required. The name of the template that you want to create. Use a descriptive string that allows you to select the correct template for transfers at a later date. There is no specific limit to the length of this string, but be aware that excessively long names might not be displayed properly in some user interfaces.

   Do not create multiple templates with the same name.

-sqgi

   Optional. Specifies that the messages are grouped by WebSphere MQ group ID. The first complete group is written to the destination file. If this parameter is not specified, all messages on the source queue are written to the destination file.

   You can only specify the **-sqgi** parameter if you have also specified the **-sq** parameter.

-sqdt *text_delimiter*

   Optional. Specifies a sequence of text to insert as the delimiter when appending multiple messages to a text file. You can include Java escape sequences for String literals in the delimiter. For example, `-sqdt \u007d\n`.

   You can only specify the **-sqdt** parameter if you have also specified the **-sq** parameter and the value `text` for the **-t** parameter.

-sqdb *hexadecimal_delimiter*

   Optional. Specifies one or more byte values to insert as the delimiter when appending multiple messages to a binary file. Each value must be specified as two hexadecimal digits in the range `00-FF`, prefixed by x. Multiple bytes must be comma-separated. For example, `-sqdb x08,xA4`.

   You can only specify the **-sqdb** parameter if you have also specified the **-sq** parameter. You cannot specify the **-sqdb** parameter if you have also specified the value `text` for the **-t** parameter.

-sqdp

Optional. Specifies the position of insertion of source text and binary delimiters. You can only specify the **-sqdp** parameter if you have also specified one of the **-sqdt** and **-sqdb** parameters.

Specify one of the following options:

prefix   The delimiters are inserted at the start of each message

postfix
>   The delimiters are inserted at the end of each message. This is the default option.

-sqwt *wait_time*

>   Optional. Specifies the time, in seconds, to wait for one of the following conditions to be met:
>   - For a new message to be put on the queue
>   - If the **-sqgi** parameter was specified, for a complete group to be put on the queue
>
>   If neither of these conditions are met within the time specified by *wait_time*, the source agent stops reading from the queue and completes the transfer. If the **-sqwt** parameter is not specified, the source agent stops reading from the source queue immediately if the source queue is empty or, in the case where the **-sqgi** parameter is specified, if there is no complete group on the queue.
>
>   You can only specify the **-sqwt** parameter if you have also specified the **-sq** parameter.

-sq

>   Optional. Specifies that the source of a transfer is a queue.

source_specification

>   Required if you have specified one of the **-df**, **-dd**, **-dp**, **-dp**, or **-ds** parameters. If you specify the **-td** parameter, do not specify *source_specification*.
>   - If you have not specified the **-sq** parameter, *source_specification* is one or more file specifications that determine the source, or sources, for the file transfer. File specifications are space delimited. File specifications can take one of five forms and can include wildcard characters. For more information about wildcard characters in WMQFTE, see "Using wildcard characters" on page 683. You can escape asterisks that are part of the file specification by using two asterisk characters (**) in the file specification.
>
>   To transfer files containing spaces in their file names, place double quotation marks around the file names that contain spaces. For example to transfer file a b.txt to file c d.txt specify the following text as part of the **fteCreateTemplate** command:
>
>       ```
>       -df "c d.txt" "a b.txt"
>       ```
>
>   Each file specification must be in one of the following formats:
>
>   **File names**
>>   The name of a file, expressed using the appropriate notation for the system where the source agent is running. When a file name is specified as a source file specification, the contents of the file are copied.
>
>   **Directories**
>>   The name of a directory, expressed using the appropriate notation for the system where the source agent is running. When a directory is specified as a source file specification, the contents of the directory are copied. More precisely, all files in the directory and in all its subdirectories, including hidden files, are copied.
>>
>>   For example, to copy the contents of DIR1 to DIR2 only, specify `DIR1/* DIR2`
>
>   **File name or directory at a Connect:Direct node**
>>   (Connect:Direct bridge agent only). The name of a Connect:Direct node, a colon character (:), and a file or directory path on the system that is hosting the Connect:Direct node. For example, *connect_direct_node_name:file_path*.

If the source agent is a Connect:Direct bridge agent, it will only accept source specifications in this form.

Note: Wildcard characters are not supported in file paths when the source agent is a Connect:Direct bridge agent.

- If you have specified the **-sq** parameter, *source_specification* is the name of a local queue on the source agent queue manager. You can specify only one source queue. The source queue is specified in the format:

  QUEUE_NAME

  The queue manager name is not included in the source queue specification, because the queue manager must be the same as the source agent queue manager.

-? or -h
    Optional. Displays command syntax.

Examples

In this example, a transfer template called `payroll accounts monthly report template` is created. When submitted, this template transfers any file with the extension .xls from the agent PAYROLL1 to the agent ACCOUNTS in the directories specified:

```
fteCreateTemplate -tn "payroll accounts monthly report template" -sa PAYROLL -sm QM_PAYROLL1 -da ACCOUNTS
-dm QM_ACCOUNTS -df C:\payroll_reports\*.xls C:\out\*.xls
```

In this example, a transfer template called `jupiter_neptune_sched_template` is created. When submitted, the template transfers the file `originalfile.txt` from the system where QM_JUPITER is located to the system where QM_NEPTUNE is located. The file transfer is scheduled to take place at 09:00 based on the system time of the system where the source agent is located and occurs every two hours four times:

```
fteCreateTemplate -tn jupiter_neptune_sched_template -sa AGENT1 -sm QM_JUPITER -da AGENT2 -dm QM_NEPTUNE
-tb source -ss 09:00 -oi hours -of 2 -oc 4
-df C:\import\transferredfile.txt C:\export\originalfile.txt
```

In this example, a transfer template called `jupiter neptune trigger template` is created. When the template is submitted, the file `originalfile.txt` is transferred from AGENT1 to AGENT2, on condition that the file `A.txt` exists on AGENT1:

```
fteCreateTemplate -tn "jupiter neptune trigger template" -sa AGENT1 -sm QM_JUPITER -da AGENT2 -dm QM_NEPTUNE
-tr file=exist,C:\export\A.txt -df C:\import\transferredfile.txt C:\export\originalfile.txt
```

Return codes

Return code	Description
0	Command completed successfully.
1	Command ended unsuccessfully.

fteCreateTransfer (create new file transfer)

The **fteCreateTransfer** command creates and starts a new file transfer from the command line. This command can start a file transfer immediately, schedule a file transfer for a future time and date, repeat a scheduled transfer one or more times, and trigger a file transfer based on certain conditions.

fteCreateTransfer
Purpose

Use the **fteCreateTransfer** command to create and then start a new file transfer from a WebSphere MQ Managed File Transfer agent. For guidance about how to transfer files, including text files, data sets, and generation data groups (GDGs), see "Guidelines for transferring files" on page 668.

You can run the **fteCreateTransfer** command from any system that can connect to the WebSphere MQ network and then route to the source agent queue manager. Specifically, for the command to run, you must install a WebSphere MQ Managed File Transfer component (either Service or Agent) on this system and configure the WebSphere MQ Managed File Transfer component on this system to communicate with the WebSphere MQ network.

This command uses a properties file called `command.properties` to connect to the WebSphere MQ network. If the `command.properties` file does not contain property information, a bindings mode connection is made to the default queue manager on the local system. If the `command.properties` file does not exist, an error is generated. For more information, see "The command.properties file" on page 545.

You can specify multiple source files for a file transfer but they must originate from a single source agent and terminate at a single destination agent. Transferring a single source file to multiple destination files on the same agent or multiple different agents is not supported within a single transfer. Ant scripting can be used to send the same source file to multiple destinations at one or more agents. For more information, see "Using Apache Ant with WebSphere MQ Managed File Transfer" on page 328.

z/OS

WebSphere MQ Managed File Transfer Version 7.5 is not available for z/OS. However, this topic contains information about z/OS to allow you to transfer data sets, for example, to and from a version of WebSphere MQ File Transfer Edition running on a z/OS system.

Special characters

Take care when you use parameters that contain special characters so that you avoid the command shell interpreting the characters in a way you do not expect. For example, fully qualified data set names that contain single quotation marks and source specifications that contain asterisk characters might be interpreted by the command shell rather than being passed through in the transfer request. To avoid characters being interpreted by the command shell, enclose the entire parameter in double quotation marks or escape the special characters by using the escape sequence of the command shell.

Relative paths

The **fteCreateTransfer** command supports the use of relative file paths. On distributed systems and z/OS UNIX System Services, by default paths are considered to be relative to the home directory of the user that the agent is running as. To change the directory that path names are evaluated relative to, set the transferRoot property in the `agent.properties` file. This file is located in the *MQ_DATA_PATH*/mqft/config/*coordination_qmgr*/agents/*agent_name* directory. Add the following line to the file:

`transferRoot=directory_name`

You must escape Windows paths or write them in UNIX format. For example, specify `C:\TransferRoot` as `C\:\\TransferRoot` or `C:/TransferRoot`

On z/OS, by default the user name that the agent is running under is added as a high-level qualifier prefix to data set specifications that have not been fully qualified. For example: //ABC.DEF. To change the value that is added as a prefix to the data set name, set the transferRootHLQ property in the `agent.properties` file. This file is located in the *MQ_DATA_PATH*/mqft/config/*coordination_qmgr*/agents/ *agent_name* directory. Add the following line to the file:

`transferRootHLQ=`*prepend_value*

However, for transfers that involve a Connect:Direct node on a z/OS system, the data set specification is interpreted as a fully qualified name. No high-level qualifier is added to the data set name.

Syntax

---

fteCreateTransfer

```
├──┬─────────────────────────────────────┬──┬──────────┬───────────────►
   └─ -srdb ─ delimiter ─┬────────────────┤  └─ -skeep ─┘
                         └─ -srdp ─┬prefix─┤
                                   └postfix┘
```

```
►──┬───────────────────────────────────────────────────────────┬── source_specification ──►◄
   └─ -sq ─┬────────┬─┬─ -sqdt ─ text_delimiter ──────────────────┬─┬─────────────────────┬─┘
           └─ -sqgi ┘ │         └─ -sqdp ─┬prefix─┤               │ └─ -sqwt ─ wait_time ─┘
                      │                   └postfix┘               │
                      └─ -sqdb ─ hexadecimal delimiter ───────────┤
                                      └─ -sqdp ─┬prefix─┤         
                                                └postfix┘
```

File splitting options:

```
├──┬─ -qs ─ message_size ─────────────────────────────────────────────┬──┤
   ├─ -dqdb ─ hexadecimal_delimiter ──────────────────────────────────┤
   │                    └─ -qi ─┘ └─ -dqdp ─┬prefix─┤                  │
   │                                        └postfix┘                  │
   └─ -dqdt ─ pattern_delimiter ────────────────────────────────────  ┘
                    └─ -qi ─┘ └─ -dqdp ─┬prefix─┤
                                        └postfix┘
```
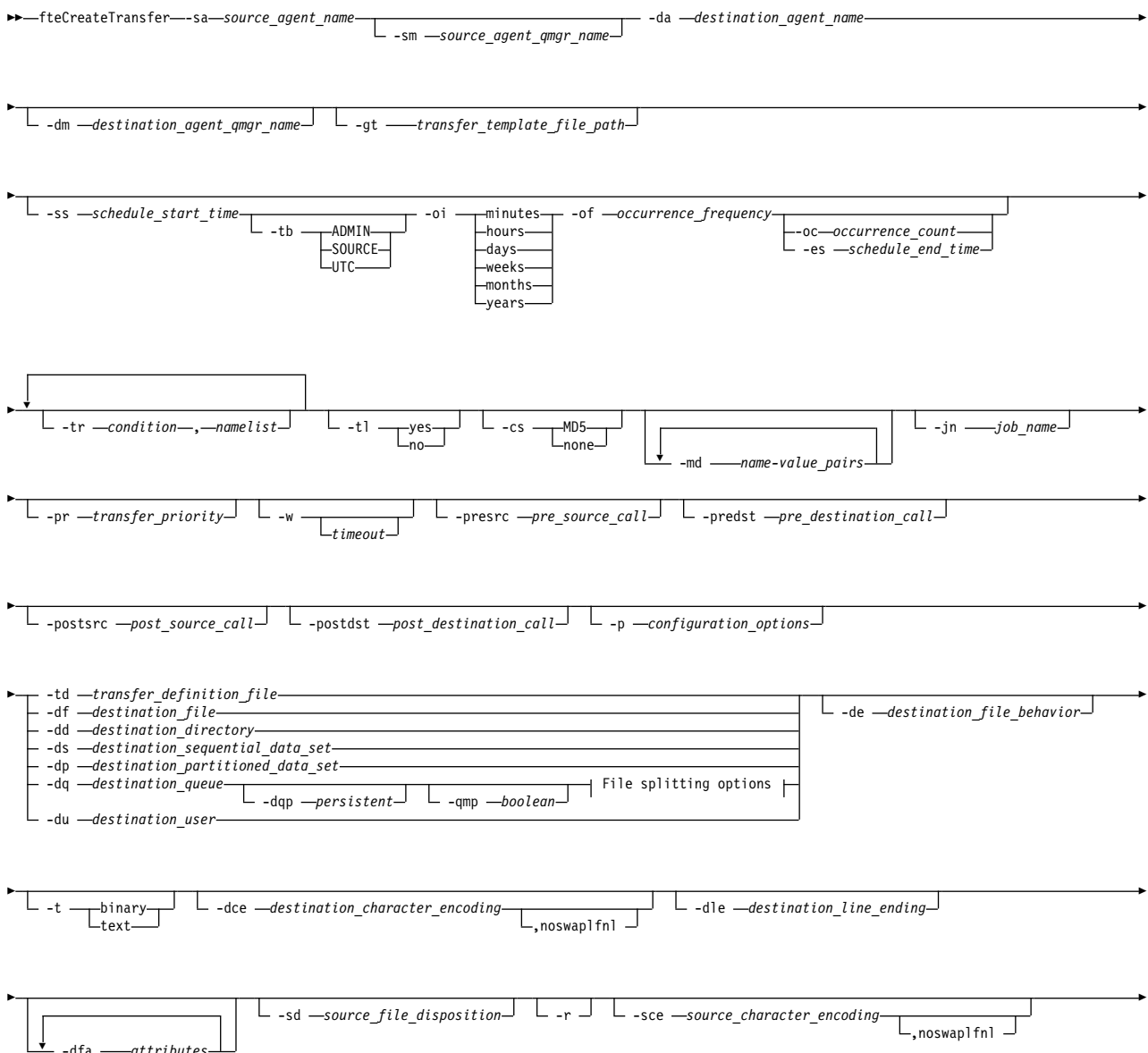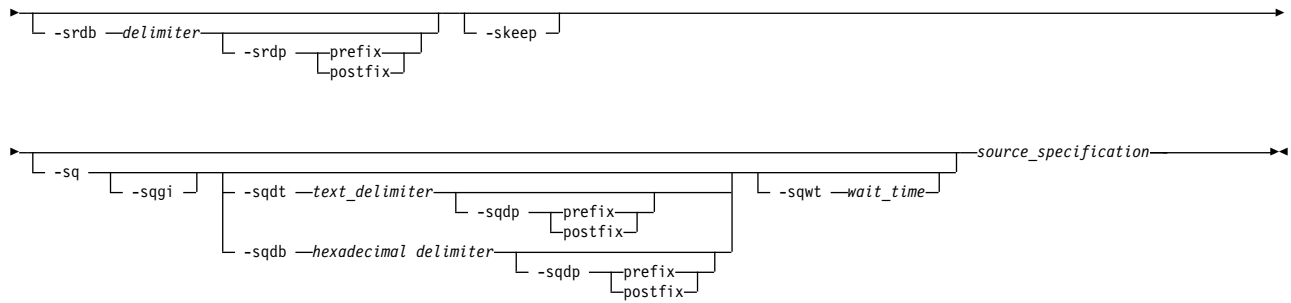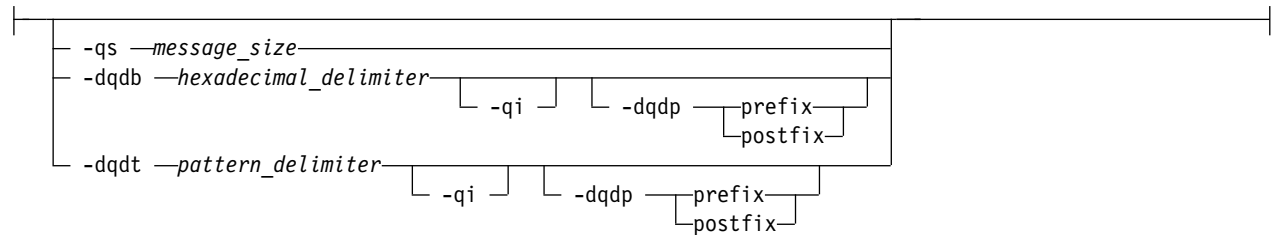
Parameters for agent specification

-sa *source_agent_name*
    Required. The name of the agent that the source files are transferred from.

    If you specify a protocol bridge agent as your source agent, you cannot then specify a data set as the source file specification.

    If you specify the **-td** parameter and the transfer definition file contains the source agent that you want to use for the transfer, do not specify the **-sa** parameter.

-sm *source_agent_qmgr_name*
    Optional. The name of the queue manager that the source agent is connected to.

    If you do not specify the **-sm** parameter, the queue manager that is used is determined by the set of configuration options in use, which is based on the source agent name. If the agent.properties file for the source agent cannot be found, the file transfer fails.

-da *destination_agent_name*
    Required. The name of the agent that the files are transferred to.

    If you specify the **-td** parameter and the transfer definition file contains the destination agent that you want to use for the transfer, do not specify the **-da** parameter.

-dm *destination_agent_qmgr_name*
    Optional. The name of the queue manager that the destination agent is connected to.

    If you do not specify the **-dm** parameter, the queue manager that is used is determined by the set of configuration options in use, which is based on the destination agent name. If the agent.properties file for the destination agent cannot be found, the file transfer fails.

Parameters for generating transfer templates

-gt *transfer_template_file_path*
    Optional. Generates a transfer template XML message and writes this message to a file. If you specify this parameter, no transfer request is sent to WebSphere MQ Managed File Transfer. Instead, the contents of the transfer request message are written to the named XML document. You can then use

this XML document to define the task for resource monitoring. See fteCreateMonitor command for information about how to create a resource monitor. If you do not specify this parameter, the default behavior takes place and an actual transfer request is carried out.

You must provide the full path and name of an XML output file as input for this parameter, for example `C:\templates\transfer_reports.xml`

On z/OS, you must store the transfer template document in a UNIX file on z/OS UNIX System Services. You cannot store transfer template documents in z/OS sequential files or PDS members.

The transfer template XML message that you create by using the **-gt** parameter is not the same as the transfer you create by using the `fteCreateTemplate` command, which means you cannot use the two different types of template interchangeably.

Parameters for scheduling transfers

-ss *schedule_start_time*
Optional. Specifies the time and date that you want the scheduled transfer to take place. Use one of the following formats to specify the time and date. Specify the time by using the 24-hour clock:

`yyyy-MM-ddThh:mm`

`hh:mm`

Scheduled file transfers start within a minute of the schedule start time, if there are no problems that might affect the transfer. For example, there might be issues with your network or agent that prevent the scheduled transfer starting.

-tb
Optional. Specifies the time base you want to use for the scheduled file transfer. That is, whether you want to use a system time or Coordinated Universal Time (UTC). You must use this parameter with the **-ss** parameter only. Specify one of the following options:

admin The start and end times used for the scheduled transfer are based on the time and date of the system used by the local administrator. This is the default value.

source The start and end times used for the scheduled transfer are based on the time and date of the system where the source agent is located.

UTC The start and end times used for the scheduled transfer are based on Coordinated Universal Time (UTC).

-oi
Optional. Specifies the interval that the scheduled transfer occurs at. You must use this parameter with the **-ss** parameter only. Specify one of the following options:

minutes

hours

days

weeks

months

years

-of *occurrence_frequency*
Optional. Specifies the frequency that the scheduled transfer occurs at. For example, every **5** weeks or every **2** months. You must specify this parameter with the **-oi** and **-ss** parameters only. If you do not specify this parameter, a default value of 1 is used.

-oc *occurrence_count*
Optional. Specifies how many times you want this scheduled transfer to occur. After the occurrence count is met, the scheduled transfer is deleted.

Specify this parameter with the **-oi** and **-ss** parameters only.

If you specify the **-oc** parameter, you cannot specify the **-es** parameter because these parameters are mutually exclusive.

You can omit both the **-oc** and **-es** parameters to create a transfer that repeats indefinitely.

-es *schedule_end_time*
Optional. The time and date that a repeating scheduled transfer ends.

You must specify this parameter with the **-oi** and **-ss** parameters only.

If you specify the **-es** parameter, you cannot specify the **-oc** parameter because these parameters are mutually exclusive.

You can omit both the **-es** and **-oc** parameters to create a transfer that repeats indefinitely.

Use one of the following formats to specify the end time and date. Specify the time by using the 24-hour clock:

`yyyy-MM-ddThh:mm`

`hh:mm`

Parameters for triggering transfers

-tr
Optional. Specifies a condition that must be true for this file transfer to take place. If the condition is not true, according to the source agent, the file transfer is discarded and no transfer takes place. Specify the following format:

`condition,namelist`

where *condition* is one of the following values:

file=exist
A minimum of one of the files in the namelist exists. That is, if *any* of the files in the namelist exists, the condition is true.

file!=exist
A minimum of one of the files in the namelist does not exist. That is, if *any* of the files in the namelist do not exist, the condition is true.

filesize>=*size*
A minimum of one of the files in the namelist exists and has a minimum size as specified by *size*. *size* is an integer with an optional size unit of KB, MB, or GB. For example, `filesize">"=10KB`. If you do not specify a size unit, the size is assumed to be bytes. On all operating systems, you must enclose the greater than symbol (>) in double quotation marks when you specify the `filesize` option on the command line, as shown in this example.

And where *namelist* is a comma-separated list of file names located on the same system as the source agent. Depending on your operating system, if you want to use path names or file names in a namelist that contain spaces, you might have to enclose the path names and file names in double quotation marks. You can specify more than one trigger condition by using the **-tr** parameter more than once. However in that case, every separate trigger condition must be true for the file transfer to take place.

Note: To continually monitor a resource for a trigger condition to be true, you are strongly recommended to use resource monitoring. You can create a resource monitor by using the fteCreateMonitor command.

In the following example, the file `file1.doc` is transferred from AGENT1 to AGENT2, on condition that either file `A.txt`, or file `B.txt`, or both files exist on AGENT1 *and* that either file `A.txt`, or file `B.txt`, or both files are equal to or larger than 1 GB:

```
fteCreateTransfer -sa AGENT1 -sm QM_JUPITER -da AGENT2 -dm QM_NEPTUNE
-tr file=exist,C:\export\A.txt,C:\export\B.txt
-tr filesize">"=1GB,C:\export\A.txt,C:\export\B.txt
-df C:\import\file1.doc C:\export\file1.doc
```

You can combine triggering parameters with scheduling parameters. If you do specify both types of parameters, the trigger conditions are applied to the file transfer created by the scheduling parameters.

The **-tr** parameter is not supported on protocol bridge agents.

-tl

Optional. Specifies whether trigger failures are written to the transfer log. Specify one of the following options:

yes   Transfer log entries are created for failed triggered transfers. This is the default behavior even if you do not specify the **-tl** parameter.

no   No transfer log entries are created for failed triggered transfers.

Parameters for specifying transfer options

-jn *job_name*

Optional. A user-defined job name identifier that is added to the transfer log message when the transfer starts.

-md

Optional. Specifies the user-defined metadata that is passed to the exit points run by the agent. The **-md** parameter can take one or more name-value pairs that are separated by commas. Each name pair consists of *name=value*. You can use the **-md** parameter more than once in a command.

▶ **V7.5.0.1** ◀ When the agent property enableUserMetadataOptions is set to a value of true, certain user-defined metadata keys provide more options to the transfer. For more information about the user-defined metadata keys that are currently supported, see "Supported user-defined metadata keys" on page 566. When the enableUserMetadataOptions property is set to true, key names starting with com.ibm.wmqfte. are not supported for user-defined use.

-cs

Optional. Specifies whether a checksum algorithm is run on the file transfer data to check the integrity of the transferred files. Specify one of the following options:

MD5   Computes an MD5 checksum for the data. The resulting checksum for the source and destination files is written to the transfer log for validation purposes. By default, WebSphere MQ Managed File Transfer computes MD5 checksums for all file transfers.

none No MD5 checksum is computed for the file transfer data. The transfer log records that checksum was set to none and the value for the checksum is blank. For example:

```
<checksum method="none"></checksum>
```

If you use the none option, you might improve file transfer performance, depending on your environment. However, selecting this option means that there is no validation of the source or destination files.

If you specify the **-cs** parameter, you cannot specify the **-td** parameter because these parameters are mutually exclusive. However, you can specify checksum behavior in the transfer definition file.

-pr *transfer_priority*

Optional. Specifies the priority level of the transfer. Priority is a value in the range 0-9, where 0 is the lowest priority. The default priority level is the priority level of the source agent.

This value matches the message priority value of WebSphere MQ, see Getting messages from a queue: priority for more information. Message traffic for file transfer data defaults to a priority level of 0, which allows your WebSphere MQ message traffic to take priority.

-qmp *boolean*

Optional. Specifies whether the first message written to the destination queue by the transfer has WebSphere MQ message properties set. The valid options are as follows:

true   Sets message properties on the first message that is created by the transfer.

false   Does not set message properties on the first message that is created by the transfer. This is the default value.

You can specify the **-qmp** parameter only if you also specify the **-dq** parameter. For more information, see "WebSphere MQ message properties set on messages written to destination queues" on page 701

-qs *message_size*

Optional. Specifies whether to split the file into multiple fixed-length messages. All the messages have the same WebSphere MQ group ID; the last message in the group has the WebSphere MQ LAST_MSG_IN_GROUP flag set. The size of the messages is specified by the value of *message_size*. The format of *message_size* is *<length><units>*, where *length* is a positive integer value and *units* is one of the following values:

B   Bytes. The minimum value that is allowed is two times the maximum bytes-per-character value of the code page of the destination messages.

K   This is equivalent to 1024 bytes.

M   This is equivalent to 1048576 bytes.

If the file is transferred in text mode, and is in a double-byte character set or multibyte character set, the file is split into messages on the closest character boundary to the specified message size.

You can specify the **-qs** parameter only if you also specify the **-dq** parameter. You can specify only one of the **-qs**, **-dqdb**, and **-dqdt** parameters.

-qi

Optional. Specifies whether to include the delimiter that is used to split the file into multiple messages in the messages. The delimiter is included at the beginning or at the end of the message, depending on the **-dqdp** parameter (which specifies prefix or postfix). By default the delimiter is not included in the messages.

You can specify the **-qi** parameter only if you also specify one of the **-dqdt** and **-dqdb** parameters.

-p *configuration_options*
Optional. This parameter determines the set of configuration options that is used to create the file transfer. Use the name of a non-default coordination queue manager as the input for this parameter. The command then uses the set of properties files that are associated with this non-default coordination queue manager.

If you do not specify this parameter, the set of configuration options that are based on the default coordination queue manager is used.

-w *timeout*
Optional. Specifying the **-w** parameter causes the **fteCreateTransfer** command to wait for the requested transfer to complete before returning. If you do not specify this parameter, the **fteCreateTransfer** command waits a maximum of five seconds to receive an acknowledgment from the source agent for the transfer that the agent has received the transfer request. If no acknowledgment is received during the five-second wait, the **fteCreateTransfer** command returns the following warning message:

`BFGCL0253W: No acknowledgment to command from agent within timeout.`

The *timeout* argument is optional. If you specify *timeout*, the **fteCreateTransfer** command waits for up to *timeout* seconds for the agent to respond. If the agent does not respond before the time limit is reached, the command produces a warning and ends with a return code of 2. If you do not specify a *timeout* value, or you specify a *timeout* value of -1, then the command waits until the agent responds.

Parameters for invoking programs

For more information about how you can start a program from WebSphere MQ Managed File Transfer, see "Specifying programs to run" on page 277. For examples of specifying a program to invoke using the parameters that are described here, see "Examples of using fteCreateTransfer to start programs" on page 864.

-presrc *pre_source_call*

Optional. Specifies a program to invoke at the source agent before the transfer starts. Use the following format for *pre_source_call*:

`[type:]commandspec[,[retrycount][,[retrywait][,successrc]]]`

In this syntax, the variables are:

type
> Optional. Valid values are **executable**, **antscript**, **jcl**, and **os4690background**. The default value is **executable**.
>
> `V7.5.0.1` The **jcl** value is only applicable when targeted at an agent in a z/OS environment.
>
> The **antscript** value is not applicable when targeted at an agent in an IBM 4690 environment.

commandspec
> Required. The command specification. Use one of the following formats:
> - Type **executable**: *command*[(arg1,arg2,...)]
> - Type **antscript**: *command*[(name1=var1|target1,name2=var2|target2,...)]
> - Type **jcl**: *command*
> - Type **os4690background**: *command*[(arg1,arg2,...)]
>
> where:
>
> *command*
> > Required. The name of the program to call.
>
> `V7.5.0.1` The **jcl** value is only applicable when targeted at an agent in a z/OS environment.
>
> The **antscript** value is not applicable when targeted at an agent in an IBM 4690 environment.
>
> Arguments in brackets ([ ]) are optional and syntax depends on command type. Parentheses, commas (,), and backslash (\) characters that are within the command or parameters must be escaped with a back slash (\) character.

retrycount
> Optional. The number of times to retry calling the program if the program does not return a successful return code. Default value is 0.

retrywait
> Optional. The time to wait, in seconds, before trying the program invocation again. Default value is 0 (no wait between retries).

successrc
> Optional. Expression that is used to determine when the program invocation successfully runs. This expression can be composed of one or more expressions. Combine these expressions with a

vertical bar character (|) to represent Boolean OR, or an ampersand (&) character to represent Boolean AND. Each expression is of the following form:

[>|<|!]*value*

where

> Optional. A greater than test of the *value*.

< Optional. A less than test of the *value*.

! Optional. A not equal to test of the *value*.

value
 Required. A valid integer.

priority
 Optional (os4690background only). The priority level to assign to a background task on an IBM 4690 system. Default value is 5 and valid values are within the range 1 - 9.

message
 Optional (os4690background only). The status message to display on an IBM 4690 system background control screen for the executed command.

-predst *pre_destination_call*
 Optional. Specifies a program to invoke at the destination agent before the transfer starts. *pre_destination_call* has the same format as *pre_source_call*.

-postsrc *post_source_call*

 Optional. Specifies a program to invoke at the source agent after the transfer has completed. *post_source_call* has the same format as *pre_source_call*.

-postdst *post_destination_call*
 Optional. Specifies a program to invoke at the destination agent after the transfer has completed. *post_destination_call* has the same format as *pre_source_call*.

Parameters for specifying the destination

One of the **-td**, **-df**, **-dd**, **-ds**, **-dq**, **-du**, and **-dp** parameters is required. You cannot specify more than one of these parameters in a transfer request; they are mutually exclusive.

-td *transfer_definition_file*
 Optional. The name of the XML document that defines one or more source and destination file specifications for the transfer. Alternatively, the name of the XML document that contains a managed transfer request (which might have been generated by the **-gt** parameter). If you specify the **-td** parameter and also specify any other parameters on the command line, these other parameters override the corresponding value from the transfer definition file.

 The **fteCreateTransfer** command locates the transfer definition file in relation to your current directory. If you cannot use relative path notation to specify the location of the transfer definition file, use the fully qualified path and file name of the transfer definition file instead.

 On z/OS, you must store the transfer definition file in a UNIX file on z/OS UNIX System Services. You cannot store transfer definition files in z/OS sequential files or PDS members.

 For more information, see Using transfer definition files.

-df *destination_file*

 Optional. The name of the destination file.

 If the destination agent is a Connect:Direct bridge agent, the destination file is specified in the format *connect_direct_node_name:file_path*. The Connect:Direct bridge agent accepts only file paths that are specified in this format. If the destination agent is a Connect:Direct bridge agent and the destination is a PDS member, you must also specify the **-de** parameter with a value of overwrite.

Note the following information:
- If the destination agent is a protocol bridge agent and you want to specify an endpoint for a file, use the following format:

  protocol_server:file_path

  where *protocol_server* is the name of the protocol server (which is optional) and where *file_path* is the path to the file on the protocol server system. If you do not specify a protocol server, the default protocol server is used.
- If you want to invoke any of the WebSphere MQ Managed File Transfer transfer I/O user exits that you have defined against the destination agent, you can use the **-df** parameter in a transfer.
- When the destination agent is on z/OS, if the file specified starts with //, it is assumed to be a partitioned z/OS data set.

-dd *destination_directory*

Optional. The name of the directory the file is transferred to. Specify a valid directory name on the system where the destination agent is running.

If the destination agent is a Connect:Direct bridge agent, the destination directory is specified in the format *connect_direct_node_name:directory_path*. If the destination agent is a Connect:Direct bridge agent and the destination is a PDS, you must also specify the **-de** parameter with a value of overwrite.

Note the following information:
- If the destination agent is a protocol bridge agent and you want to specify a directory at a particular endpoint, use the following format:

  protocol_server:directory_path

  where *protocol_server* is the name of the protocol server (which is optional) and where *directory_path* is the path to the directory on the protocol server system. If you do not specify a protocol server, the default protocol server is used.
- If you want to invoke any of the WebSphere MQ Managed File Transfer transfer I/O user exits that you have defined against the destination agent, you can use the **-dd** parameter in a transfer.
- When the destination agent is on z/OS, if the file specified starts with //, it is assumed to be a z/OS partitioned data set.

-ds *destination_sequential_data_set*

z/OS only. Optional. The name of the sequential data set or PDS member that files are transferred into. Specify a sequential data set name or a partitioned data set member. For information about transferring data sets, see "Guidelines for transferring files" on page 668.

The syntax for the data set name is as follows:

 //data_set_name{;*attribute*(*value*);..;*attribute*(*value*)}

or

 //pds_data_set_name(*member_name*){;*attribute*(*value*);..;*attribute*(*value*)}

That is, a data set name specifier prefixed with // and optionally followed by a number of attributes that are separated by semicolons.

For example:
`//'TEST.FILE.NAME';DSNTYPE(PDS);RECFM(F,B);BLKSIZE(800);LRECL(80);CYL;SPACE(2,2)`

If the data set is located at a Connect:Direct node, you must prefix the data set name with the node name. For example:
`CD_NODE1://'OBJECT.LIB';RECFM(F,B);BLKSIZE(800);LRECL(80)`

If the destination agent is a Connect:Direct bridge agent and the destination is a PDS member, you must also specify the **-de** parameter with a value of overwrite. For more information about data set transfers to or from Connect:Direct nodes, see "Transferring data sets to and from Connect:Direct nodes" on page 673.

For transfers that only involve WebSphere MQ Managed File Transfer agents, if the data set name part is enclosed by single quotation mark characters, it specifies a fully qualified data set name. If the data set name is not enclosed by single quotation mark characters, the system adds the default high-level qualifier for the destination agent (either the value for the transferRootHLQ agent property or the user ID that the agent runs under, if you have not set transferRootHLQ).

Note: However, for transfers that involve a Connect:Direct node on a z/OS system, the data set specification is interpreted as a fully qualified name. No high-level qualifier is added to the data set name. This is the case even if the data set name is enclosed by single quotation mark characters.

When you transfer a file or data set to tape, any existing data set that is already on the tape is replaced. The attributes for the new data set are set from attributes that are passed in the transfer definition. If no attributes are specified, attributes are set to the same as the source data set or to the default values when the source is a file. The attributes of an existing tape data set are ignored.

The data set attributes are used either to create a data set or to ensure that an existing data set is compatible. The specification of data set attributes is in a form suitable for BPXWDYN (see Requesting dynamic allocation for more information). When the agent is to create a destination data set, the following BPXWDYN attributes are automatically specified: DSN(*data_set_name*) NEW CATALOG MSG(*numeric_file_descriptor*). The value of *numeric_file_descriptor* is generated by WebSphere MQ Managed File Transfer. For a data set to data set transfer, the attributes of RECFM, LRECL, and BLKSIZE from the source are selected for a new destination data set. The SPACE setting for a new destination data set is not set by WebSphere MQ Managed File Transfer and system defaults are used. Therefore, you are recommended to specify the SPACE attribute when a new data set is to be created. You can use the **bpxwdynAllocAdditionalProperties** property in the `agent.properties` file to set BPXWDYN options that apply to all transfers. For more information, see "The agent.properties file" on page 548.

Some BPXWDYN options must not be specified when using the **fteCreateTemplate** command, the **fteCreateTransfer** command or the **bpxwdynAllocAdditionalProperties** property in the `agent.properties` file. For a list of these properties, see "BPXWDYN properties you must not use with WebSphere MQ Managed File Transfer" on page 679.

The **-ds** parameter is not supported when the destination agent is a protocol bridge agent.

If you want to invoke any of the WebSphere MQ Managed File Transfer transfer I/O user exits that you have defined against an agent, do not specify the **-ds** parameter in a transfer. Using the **-ds** parameter prevents the transfer I/O user exits from being invoked for the destination and means that the standard WebSphere MQ Managed File Transfer I/O is used instead.

-dp *destination_partitioned_data_set*
  z/OS only. Optional. The name of the destination PDS that files are transferred into. Specify a partitioned data set name. If a PDS is created as a result of the transfer, this PDS is created as a PDSE by default. You can override the default by specifying DSNTYPE=PDS.

The syntax for the PDS data set name is as follows:

//pds_data_set_name{;*attribute*;..;*attribute*}

The syntax for the data set name is the same as described for the **-ds** (*destination_sequential_data_set*) parameter. All the syntax details for specifying data sets that are located on Connect:Direct nodes also apply to the **-dp** parameter. If the destination agent is a Connect:Direct bridge agent, you must also specify the **-de** parameter with a value of overwrite.

The **-dp** parameter is not supported when the destination agent is a protocol bridge agent.

If you want to invoke any of the WebSphere MQ Managed File Transfer transfer I/O user exits that you have defined against an agent, do not specify the **-dp** parameter in a transfer. Using the **-dp** parameter prevents the transfer I/O user exits from being invoked for the destination and means that the standard WebSphere MQ Managed File Transfer I/O is used instead.

-du *destination_user*

Optional. The name of the user whose destination file space the files are transferred into. For more information about file spaces, see "File spaces" on page 313.

The **-du** parameter is not supported when the destination agent is a protocol bridge agent or a Connect:Direct bridge agent.

If you want to invoke any of the WebSphere MQ Managed File Transfer transfer I/O user exits that you have defined against an agent, do not specify the **-du** parameter in a transfer. Using the **-du** parameter prevents the transfer I/O user exits from being invoked for the destination and means that the standard WebSphere MQ Managed File Transfer I/O is used instead.

-dq *destination_queue*

Optional. The name of a destination queue that files are transferred onto. You can optionally include a queue manager name in this specification, by using the format QUEUE@QUEUEMANAGER. If you do not specify a queue manager name the destination agent queue manager name is used. You must specify a valid queue name that exists on the queue manager.

The **-dq** parameter is not supported when the destination agent is a protocol bridge agent or a Connect:Direct bridge agent, or when the source specification is a queue.

If you want to invoke any of the WebSphere MQ Managed File Transfer transfer I/O user exits that you have defined against an agent, do not specify the **-dq** parameter in a transfer. Using the **-dq** parameter prevents the transfer I/O user exits from being invoked for the destination and means that the standard WebSphere MQ Managed File Transfer I/O is used instead.

-dqp *persistent*

Optional. Specifies whether messages written to the destination queue are persistent. The valid options are as follows:

true   Writes persistent messages to the destination queue. This is the default value.

false   Writes non-persistent messages to the destination queue.

qdef   The persistence value is taken from the DefPersistence attribute of the destination queue.

You can specify the **-dqp** parameter only if you also specify the **-dq** parameter.

-dqdb *hexadecimal_delimiter*

Optional. Specifies the hexadecimal delimiter to use when splitting a binary file into multiple messages. All the messages have the same WebSphere MQ group ID; the last message in the group has the WebSphere MQ LAST_MSG_IN_GROUP flag set. The format for specifying a hexadecimal byte as a delimiter is xNN, where N is a character in the range 0-9 or a-f. You can specify a sequence of hexadecimal bytes as a delimiter by specifying a comma-separated list of hexadecimal bytes, for example: x3e,x20,x20,xbf.

You can specify the **-dqdb** parameter only if you also specify the **-dq** parameter and the transfer is in binary mode. You can specify only one of the **-qs**, **-dqdb**, and **-dqdt** parameters.

-dqdt *pattern*

Optional. Specifies the Java regular expression to use when splitting a text file into multiple messages. All the messages have the same WebSphere MQ group ID; the last message in the group has the WebSphere MQ LAST_MSG_IN_GROUP flag set. The format for specifying a regular expression as a delimiter is a regular expression that is enclosed in parentheses,

(*regular_expression*), or enclosed in double quotation marks, "*regular_expression*". For more information, see "Regular expressions used by WebSphere MQ Managed File Transfer" on page 685.

By default, the length of the string that the regular expression can match is limited by the destination agent to five characters. You can change this behavior by editing the **maxDelimiterMatchLength** agent property. For more information, see "Advanced agent properties" on page 549.

You can specify the **-dqdt** parameter only if you also specify the **-dq** parameter and the value text for the **-t** parameter. You can specify only one of the **-qs**, **-dqdb**, and **-dqdt** parameters.

-dqdp *position*

Optional. Specifies the expected position of destination text and binary delimiters when splitting files. You can specify the **-dqdp** parameter only if you also specify one of the **-dqdt** and **-dqdb** parameters.

Specify one of the following options:

prefix The delimiters are expected at the beginning of each line.

postfix
The delimiters are expected at the end of each line. This is the default option.

-de *destination_file_behavior*
Optional. Specifies the action that is taken if a destination file exists on the destination system. The valid options are as follows:

error Reports an error and the file is not transferred. This is the default value.

overwrite
Overwrites the existing destination file.

If you specify the **-de** parameter, you cannot specify the **-td** parameter because these parameters are mutually exclusive. However, you can specify destination file exists behavior in the transfer definition file.

-t Optional. Specifies the type of file transfer: binary mode or text mode.

binary The data in the file is transferred without any conversion. This is the default value.

text The code page and end-of-line characters of the file are converted. You can specify which code page and line ending to use for the conversion with the **-sce**, **-dce** or **-dle** parameters. If you do not specify the **-sce**, **-dce** or **-dle** parameters, the exact conversions performed depend on the operating system of the source agent and destination agent.

For example, a file that is transferred from Windows to z/OS has its code page converted from ASCII to EBCDIC. When a file is converted from ASCII to EBCDIC, the end-of-line characters are converted from ASCII carriage return (CR) and line feed (LF) character pairs to an EBCDIC new line (NL) character.

For more information about how z/OS data sets are transferred, see Transferring files and data sets between z/OS and distributed systems.

If you specify the **-t** parameter, you cannot specify the **-td** parameter because these parameters are mutually exclusive. However, you can specify transfer mode behavior in the transfer definition file.

-dce *destination_character_encoding*
Optional. Specifies which character encoding to use to write the file at the destination. This option is only applicable to text files and so **-t text** must also be specified. The code pages available for conversion depend on the platform of the destination agent. For a list of available code pages, see the topic "Available code pages" on page 707.

noswaplfnl
By default WebSphere MQ Managed File Transfer uses swaplfnl with supported EBCDIC character sets. Using swaplfnl changes the behavior of the character set mapping from and to

the EBCIDIC LF 0x25 character. However, this can sometimes result in a mapping that is not desired. Use noswaplfnl to override this behavior.

-dle *destination_line_ending*

Optional. Specifies the end-of-line characters that are used when the file is written at the destination. This option is applicable to text files only and so you must also specify the **-t text** parameter. The valid options are:

LF   Line feed. This is the default for UNIX platforms and for z/OS UNIX System Services files. When you use the standard EBCDIC code pages that are supplied with WebSphere MQ Managed File Transfer for EBCDIC files, the end-of-line characters are mapped to a NL character (0x15) and not to a LF character (0x25).

CRLF Carriage return followed by line feed. This is the default for Microsoft Windows.

If the destination of the transfer is a z/OS data set, this option is ignored.

-dfa *attributes*

Optional. Specifies a semicolon separated list of file attributes that are associated with the destination files in the transfer. The **-dfa** parameter can be specified with or without a value. For example, without a value:

`-dfa ATTRIBUTE1;ATTRIBUTE2`

For example, with a value:

`-dfa ATTRIBUTE1(VALUE);ATTRIBUTE2(VALUE)`

For example, one attribute with a value and one without:

`-dfa ATTRIBUTE1;ATTRIBUTE2(VALUE)`

You can use the **-dfa** parameter more than once in a command.

For more information about file attributes, see "File distribution attributes" on page 87.

Parameters for specifying the source

-sd *source_file_disposition*

Optional. Specifies the action that is taken on a source file when that source file is successfully transferred to its destination. The valid options are as follows:

leave   The source files are left unchanged. This is the default value.

delete The source files are deleted from the source system after the source files are successfully transferred.

On z/OS, if the source is a tape data set and you specify the `delete` option, the tape is remounted to delete the data set. This behavior is because of the behavior of the system environment.

If the source is a queue and you specify the `leave` option, the command returns an error and a transfer is not requested.

If the source agent is a Connect:Direct bridge agent and you specify the `delete` option, the behavior is different to the usual source disposition behavior. One of the following cases occurs:

• If Connect:Direct uses a process that is generated by WebSphere MQ Managed File Transfer to move the file or data set from the source, specifying the `delete` option causes the transfer to fail. To specify that the source file is deleted, you must submit a user-defined Connect:Direct process. For more information, see "Submitting a user-defined Connect:Direct process from a file transfer request" on page 269.

• If Connect:Direct uses a user-defined process to move the file or data set from the source, this parameter is passed to the process through the **%FTEFDISP** intrinsic symbolic variable. The

user-defined process determines whether the source is deleted. The result that the transfer returns depends on the result that is returned by the user-defined process.

If you specify the **-sd** parameter, you cannot specify the **-td** parameter because these parameters are mutually exclusive. However, you can specify source disposition behavior in the transfer definition file.

-r

Optional. Recursively transfer files in subdirectories when *source_specification* contains wildcard characters. When WebSphere MQ Managed File Transfer is presented with a wildcard character as a *source_specification*, any subdirectories that match the wildcard character are transferred only if you specify the **-r** parameter. When *source_specification* matches a subdirectory, all files in that directory and its subdirectories (including hidden files) are always transferred.

For more information about how WebSphere MQ Managed File Transfer handles wildcard characters, see Using wildcard characters

If you specify the **-r** parameter, you cannot specify the **-td** parameter because these parameters are mutually exclusive. However, you can specify recursive behavior in the transfer definition file.

-sce *source_character_encoding*

Optional. Specifies which character encoding to use to read the source file when performing character conversion. This option is only applicable to text files and so **-t text** must also be specified. The code pages available for conversion depend on the platform of the destination agent, because the conversion is performed on the destination system. For a list of available code pages, see the topic "Available code pages" on page 707.

noswaplfnl

By default WebSphere MQ Managed File Transfer uses swaplfnl with supported EBCDIC character sets. Using swaplfnl changes the behavior of the character set mapping from and to the EBCIDIC LF 0x25 character. However, this can sometimes result in a mapping that is not desired. Use noswaplfnl to override this behavior.

-skeep

Optional. Specifies that trailing spaces are kept on source records that are read from a fixed-length-format record-oriented file (for example, a z/OS data set) as part of a text mode transfer. If you do not specify this parameter, trailing spaces are stripped from source records.

-srdb *delimiter*

Optional. For source files that are record oriented (for example, z/OS data sets), specifies one or more byte values to insert as the delimiter when appending records into a binary file. You must specify each value as two hexadecimal digits in the range 00-FF, prefixed by x. Separate multiple bytes with commas. For example:

```
-srdb x0A
```

or

```
-srdb x0D,x0A
```

You must configure the transfer in binary mode.

-srdp *position*

Optional. Specifies the position to insert source record delimiters. You can specify the **-srdp** parameter only if you also specify the **-srdb** parameter.

Specify one of the following options:

prefix The delimiters are inserted at the start of each record.

postfix

The delimiters are inserted at the end of each record. This is the default option.

-sq

Optional. Specifies that the source of a transfer is a queue.

If you want to invoke any of the WebSphere MQ Managed File Transfer transfer I/O user exits that you have defined against an agent, do not specify the **-sq** parameter in a transfer. Using the **-sq** parameter prevents the transfer I/O user exits from being invoked for the source and means that the standard WebSphere MQ Managed File Transfer I/O is used instead.

-sqgi

Optional. Specifies that the messages are grouped by WebSphere MQ group ID. The first complete group is written to the destination file. If this parameter is not specified, all messages on the source queue are written to the destination file.

You can specify the **-sqgi** parameter only if you also specify the **-sq** parameter.

-sqdt *text_delimiter*

Optional. Specifies a sequence of text to insert as the delimiter when appending multiple messages to a text file. You can include Java escape sequences for String literals in the delimiter. For example, -sqdt \u007d\n.

The text delimiter is encoded to binary format by using the source encoding of the transfer. Each message is read in binary format. The encoded delimiter is prepended or appended in binary format to the message (as specified by the **-sqdp** parameter) and the result is transferred in binary format to the destination agent. If the source agent code page includes shift-in and shift-out states, the agent assumes that each message is in the shift-out state at the end of the message. At the destination agent the binary data is converted in the same way as a file to file text transfer.

You can specify the **-sqdt** parameter only if you also specify the **-sq** parameter and the value text for the **-t** parameter.

-sqdb *hexadecimal_delimiter*

Optional. Specifies one or more byte values to insert as the delimiter when appending multiple messages to a binary file. Each value must be specified as two hexadecimal digits in the range 00-FF, prefixed by x. Multiple bytes must be comma-separated. For example, -sqdb x08,xA4.

You can specify the **-sqdb** parameter only if you also specify the **-sq** parameter. You cannot specify the **-sqdb** parameter if you also specify the value text for the **-t** parameter.

-sqdp *position*

Optional. Specifies the position of insertion of source text and binary delimiters. You can specify the **-sqdp** parameter only if you have also specified one of the **-sqdt** and **-sqdb** parameters.

Specify one of the following options:

prefix The delimiters are inserted at the start of each message

postfix
The delimiters are inserted at the end of each message. This is the default option.

-sqwt *wait_time*

Optional. Specifies the time, in seconds, to wait for one of the following conditions to be met:
- For a new message to appear on the queue
- If the **-sqgi** parameter was specified, for a complete group to appear on the queue

If neither of these conditions is met within the time that is specified by *wait_time*, the source agent stops reading from the queue and completes the transfer. If the **-sqwt** parameter is not specified, the source agent stops reading from the source queue immediately if the source queue is empty or, in the case where the **-sqgi** parameter is specified, if there is no complete group on the queue.

For information about using the **-sqwt** parameter, see "Guidance for specifying a wait time on a message-to-file transfer" on page 706.

You can specify the **-sqwt** parameter only if you also specify the **-sq** parameter.

source_specification

One or more file specifications that determine the source, or sources, for the file transfer.

Required if you specify one of the **-df**, **-dd**, **-dp**, **-dq**, **-du**, or **-ds** parameters. If you specify the **-td** parameter, do not specify *source_specification*.

- If you have not specified the **-sq** parameter, *source_specification* is one or more file specifications that determine the source, or sources, for the file transfer. File specifications can take one of five forms and can include wildcard characters. For more information about wildcard characters, see "Using wildcard characters" on page 683. You can escape asterisks that are part of the file specification by using two asterisk characters (**) in the file specification.

  You can specify multiple source file specifications separated by the space character. However, if you specify multiple source specifications for the **-df** or **-ds** parameters and also specify **-de overwrite**, the destination will contain only the data for the source file that you specified last. If you do not specify **-de overwrite** the transfer can only be partially successful. If the destination file did not previously exist, it will contain the data for the source file that you specified first.

  To transfer files that contain spaces in their file names, for example a b.txt to file c d.txt, place double quotation marks around the file names that contain spaces. Specify the following text as part of the **fteCreateTransfer** command:

  ```
  -df "c d.txt" "a b.txt"
  ```

  Each file specification must be in one of the following categories:

  File names

  The name of a file, expressed in the appropriate notation for the system where the source agent is running. When a file name is specified as a source file specification, the contents of the file are copied.

  Directories

  The name of a directory, expressed in the appropriate notation for the system where the source agent is running. When a directory is specified as a source file specification, the contents of the directory are copied. More precisely, all files in the directory and in all its subdirectories, including hidden files, are copied.

  For example, to copy the contents of DIR1 to DIR2 only, specify `fteCreateTransfer ... -dd DIR2 DIR1/*`

  Sequential data set

  (z/OS only). The name of a sequential data set or partitioned data set member. Denote data sets by preceding the data set name with two forward slash characters (//).

  If you specify a protocol bridge agent as your source agent, you cannot then specify a data set as the source file specification.

  Partitioned data set

  (z/OS only). The name of a partitioned data set. Denote data set names by preceding the data set name with two forward slash characters (//).

  If you specify a protocol bridge agent as your source agent, you cannot then specify a data set as the source file specification.

  File name or directory at a Connect:Direct node

  (Connect:Direct bridge agent only). The name of a Connect:Direct node, a colon character (:), and a file or directory path on the system that is hosting the Connect:Direct node. For example, *connect_direct_node_name:file_path*.

If the source agent is a Connect:Direct bridge agent, it will only accept source specifications in this form.

Note: Wildcard characters are not supported in file paths when the source agent is a Connect:Direct bridge agent.

File name or directory on a protocol file server

The name of a protocol file server, a colon character (:), and a file or directory path on the protocol server system. For example, *protocol_server*:*file_path*.

If you do not specify a protocol server, the default protocol server is used.

- If you specify the **-sq** parameter, *source_specification* is the name of a local queue on the source agent queue manager. You can specify only one source queue. The source queue is specified in the format:

  QUEUE_NAME

  The queue manager name is not included in the source queue specification, because the queue manager must be the same as the source agent queue manager.

- If the source agent is on z/OS, source files that start with // are assumed to be z/OS partitioned data sets.

Other parameters

-? or -h

Optional. Displays command syntax.

Examples

In this basic example, the file `originalfile.txt` is transferred from AGENT1 to AGENT2 on the same system and renamed to `transferredfile.txt`

```
fteCreateTransfer -sa AGENT1 -da AGENT2 -df C:\import\transferredfile.txt C:\export\originalfile.txt
```

In this example, the files `originalfile.txt` and `originalfile2.txt` are transferred from AGENT1 to AGENT2 on the same system, to the directory `C:\import`

```
fteCreateTransfer -sa AGENT1 -da AGENT2 -dd C:\import C:\export\originalfile.txt C:\export\originalfile2.txt
```

In this example, the file `originalfile.txt` is transferred from AGENT1's system to AGENT2's system. The file transfer is scheduled to take place at 09:00 based on the system time of the source agent's system and occurs every two hours four times:

```
fteCreateTransfer -sa AGENT1 -sm QM_JUPITER -da AGENT2 -dm QM_NEPTUNE
-tb source -ss 09:00 -oi hours -of 2 -oc 4
-df C:\import\transferredfile.txt C:\export\originalfile.txt
```

In this example, the file `originalfile.txt` is transferred from AGENT1 to AGENT2, on condition that the file `A.txt` exists on AGENT1:

```
fteCreateTransfer -sa AGENT1 -sm QM_JUPITER -da AGENT2 -dm QM_NEPTUNE
-tr file=exist,C:\export\A.txt -df C:\import\transferredfile.txt C:\export\originalfile.txt
```

In this example, the file `originalfile.txt` is transferred from AGENT1's system to a data set //'USERID.TRANS.FILE.TXT' on AGENT2's system. Text mode is selected to convert data from ASCII to EBCDIC.

```
fteCreateTransfer -t text -sa AGENT1 -da AGENT2
-ds "//TRANS.FILE.TXT;RECFM(V,B);BLKSIZE(6144);LRECL(1028);
SPACE(5,1)" C:\export\originalfile.txt
```

In this example, a member of a fully qualified data set on AGENT1's system is transferred to a file on AGENT2's system. Text mode is selected to convert the file from EBCDIC to the default code page of AGENT2's system.

```
fteCreateTransfer -t text -sa AGENT1 -da AGENT2 -df /tmp/IEEUJV.txt "//'SYS1.SAMPLIB(IEEUJV)'"
```

In this example, a file that is called `file.bin` on agent AGENT1 is transferred to a destination file called `file.bin` on the protocol file server accountshost.ibm.com by using the destination agent BRIDGE1.

```
fteCreateTransfer -sa AGENT1 -da BRIDGE1 -df accountshost.ibm.com:/tmp/file.bin /tmp/file.bin
```

In this example, a wildcard is used without quotation marks. All files in AGENT1's current working directory that end in `.txt` are transferred to directory `C:\import` on AGENT2. The file names remain unchanged.

```
fteCreateTransfer -sa AGENT1 -da AGENT2 -dd C:\import  *.txt
```

In this example, a wildcard is used with double quotation marks. All files in AGENT1's transfer root directory that end in `.txt` are transferred to directory `C:\import` on AGENT2. The file names remain unchanged.

```
fteCreateTransfer -sa AGENT1 -da AGENT2 -dd C:\import  "*.txt"
```

In this example, an attribute is specified once by using a semicolon to concatenate values.

```
fteCreateTransfer –sa AGENT1 –da AGENT2 –dfa NAME1(VALUE1);NAME2(VALUE2) -dd c:\adx_test xyz.tx
```

In this example, an attribute is specified multiple times on the same command to allow the passing of more than one attribute.

```
fteCreateTransfer –sa AGENT1 –da AGENT2 –dfa NAME1(VALUE1) -dfa NAME2(VALUE2) -dd c:\adx_test xyz.tx
```

Return codes

Return code	Description
0	Command completed successfully.
1	Command ended unsuccessfully.
2	Command ended with a timeout. The command sent a message to the agent, but the agent did not respond within the time specified.
20	Command completed with partial success and some files were transferred.
21	The queue manager that the **fteCreateTransfer** command was connected to was stopped before the transfer result was determined.
40	Failed. None of the files specified were transferred.
41	The transfer was canceled.
42	The transfer did not take place because the transfer was conditional and the required condition was not met.
43	The transfer request message was malformed.
44	The source agent did not have sufficient capacity to carry out the transfer.
45	The destination agent did not have sufficient capacity to carry out the transfer.
46	The number of files that are being transferred exceeded the source agent's limit.
47	The number of files transferred exceeds the destination agent's limit.

fteCreateWebAgent (create a WebSphere MQ Managed File Transfer web agent)

The **fteCreateWebAgent** command creates an agent and its associated configuration for use with the Web Gateway. This command is provided with WebSphere MQ Managed File Transfer Server.

Purpose

Use the **fteCreateWebAgent** command to create a web agent. This command provides you with the MQSC commands that you must run on the queue manager that is used by the agent to create the following agent queues:

- SYSTEM.FTE.AUTHADM1.*agent_name*
- SYSTEM.FTE.AUTHAGT1.*agent_name*
- SYSTEM.FTE.AUTHMON1.*agent_name*
- SYSTEM.FTE.AUTHOPS1.*agent_name*
- SYSTEM.FTE.AUTHSCH1.*agent_name*
- SYSTEM.FTE.AUTHTRN1.*agent_name*
- SYSTEM.FTE.COMMAND.*agent_name*
- SYSTEM.FTE.DATA.*agent_name*
- SYSTEM.FTE.EVENT.*agent_name*
- SYSTEM.FTE.REPLY.*agent_name*
- SYSTEM.FTE.STATE.*agent_name*

Because the agent is for use with the Web Gateway, two queues are created in addition to the previous list:

- SYSTEM.FTE.WEB.RESP.*agent_name*
- SYSTEM.FTE.WEB.*gateway_name*

These queues are internal system queues that you must not modify, delete, or read messages from unless you are deleting the agent. The MQSC commands to run are also supplied in a file in the following location: *MQ_DATA_PATH*\mqft\config*coordination_qmgr_name*\agents*agent_name*\ *agent_name*_create.mqsc.

If you later want to delete the agent, this command also provides you with the MQSC commands you must run to clear then delete the queues belonging to the agent. The MQSC commands are in a file in the following location: *MQ_DATA_PATH*\mqft\config*coordination_qmgr_name*\agents*agent_name*\ *agent_name*_delete.mqsc.

WebSphere MQ Managed File Transfer provides advanced agent properties that help you configure agents. These properties are described in Properties files for WebSphere MQ File Transfer Edition.

Note: The user that your web agent runs as must be the same as, or in the same group as, the user that your application server runs as.

Limitations of the web agent

- A web agent can be only the source agent for transfers initiated by a Web Gateway. If you attempt to perform a transfer with a web agent as the source by another method, the transfer fails with return code 68 (TRANSFER_NOT_SUPPORTED).
- A web agent can only be the destination agent for a transfer when the destination is specified as a file space. If you attempt to perform a transfer with a web agent as the destination agent but a different destination type the transfer will fail with the following error message: BFGCH0103: The transfer request specifies Web Gateway agent '*web_agent_name*' as the destination agent. Web Gateway agents can be the destination only for a transfer to a file space.

- A web agent cannot monitor a resource. If you attempt to create a resource monitor for a web agent, the command fails with return code 113 (MONITOR_NOT_SUPPORTED).

Syntax

fteCreateWebAgent

►►──fteCreateWebAgent──-agentName────*agent_name*──-agentQMgr────*agent_qmgr_name*──-webGatewayName────*gateway_name*──────►

►──►
　　└─-agentQMgrHost────*agent_qmgr_host*─┘　└─-agentQMgrPort────*agent_qmgr_port*─┘

►──►
　　└─-agentQMgrChannel────*agent_qmgr_channel*─┘　└─-agentDesc────*agent_description*─┘　┌─-ac─────────────┐
　　└─-authorityChecking─┘

►──►
　　└─-p──*configuration_options*─┘　└─-f─┘

►──►◄
　　┌─-s──────────────────┬──-su────*user_name*──┐
　　│　　└─*service_name*─┘　　　　　　　　　　└─-sp────*password*─┘　└─-sj────*options*─┘　└─-sl────*options*─┘
　　└─-n──────────────────┘

Parameters

-agentName *agent_name*
Required. The name of the agent to create. The agent name must be unique to its coordination queue manager.

For more information about naming agents, see Object naming conventions .

-agentQMgr *agent_qmgr_name*
Required. The name of the agent queue manager.

-webGatewayName *gateway_name*
Required. The name of the Web Gateway that the agent is a component of.

For more information about naming Web Gateways, see Object naming conventions .

-agentQMgrHost *agent_qmgr_host*
Optional. The host name or IP address of the agent queue manager. If you do not specify this parameter, a bindings mode connection is assumed.

-agentQMgrPort *agent_qmgr_port*
Optional. The port number used for client connections to the agent queue manager. This parameter is used only if you have also specified the **agentQMgrHost** parameter. If you do not specify the **agentQMgrPort** parameter, a default port of 1414 is used.

-agentQMgrChannel *agent_qmgr_channel*
Optional. This parameter is used only if you have also specified the **agentQMgrHost** parameter. If you do not specify the **agentQMgrChannel** parameter, a default channel of SYSTEM.DEF.SVRCONN is used.

-agentDesc *agent_description*
Optional. A description of the agent, which is displayed in WebSphere MQ Explorer.

-ac or -authorityChecking
Optional. This parameter enables authority checking. If you specify this parameter, the agent checks that users who are submitting requests are authorized to perform the requested action.

-p *configuration_options*
Optional. The name of the set of configuration options that is used to create the agent. By convention, this is the name of a coordination queue manager. If you do not specify this parameter, the default set of configuration options is used.

-f   Optional. Forces the command to overwrite the existing configuration.

-s *service_name*
Optional (Windows only). Indicates that the agent is to run as a Windows service. If you do not specify *service_name*, the service is named mqmftAgent*<AGENT><QMGR>*, where *<AGENT>* is the agent name and *<QMGR>* is your agent queue manager name.

The display name for the service, which is shown in the Windows **Services** window in the **Name** column, is always **WebSphere MQ Managed File Transfer agent** *<AGENT>@<QMGR>*.

-su *user_name*
Optional (Windows only). When the agent is to run as a Windows service, this parameter specifies the name of the account under which the service should run. To run the agent using a Windows domain user account specify the value in the form DomainName\UserName. To run the service using an account from the local built-in domain specify the value in the form UserName.

The Windows user account that you specify using the **-su** parameter must have the **Log on as a service** right. For information about how to grant this right, see "Guidance for running an agent or logger as a Windows service" on page 366.

Required when **-s** specified. Equivalent to **-serviceUser**.

-sp *password*
Optional (Windows only). Password for the user account set by **-su** or **-serviceUser** parameter.

This parameter is only valid when **-s** is specified. Equivalent to **-servicePassword**. If you do not specify this parameter when you specify the **-s** parameter, a warning message is produced. This message warns you that you must set the password using the Windows Services tool before the service will start successfully.

-sj *options*
Optional (Windows only). When the agent is started as a Windows service, defines a list of options in the form of -D or -X that will be passed to the JVM. The options are separated using a number sign (#) or semicolon (;) character. If you need to embed any # or ; characters, put them inside single quotation marks.

This parameter is only valid when **-s** is specified. Equivalent to **-serviceJVMOptions**.

-sl *options*
Optional (Windows only). Sets the Windows service log level. Valid options are: error, info, warn, debug. The default is info. This option can be useful if you are having problems with the Windows service. Setting it to debug gives more detailed information in the service log file.

This parameter is only valid when **-s** is specified. Equivalent to **-serviceLogLevel**.

-n   Optional (Windows only). Indicates that the agent is to be run as a normal process. This is mutually exclusive with the **-s** option. If neither the **-s** nor the **-n** option is specified, then the agent is configured as a normal Windows process.

Equivalent to **-normal**.

-? or -h
Optional. Displays the command syntax.

Example

In this example, the agent WEBAGENT1 is created with an agent queue manager QM_NEPTUNE and the Web Gateway GATEWAY_ONE. The agent uses the default coordination queue manager:

```
fteCreateWebAgent -agentName WEBAGENT1 -webGatewayName GATEWAY_ONE -agentQMgr QM_NEPTUNE
                  -agentQMgrHost myhost.ibm.com -agentQMgrPort 1415 -agentQMgrChannel CHANNEL1
```

Return codes

0   Command completed successfully.

1　　Command ended unsuccessfully.

fteDefine (generate configuration scripts)

▶ V 7.5.0.2

Use the **fteDefine** command to generate the configuration scripts necessary to define the specified objects.

Purpose

You would expect to use the **fteDefine** command when some configuration steps need to be run on a system that is remote to the one containing the configuration data. For example, configuring the queues for an agent on a queue manager to be accessed over a client connection.

Syntax

fteDefine

```
▶▶──fteDefine──-t── type ──┬──────────────────┬── name ──────────────────────────────▶◀
                           └──-d── outputDir──┘
```

Parameters

-t *type*
  Required. The type of object to be defined. The options for type are `agent`.

-d *outputDir*
  Optional. A directory path in which the scripts are written. If not provided, the scripts are written to the standard output stream.

name
  Required. One or more names of the objects to be defined. To specify names for more than one object, separate them with a space. For example, *name1 name2...*

-? or -h
  Optional. Displays command syntax.

Examples

In this example, the **fteDefine** command is specified with the **-t agent** parameter and a single agent name.

```
fteDefine -t agent EXAMPLE.AGENT
```

The output that is generated from this command are the MQSC command scripts to be run against the agent queue manager to create the necessary agent queues:

```
$ fteDefine -t agent EXAMPLE.AGENT
5724-H72 Copyright IBM Corp.  2008, 2012.  ALL RIGHTS RESERVED
DEFINE QLOCAL(SYSTEM.FTE.COMMAND.EXAMPLE.AGENT) +
 DEFPRTY(0) +
 DEFSOPT(SHARED) +
 GET(ENABLED) +
 MAXDEPTH(5000) +
 MAXMSGL(4194304) +
 MSGDLVSQ(PRIORITY) +
 PUT(ENABLED) +
 RETINTVL(999999999) +
 SHARE +
 NOTRIGGER +
 USAGE(NORMAL) +
 REPLACE
```

```
DEFINE QLOCAL(SYSTEM.FTE.DATA.EXAMPLE.AGENT) +
 DEFPRTY(0) +
 DEFSOPT(SHARED) +
 GET(ENABLED) +
 MAXDEPTH(5000) +
 MAXMSGL(4194304) +
 MSGDLVSQ(PRIORITY) +
 PUT(ENABLED) +
 RETINTVL(999999999) +
 SHARE +
 NOTRIGGER +
 USAGE(NORMAL) +
 REPLACE
...
etc.
```

In this example, the **fteDefine** command is specified with the **-d outputDir** parameter and several agent names.

```
fteDefine -t agent -d /tmp EXAMPLE.AGENT.1 EXAMPLE.AGENT.2 EXAMPLE.AGENT.3
```

The output that is generated from this command are the absolute file paths to the locations of the MQSC command scripts:

```
$ fteDefine -t agent -d /tmp EXAMPLE.AGENT.1 EXAMPLE.AGENT.2 EXAMPLE.AGENT.3
5724-H72 Copyright IBM Corp.  2008, 2012.  ALL RIGHTS RESERVED
BFGCM0239I: A file has been created containing the MQSC definitions to define the agent EXAMPLE.AGENT.1. The file can be found here: '/tmp/EXAMP
BFGCM0239I: A file has been created containing the MQSC definitions to define the agent EXAMPLE.AGENT.2. The file can be found here: '/tmp/EXAMP
BFGCM0239I: A file has been created containing the MQSC definitions to define the agent EXAMPLE.AGENT.3. The file can be found here: '/tmp/EXAMP
```

Return codes

0   Command completed successfully.

1   Command ended unsuccessfully.

fteDelete (generate scripts to remove configuration)

▶ V 7.5.0.2

Use the **fteDelete** command to generate the configuration scripts necessary to remove the specified objects.

Purpose

You would expect to use the **fteDelete** command when some configuration steps need to be run on a system that is remote to the one containing the configuration data. For example, removing the queues for a remote client agent on a local queue manager.

Syntax

fteDelete

```
►►──fteDelete──-t── type ─────────────────────────── name ──────────────────────────────────────►◄
                             └─-d── outputDir─┘
```

Parameters

-t *type*
> Required. The type of object to be delete. The options for type are `agent`.

-d *outputDir*
> Optional. A directory path in which the scripts are written. If not provided, the scripts are written to the standard output stream.

name
> Required. One or more names of the objects to be delete. To specify names for more than one object, separate them with a space. For example, *name1 name2...*

-? or -h
> Optional. Displays command syntax.

Examples

In this example, the **fteDelete** command is specified with the **-t agent** parameter and a single agent name.

```
fteDelete -t agent EXAMPLE.AGENT
```

The output that is generated from this command are the MQSC command scripts to be run against the agent queue manager to delete the agent queues:

```
$ fteDelete -t agent EXAMPLE.AGENT
5724-H72 Copyright IBM Corp.  2008, 2012.  ALL RIGHTS RESERVED
CLEAR QLOCAL(SYSTEM.FTE.COMMAND.EXAMPLE.AGENT)
DELETE QLOCAL(SYSTEM.FTE.COMMAND.EXAMPLE.AGENT)
CLEAR QLOCAL(SYSTEM.FTE.DATA.EXAMPLE.AGENT)
DELETE QLOCAL(SYSTEM.FTE.DATA.EXAMPLE.AGENT)
CLEAR QLOCAL(SYSTEM.FTE.REPLY.EXAMPLE.AGENT)
DELETE QLOCAL(SYSTEM.FTE.REPLY.EXAMPLE.AGENT)
CLEAR QLOCAL(SYSTEM.FTE.STATE.EXAMPLE.AGENT)
DELETE QLOCAL(SYSTEM.FTE.STATE.EXAMPLE.AGENT)
CLEAR QLOCAL(SYSTEM.FTE.EVENT.EXAMPLE.AGENT)
DELETE QLOCAL(SYSTEM.FTE.EVENT.EXAMPLE.AGENT)
...
etc.
```

In this example, the **fteDelete** command is specified with the **-d outputDir** parameter and several agent names.

```
fteDelete -t agent -d /tmp EXAMPLE.AGENT.1 EXAMPLE.AGENT.2 EXAMPLE.AGENT.3
```

The output that is generated from this command are the absolute file paths to the locations of the MQSC command scripts:

```
$ fteDelete -t agent -d /tmp EXAMPLE.AGENT.1 EXAMPLE.AGENT.2 EXAMPLE.AGENT.3
5724-H72 Copyright IBM Corp.  2008, 2012.  ALL RIGHTS RESERVED
BFGCM0241I: A file has been created containing the MQSC definitions to delete the agent EXAMPLE.AGENT.1. The file can be found here: '/tmp/EX
BFGCM0241I: A file has been created containing the MQSC definitions to delete the agent EXAMPLE.AGENT.2. The file can be found here: '/tmp/EX
BFGCM0241I: A file has been created containing the MQSC definitions to delete the agent EXAMPLE.AGENT.3. The file can be found here: '/tmp/EX
```

Return codes

0   Command completed successfully.

1   Command ended unsuccessfully.

fteDeleteAgent (delete a WebSphere MQ Managed File Transfer agent)

The **fteDeleteAgent** command deletes a WebSphere MQ Managed File Transfer agent and its configuration. If the agent is protocol a bridge agent, the user credentials file is left on the file system.

Purpose

Stop the agent with the fteStopAgentcommand before running the **fteDeleteAgent** command.

If you have configured your agent to run as a Windows service, running the **fteDeleteAgent** command deletes the service definition. Only users who are WebSphere MQ administrators (and members of the mqm group) can run this command. If you try to run this command as a user who is not a WebSphere MQ administrator, you will receive an error message and the command will not run.

The **fteDeleteAgent** command provides you with the MQSC commands that you must run against your agent's queue manager to clear and delete the agent's system queues. These queues are as follows:

- SYSTEM.FTE.AUTHADM1.*agent_name*
- SYSTEM.FTE.AUTHAGT1.*agent_name*
- SYSTEM.FTE.AUTHMON1.*agent_name*
- SYSTEM.FTE.AUTHOPS1.*agent_name*
- SYSTEM.FTE.AUTHSCH1.*agent_name*
- SYSTEM.FTE.AUTHTRN1.*agent_name*
- SYSTEM.FTE.COMMAND.*agent_name*
- SYSTEM.FTE.DATA.*agent_name*
- SYSTEM.FTE.EVENT.*agent_name*
- SYSTEM.FTE.REPLY.*agent_name*
- SYSTEM.FTE.STATE.*agent_name*

If your agent is a web agent there are two additional queues that must be deleted. The **fteDeleteAgent** command clears and deletes the following queue:

- SYSTEM.FTE.WEB.RESP.*agent_name*

The **fteDeleteAgent** command does not delete the SYSTEM.FTE.WEB.<*gateway_name*> queue, because this queue is shared between multiple web agents. After running the **fteDeleteAgent** command for a web agent, you must manually delete the SYSTEM.FTE.WEB.*gateway_name* queue.

Note: Delete the SYSTEM.FTE.WEB.*gateway_name* queue only if all web agents associated with this Web Gateway have been deleted.

The **fteCreateAgent** command also provides these commands in a file in the following location:

MQ_DATA_PATH/mqft/config/*coordination_qmgr_name*/agents/*agent_name*/*agent_name*_delete.mqsc

Syntax

fteDeleteAgent

Parameters

-p (*configuration_ options***)**
Optional. If you have more than one coordination queue manager, use this parameter to explicitly specify which agent configuration you want to delete. Use the name of a non-default coordination queue manager as the input for this parameter. The command then uses the configuration options associated with this non-default coordination queue manager.

Specify the optional **-p** parameter only if you want to use configuration options different from your defaults. If you do not specify **-p**, the configuration options defined in the installation.properties file are used. See "Configuration options" on page 124 for more information.

-f Optional. Forces the command to deregister the agent from the coordination queue manager even if the agent's configuration files cannot be found. Because information about the agent's queue manager is not available in this situation, the command will connect directly to the coordination queue manager instead of using the agent queue manager as it normally would.

agent_name
Required. The name of the agent that you want to delete.

-? or -h
Optional. Displays command syntax.

Example

In this example, AGENT3 and its configuration on coordination queue manager QM_COORD1 are deleted:

```
fteDeleteAgent -p QM_COORD1 AGENT3
```

This example command outputs the following MQSC commands to delete the agent's three queues:

```
CLEAR QLOCAL(SYSTEM.FTE.COMMAND.AGENT3)
DELETE QLOCAL(SYSTEM.FTE.COMMAND.AGENT3)
CLEAR QLOCAL(SYSTEM.FTE.DATA.AGENT3)
DELETE QLOCAL(SYSTEM.FTE.DATA.AGENT3)
CLEAR QLOCAL(SYSTEM.FTE.REPLY.AGENT3)
DELETE QLOCAL(SYSTEM.FTE.REPLY.AGENT3)
CLEAR QLOCAL(SYSTEM.FTE.STATE.AGENT3)
DELETE QLOCAL(SYSTEM.FTE.STATE.AGENT3)
CLEAR QLOCAL(SYSTEM.FTE.EVENT.AGENT3)
DELETE QLOCAL(SYSTEM.FTE.EVENT.AGENT3)
CLEAR QLOCAL(SYSTEM.FTE.AUTHADM1.AGENT3)
DELETE QLOCAL(SYSTEM.FTE.AUTHADM1.AGENT3)
CLEAR QLOCAL(SYSTEM.FTE.AUTHAGT1.AGENT3)
DELETE QLOCAL(SYSTEM.FTE.AUTHAGT1.AGENT3)
CLEAR QLOCAL(SYSTEM.FTE.AUTHTRN1.AGENT3)
DELETE QLOCAL(SYSTEM.FTE.AUTHTRN1.AGENT3)
CLEAR QLOCAL(SYSTEM.FTE.AUTHOPS1.AGENT3)
DELETE QLOCAL(SYSTEM.FTE.AUTHOPS1.AGENT3)
CLEAR QLOCAL(SYSTEM.FTE.AUTHSCH1.AGENT3)
DELETE QLOCAL(SYSTEM.FTE.AUTHSCH1.AGENT3)
CLEAR QLOCAL(SYSTEM.FTE.AUTHMON1.AGENT3)
DELETE QLOCAL(SYSTEM.FTE.AUTHMON1.AGENT3)
```

Return codes

0   Command completed successfully.

1   Command ended unsuccessfully.

Related reference:

"fteStopAgent (stop a WebSphere MQ Managed File Transfer agent)" on page 532
Use the **fteStopAgent** command to either stop a WebSphere MQ Managed File Transfer agent in a controlled way or to stop an agent immediately if necessary using the **-i** parameter.

"fteCleanAgent (cleans up a WebSphere MQ Managed File Transfer agent)" on page 425
Use the **fteCleanAgent** command to clean up the queues that a WebSphere MQ Managed File Transfer agent uses, by deleting messages from the persistent and non-persistent queues used by the agent. Use the **fteCleanAgent** command if you are having problems starting an agent, which might be caused by information remaining on the queues used by the agent.

"fteCreateAgent (create a WebSphere MQ Managed File Transfer agent)" on page 429
The **fteCreateAgent** command creates an agent and its associated configuration.

"**fteStartAgent** (start a WebSphere MQ Managed File Transfer agent)" on page 529
The **fteStartAgent** command starts a WebSphere MQ Managed File Transfer agent from the command line.

fteDeleteLogger (delete a WebSphere MQ Managed File Transfer logger)

Use the **fteDeleteLogger** command to delete a WebSphere MQ Managed File Transfer logger and its configuration. Existing log files associated with the logger can either be retained or deleted.

Important: Only users who are WebSphere MQ administrators (and members of the mqm group) can run this command. If you try to run this command as a user who is not a WebSphere MQ administrator, you will receive an error message and the command will not run.

Purpose

Stop the logger with the **fteStopLogger** command before running the **fteDeleteLogger** command.

If you have configured your logger to run as a Windows service, running the **fteDeleteLogger** command deletes the service definition.

The logger configuration directory contains a MQSC script to delete the queues and the subscription for the logger. These queues are as follows:
- SYSTEM.FTE.LOG.CMD.*logger_name*
- SYSTEM.FTE.LOG.RJCT.*logger_name*

The subscription name is as follows:
- SYSTEM.FTE.AUTO.SUB.*logger_name*

The MQSC script can be found at *MQ_DATA_PATH*\mqft\config*coordination_qmgr*\loggers*logger_name*\ *logger_name*_delete.mqsc

Syntax

fteDeleteLogger

```
►►──fteDeleteLogger─────────────────────────────────────loggerName──────────────────►◄
                    └─ -p─configuration options─┘  └─ -f─┘
```

Parameters

-p *configuration_ options*
Optional. Determines the set of configuration options that is used to start the stand-alone database logger. Use the name of a set of configuration options as the value for the **-p** parameter. By convention this value is the name of a coordination queue manager. If you do not specify this parameter, the default set of configuration options is used.

-f Optional. Forces the removal of any log files created by this logger. If this parameter is omitted, any log files created by the logger will be retained, and must be manually removed when they are no longer required.

logger_name
Required. The name of the logger that you want to delete.

-? or -h
Optional. Displays command syntax.

Example

In this example, a logger called logger1 is deleted. The **-f** parameter has been specified, which causes the logger's log files to be removed as well as the logger's configuration files.
```
fteDeleteLogger -f logger1
```

Return codes

0 Command completed successfully.

1   Command ended unsuccessfully.

Related reference:

"fteCreateLogger (create a WebSphere MQ Managed File Transfer logger)" on page 441
Use the **fteCreateLogger** command to create a file or database logger.

"fteStartLogger (start a logger)" on page 531
The **fteStartLogger** command starts a WebSphere MQ Managed File Transfer logging application.

"fteStopLogger (stop a logger)" on page 536
The **fteStopLogger** command stops a logger.

"fteModifyLogger (run a WebSphere MQ Managed File Transfer logging application as a Windows service)" on page 509
Use the **fteModifyLogger** command to modify a logger so that it can be run as a Windows service. You can use this command only on Windows platforms, and you must first stop the logger by using the **fteStopLogger** command.

fteDeleteMonitor (delete a WebSphere MQ Managed File Transfer resource monitor)

Use the **fteDeleteMonitor** command to stop and delete an existing WebSphere MQ Managed File Transfer resource monitor using the command line. Issue this command against the resource monitoring agent.

Purpose

Use the **fteDeleteMonitor** command to stop monitoring a resource and remove the monitor's definition from the monitoring agent. When you run this command, no more polls of the resource occur and no further tasks are started.

You can run the **fteDeleteMonitor** command from any system that can connect to the WebSphere MQ network and subsequently route to the agent's queue manager. Specifically for the command to run, you must have installed a WebSphere MQ Managed File Transfer component (either Service or Agent) on this system and you must have configured this system's WebSphere MQ Managed File Transfer to communicate with the WebSphere MQ network. If no connectivity details are available, the agent queue manager details are used for connection instead, provided these details are available.

Specify the optional **-p** parameter for this command only if you want to use a set of configuration options different from your default set. See Configuration options for more information.

Syntax

fteDeleteMonitor

▶▶──fteDeleteMonitor──-ma──*(monitoring_agent_name)*── -mn ──*(monitor_name)*────────────────▶

▶───◀
   └─ -mm ──*(monitoring_agent_qmgr_name)*─┘ └─-p──*(configuration_options)*─┘

Parameters

-ma (*monitoring_agent_name*)
    Required. The name of the agent that performs the resource monitoring. This monitoring agent must also have been the source agent for the file transfer that you wanted to trigger.

-mn (*monitor_name*)
    Required. The name that you assigned to this resource monitor. You can delete a resource monitor and then create a new monitor with the same name.

-mm (*monitoring_agent_qmgr_name*)
Optional. The name of the monitoring agent's queue manager. Because the monitoring agent and the source agent of the transfer the monitor triggered must be same, this queue manager is also your source agent's queue manager.

-p (*configuration_options*)
Optional. This parameter determines the set of configuration options to use to cancel the transfer. By convention use the name of a non-default coordination queue manager as the input for this parameter. The command then uses the set of properties files associated with this non-default coordination queue manager.

If you do not specify this parameter, the set of configuration options based on the default coordination queue manager is used.

-? or -h
Optional. Displays command syntax.

Example

In this example, the resource monitor MONITOR1 with a monitoring (and file transfer source agent) AGENT1 is deleted:

```
fteDeleteMonitor -ma AGENT1 -mm QM_JUPITER -mn MONITOR1
```

Return codes

0   Command completed successfully.

1   Command ended unsuccessfully.

Related concepts:

"Resource monitoring" on page 215
You can monitor WebSphere MQ Managed File Transfer resources; for example, a queue or a directory. When a condition on this resource is satisfied, the resource monitor starts a task, such as a file transfer. You can create a resource monitor by using the **fteCreateMonitor** command or the Monitors view in the WebSphere MQ Managed File Transfer plug-in for WebSphere MQ Explorer.

Related reference:

"**fteCreateMonitor** (create new resource monitor)" on page 444
The **fteCreateMonitor** command creates and starts a new resource monitor from the command line. You can monitor a resource (for example, the contents of a directory) using WebSphere MQ Managed File Transfer so that when a trigger condition is satisfied, a specified task, such as a file transfer, is started.

"fteListMonitors (list WebSphere MQ Managed File Transfer resource monitors)" on page 497
Use the **fteListMonitors** command to list all of the existing resource monitors in a WebSphere MQ Managed File Transfer network using the command line.

fteDeleteScheduledTransfer (delete a scheduled file transfer)

Purpose

Use the **fteDeleteScheduledTransfer** command to delete a WebSphere MQ Managed File Transfer scheduled transfer that you have previously created either using the command line or the WebSphere MQ Explorer.

Specify the optional **-p** parameter for this command only if you want to use configuration options different from your defaults. If you do not specify **-p**, the configuration options defined in `installation.properties` are used. See "Configuration options" on page 124 for more information.

Syntax

fteDeleteScheduledTransfer
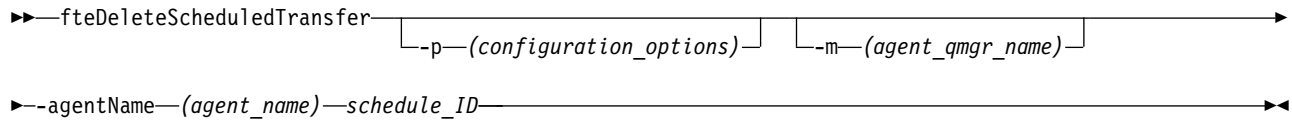
```
►►──fteDeleteScheduledTransfer─────────────────────────────────────────────────►
                                └─-p──(configuration_options)─┘  └─-m──(agent_qmgr_name)─┘

►──-agentName──(agent_name)──schedule_ID──────────────────────────────────────►◄
```

Parameters

-p (*configuration_options***)**
Optional. If you have more than one coordination queue manager, use this parameter to explicitly specify which scheduled transfer you want to delete. Use the name of a non-default coordination queue manager as the input for this parameter. The command then uses the configuration options associated with this non-default coordination queue manager.

If you do not specify this parameter, the configuration options based on the default coordination queue manager are used.

-m (*agent_qmgr_name***)**
Optional. The name of the queue manager that the source agent is connected to. If you do not specify this parameter, the agent's queue manager is determined from the configuration options in use.

-agentName (*agent_name***)**
Required. The name of the source agent that you want to delete the scheduled transfer from.

schedule_ID
Required. The ID of the scheduled transfer that you want to delete.

You can find the schedule ID by running the fteListScheduledTransfers command against the name of the source agent.

-? or -h
Optional. Displays command syntax.

Example

In this example, a scheduled transfer on source agent AGENT2 with the ID 27 is deleted:
```
fteDeleteScheduledTransfer -agentName AGENT2 27
```

Return codes

0   Command completed successfully.

1   Command ended unsuccessfully.

Related tasks:

"Creating a scheduled file transfer" on page 208
You can schedule a new file transfer either from the WebSphere MQ Explorer or from the command line. The scheduled transfer can contain single files or multiple files in a group. You can perform a scheduled file transfer once or repeat the transfer multiple times.

Related reference:

"fteListScheduledTransfers (list scheduled file transfers)" on page 498
Use the **fteListScheduledTransfers** command to list all of the WebSphere MQ Managed File Transfer transfers that you previously created using the command line or the WebSphere MQ Explorer.

fteDeleteTemplates (delete WebSphere MQ Managed File Transfer templates)

Use the **fteDeleteTemplates** command to delete an existing WebSphere MQ Managed File Transfer template from a coordination queue manager.

Purpose

The **fteDeleteTemplates** command removes one or more file transfer templates from a coordination queue manager. When you run this command a request is passed to the WebSphere MQ system to remove the templates from the coordination queue manager so that the templates are no longer available to the WebSphere MQ Explorer or the command line. The templates you are deleting might continue to be accessed for a brief interval after the command completes until the WebSphere MQ system actions the request.

You can run the **fteDeleteTemplates** command from any system that can connect to the WebSphere MQ network and subsequently route to the coordination queue manager. Specifically for the command to run, you must have installed WebSphere MQ Managed File Transfer on this system and you must have configured this system's WebSphere MQ Managed File Transfer to communicate with the WebSphere MQ network. If no connectivity details are available, the agent queue manager details are used for connection instead, provided these details are available.

Specify the optional **-p** parameter for this command only if you want to use a set of configuration options different from your default set. See Configuration options for more information.

Syntax

fteDeleteTemplates

```
►►──fteDeleteTemplates──────────────────────────────template_names──────────────────────►◄
                        └─-p──configuration_options─┘
```

Parameters

-p (*configuration_options***)**
Optional. This parameter determines the set of configuration options to use to delete the template. By convention use the name of a non-default coordination queue manager as the input for this parameter. The command then uses the set of properties files associated with this non-default coordination queue manager.

If you do not specify this parameter, the set of configuration options based on the default coordination queue manager is used.

(*template_names***)**
Required. Specify one or more template names that you want to delete. Specify the name as displayed by the **fteListTemplates** command.

-? or -h
Optional. Displays command syntax.

Example

In this example, the template STANDBY is deleted:
```
fteDeleteTemplates STANDBY
```

Return codes

0   Command completed successfully.

1   Command ended unsuccessfully.

Related tasks:

"Working with transfer templates" on page 231
You can use file transfer templates to store common file transfer settings for repeated or complex transfers. Either create a transfer template from the command line by using the **fteCreateTemplate** command or use the WebSphere MQ Explorer to create a transfer template by using the **Create New Template for Managed File Transfer** wizard, or save a template while you are creating a file transfer by selecting the **Save transfer settings as a template** check box. The **Transfer Templates** window displays all of the transfer templates that you have created in your WebSphere MQ Managed File Transfer network.

"Creating a file transfer template using the WebSphere MQ Explorer" on page 233
You can create a file transfer template from the WebSphere MQ Explorer or from the command line. You can then use that template to create new file transfers using the template details or submit the template to start the file transfer.

Related reference:

"**fteCreateTemplate** (create new file transfer template)" on page 449
The **fteCreateTemplate** command creates a file transfer template that you can keep for future use. The only required parameter is the **-tn** (*template_name*) parameter. All other parameters are optional, although if you specify a source file specification, you must also provide a destination file. Similarly, if you specify a destination file, you must also specify a source file specification.

"fteListTemplates (list WebSphere MQ Managed File Transfer templates)" on page 500
Use the **fteListTemplates** command to list the available WebSphere MQ Managed File Transfer transfer templates on a coordination queue manager.

fteDisplayVersion (display the version of WebSphere MQ Managed File Transfer)

Use the **fteDisplayVersion** command to display the version of WebSphere MQ Managed File Transfer that you have installed.

Purpose

You might be asked to run the **fteDisplayVersion** command by an IBM Service Representative to help with problem determination.

Syntax

fteDisplayVersion

```
►►──fteDisplayVersion──┬────┬──────────────────────────────────────►◄
                       └─-v─┘
```

Parameters

-v Optional. Displays a verbose amount of information about the product version.

The precise details that are displayed when you specify the **-v** parameter might vary between product releases. You are not recommended to rely on specific information being available in the output from the `fteDisplayVersion -v` command.

-? or -h
Optional. Displays command syntax.

Example

In this example, the **fteDisplayVersion** command is specified with no parameters.

```
fteDisplayVersion
```

The output from this command is the product version level as follows:

```
5724-H72 Copyright IBM Corp.  2008, 2012.  ALL RIGHTS RESERVED
Name:         WebSphere MQ Managed File Transfer
Version:      7.5
```

In this example, the **fteDisplayVersion** command is specified with the **-v** parameter.

```
fteDisplayVersion -v
```

The output from this command is the following more detailed information about the product version:

```
C:\Program Files\IBM\WebSphere MQ\bin>fteDisplayVersion.cmd -v
5724-H72 Copyright IBM Corp.  2008, 2012.  ALL RIGHTS RESERVED
Name:         WebSphere MQ Managed File Transfer
Version:      7.5
Level:        f000-20120518-1027
Platform:     Windows 7 (6.1 build 7601 Service Pack 1)
Architecture: x86
JVM:          JRE 1.6.0 IBM J9 2.4 Windows 7 x86-32 jvmwi3260sr10fp1-20120202_101568 (JIT enabled, AOT enabled)
              J9VM - 20120202_101568
              JIT  - r9_20111107_21307ifix1
              GC   - 20120202_AA
Product:      C:\Program Files\IBM\WebSphere MQ
Configuration: C:\Program Files\IBM\WebSphere MQ\mqft

WebSphere MQ Components:

   Name:      Java Message Service Client
   Version:   7.5.0.0
   Level:     p000-L120520

   Name:      WebSphere MQ classes for Java Message Service
   Version:   7.5.0.0
   Level:     p000-L120520

   Name:      IBM WebSphere MQ JMS Provider
   Version:   7.5.0.0
   Level:     p000-L120520

   Name:      Common Services for Java Platform, Standard Edition
   Version:   7.5.0.0
   Level:     p000-L120520
```

Return codes

0   Command completed successfully.

1   Command ended unsuccessfully.

fteListAgents (list the WebSphere MQ Managed File Transfer agents for a coordination queue manager)

Use the **fteListAgents** command to list all of the WebSphere MQ Managed File Transfer agents that are registered with a particular coordination queue manager from the command line.

Purpose

You can run the **fteListAgents** command from any system that can connect to the coordination queue manager. The following details for each agent are directed to the standard output device (STDOUT):

- Agent name
- Agent queue manager
- If the agent is a protocol bridge agent, the agent name is appended with either (FTP bridge) or (SFTP bridge)
- If the agent is a web agent, the agent name is appended with (Web Gateway)
- If the agent is a Connect:Direct bridge agent, the agent name is appended with (Connect:Direct bridge)

- Agent status

This command uses the `coordination.properties` file to connect to the coordination queue manager. For more information, see "The coordination.properties file" on page 540.

Specify the optional **-p** parameter for this command only if you want to use a set of configuration options different from your default set. For more information, see "Configuration options" on page 124.

If an agent is not listed by the **fteListAgents** command, use the diagnosis flowchart in the following topic to locate and fix the problem: If your agent is not listed by the **fteListAgents** command.

Agent status information

The agent status information produced by this command is generated from the status messages that the agent publishes to the SYSTEM.FTE topic. These messages are described in the topic "Agent status message format" on page 611. The status information produced by the **fteListAgents** command gives the agent status at the time when the last status message was published. The frequency of these status messages depends on the value of the agentStatusPublishRateLimit property. For more details about this property, see the topic "The agent.properties file" on page 548.

Syntax

fteListAgents
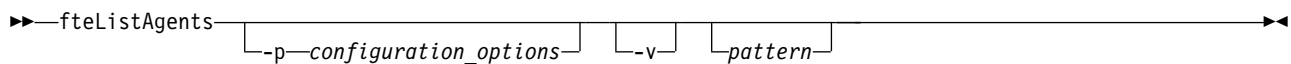
```
►►──fteListAgents──┬─────────────────────────────┬──┬──────┬──┬───────────┬────────────►◄
                   └─-p─configuration_options─┘  └─-v─┘  └─pattern─┘
```

Parameters

-p (configuration_options**)**

Optional. This parameter determines the set of configuration options that is used to issue the request to list agents. Use the name of a non-default coordination queue manager as the input for this parameter. The command then uses the set of properties files associated with this non-default coordination queue manager.

If you do not specify this parameter, the set of configuration options based on the default coordination queue manager is used.

-v

Optional. Specifies verbose mode. Verbose mode generates additional output for each agent, including the current number of transfers in the form `Source/Destination`, where `Source` is the current number of source transfers and `Destination` is the current number of destination transfers.

The current transfer information is obtained from the agent status publication, which is described in the following topic: "Agent status message format" on page 611. As a result, this transfer information is only accurate to within the setting for the agentStatusPublishRateLimit agent property value (which defaults to 30 seconds).

pattern

Optional. The pattern to use to filter the list of WebSphere MQ Managed File Transfer agents. This pattern is matched against the agent name. Asterisk (*) characters are interpreted as wildcards, that match any value, including zero characters.

On UNIX and Linux systems, you must escape special characters like the asterisk (*) and the number sign (#) with quotation marks (' ') or double quotation marks (" ") if you want them to be handled as literals. If you do not escape these characters, they are interpreted according to their meaning on the specific UNIX or Linux system.

If you do not specify this parameter, all agents registered with the coordination queue manager are listed.

-? or -h
  Optional. Displays command syntax.

Example

In this example, all of the agents registered on the queue manager detailed in the configuration options with names beginning with B are listed:

```
fteListAgents "B*"
```

In this example, agents that are registered with the coordination queue manager QM_EUROPE (the non-default coordination queue manager) are listed in verbose mode:

```
fteListAgents -p QM_EUROPE -v
```

The output from this command is as follows:

```
Agent Name:      Queue Manager Name:    Transfers:              Status:
                                        (Source/Destination)
BERLIN           QM_BERLIN              7/0                     RUNNING
LONDON           QM_LONDON              0/0                     RUNNING
MADRID           QM_MADRID              0/1                     UNREACHABLE
```

For a list of the possible agent status values and their meanings, see the topic "Agent status values" on page 665.

In this example, all agents that are registered with the coordination queue manager and that have names beginning with BRIDGE are listed in verbose mode:

```
fteListAgents -v  "BRIDGE*"
```

The output from this command is as follows:

```
C:\Program Files\IBM\WMQFTE\bin>fteListAgents -v
5724-H72 Copyright IBM Corp.  2008, 2010.  ALL RIGHTS RESERVED
Agent Name:                          Queue Manager Name:    Transfers:    Status:
                                                            (Source/Destination)
BRIDGE_FTP (FTP bridge)              QM_JUPITER             0/0           STOPPED
BRIDGE_CD1 (Connect:Direct bridge)   QM_JUPITER             0/0           STOPPED
```

Return codes

0   Command completed successfully.

1   Command ended unsuccessfully.

Related tasks:

"Listing WebSphere MQ Managed File Transfer agents" on page 247
You can list the agents registered with a particular queue manager using the command line or the WebSphere MQ Explorer.

Related reference:

"Agent status values" on page 665
The **fteListAgents** and **fteShowAgentDetails** commands produce agent status information. There are several possible values for this status.

"fteShowAgentDetails (display WebSphere MQ Managed File Transfer agent details)" on page 522
Use the **fteShowAgentDetails** command to display the details of a particular WebSphere MQ Managed File Transfer agent. These are the details that are stored by its WebSphere MQ Managed File Transfer coordination queue manager.

"What to do if the fteListAgents command shows an agent status of UNREACHABLE" on page 351
Your agent is running and responds successfully to the **ftePingAgent** command, and files are being

transferred normally, but the agent is listed as UNREACHABLE by the **fteListAgents** command.

fteListMonitors (list WebSphere MQ Managed File Transfer resource monitors)

Use the **fteListMonitors** command to list all of the existing resource monitors in a WebSphere MQ Managed File Transfer network using the command line.

Purpose

The **fteListMonitors** command lists existing resource monitors. You can filter the command output by specifying an agent name and a resource monitor name.

This command uses the `coordination.properties` file to connect to the coordination queue manager. For more information, see "The coordination.properties file" on page 540.

You can use the **-ox** parameter to export a resource monitor to an XML file. See "**fteCreateMonitor** (create new resource monitor)" on page 444 for information on how to use this XML file.

Specify the optional **-p** parameter for this command only if you want to use a set of configuration options different from your default set. See Configuration options for more information.

Syntax

fteListMonitors

```
►►──fteListMonitors──────────────────────────────────────────────────────►
                     └─-ma─ monitoring_agent_name ─┘  └─-mn ─ monitor_name ─┘

►─────────────────────────────────────────────────────────────────────────►◄
    └─ -ox ─ xml_filename ─┘  └─-v─┘  └─-p─ configuration_options ─┘
```
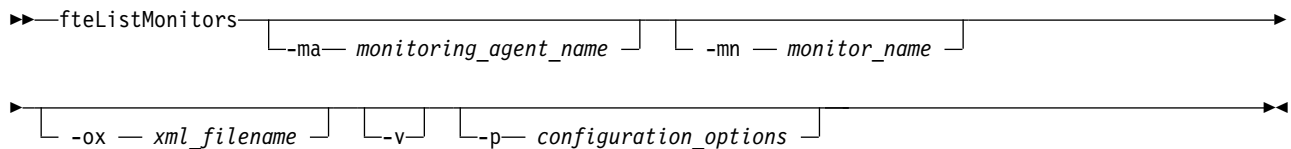
Parameters

-ma (*monitoring_agent_name***)**
    Optional. Filters resource monitors by agent name using the pattern you provide as input. Asterisk (*) characters are interpreted as wildcards that match zero or more characters. If you do not specify the **-ma** parameter, all resource monitors associated with all agents for the default coordination queue manager are listed by default.

-mn (*monitor_name***)**
    Optional. Filters resource monitors by monitor name using the pattern you provide as input. Asterisk (*) characters are interpreted as wildcards that match zero or more characters. If you do not specify the **-mn** parameter, all resource monitors associated with all agents for the default coordination queue manager are listed by default.

-ox (*xml_filename***)**
    Optional. You must specify this parameter with the **-ma** and **-mn** parameters. Exports the resource monitor to an XML file which can then be used by the **fteCreateMonitor** command.

-v Optional. Generates verbose output which includes additional information about the status of the monitor, including whether the monitor is started or stopped, the directory resource path being monitored and the trigger conditions.

-p (*configuration_options***)**
    Optional. This parameter determines the set of configuration options to use to cancel the transfer. By

convention use the name of a non-default coordination queue manager as the input for this parameter. The command then uses the set of properties files associated with this non-default coordination queue manager.

If you do not specify this parameter, the set of configuration options based on the default coordination queue manager is used.

-? or -h
　　Optional. Displays command syntax.

Examples

In this example, all resource monitors associated with the monitoring agent (and source agent for the file transfers associated with the monitor) AGENT1 are listed:

```
fteListMonitors -ma AGENT1
```

In this example the resource monitor MONITOR1 on AGENT1 is exported to the XML file `filename1.xml`:

```
fteListMonitors -ma AGENT1 -mn MONITOR1 -ox filename1.xml
```

Return codes

0　　Command completed successfully.

1　　Command ended unsuccessfully.

Related concepts:

"Resource monitoring" on page 215
You can monitor WebSphere MQ Managed File Transfer resources; for example, a queue or a directory. When a condition on this resource is satisfied, the resource monitor starts a task, such as a file transfer. You can create a resource monitor by using the **fteCreateMonitor** command or the Monitors view in the WebSphere MQ Managed File Transfer plug-in for WebSphere MQ Explorer.

Related reference:

"**fteCreateMonitor** (create new resource monitor)" on page 444
The **fteCreateMonitor** command creates and starts a new resource monitor from the command line. You can monitor a resource (for example, the contents of a directory) using WebSphere MQ Managed File Transfer so that when a trigger condition is satisfied, a specified task, such as a file transfer, is started.

"fteDeleteMonitor (delete a WebSphere MQ Managed File Transfer resource monitor)" on page 489
Use the **fteDeleteMonitor** command to stop and delete an existing WebSphere MQ Managed File Transfer resource monitor using the command line. Issue this command against the resource monitoring agent.

fteListScheduledTransfers (list scheduled file transfers)

Use the **fteListScheduledTransfers** command to list all of the WebSphere MQ Managed File Transfer transfers that you previously created using the command line or the WebSphere MQ Explorer.

Purpose

You can either list all scheduled transfers based on source agent names or based on the coordination queue manager.

Specify the optional **-p** parameter for this command only if you want to use configuration options different from your defaults. If you do not specify **-p**, the configuration options defined in `installation.properties` are used. See "Configuration options" on page 124 for more information.

Syntax

fteListScheduledTransfers
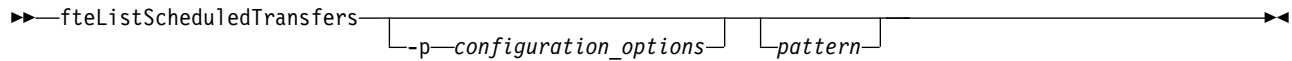
```
►►──fteListScheduledTransfers──┬──────────────────────────┬──┬─────────┬──►◄
                               └─-p──configuration_options─┘  └─pattern─┘
```

Parameters

-p (*configuration_options*)
Optional. If you have more than one coordination queue manager, use this parameter to explicitly specify which agents you want to list scheduled transfers for. Use the name of a non-default coordination queue manager as the input for this parameter. The command then uses the configuration options associated with this non-default coordination queue manager.

If you do not specify this parameter, the configuration options based on the default coordination queue manager are used.

pattern
Optional. The pattern to use to filter the list of WebSphere MQ Managed File Transfer scheduled transfers. This pattern is matched against the source agent name. Asterisk (*) characters are interpreted as wildcards that match zero or more characters.

If you do not specify this parameter, all of the scheduled transfers registered with the coordination queue manager are listed by default.

-? or -h
Optional. Displays command syntax.

Example

In this example, all of the scheduled transfers with source agents that match the pattern *2 are listed:

```
fteListScheduledTransfers "*2"
```

This example command produces the following output. The schedule start time and next transfer time are displayed in Coordinated Universal Time (UTC):

```
Schedule Identifier:    1
Source Agent Name:      AGENT2
Source File Name:       C:/export/Test/workspace/A.exe
Conversion Type:        binary
Destination File Name:  C:/import/Test/workspace/B001.zzx
Destination Agent Name: AGENT1
Schedule Start Time:    2008-10-23T16:08+0100
Next Transfer:          2008-10-23T16:08+0100
Schedule Time Base:     source
Repeat Interval:        minutes
Repeat Frequency:       1
Repeat Count:           30
```

Return codes

0   Command completed successfully.

1   Command ended unsuccessfully.

Related tasks:

"Creating a scheduled file transfer" on page 208
You can schedule a new file transfer either from the WebSphere MQ Explorer or from the command line. The scheduled transfer can contain single files or multiple files in a group. You can perform a scheduled file transfer once or repeat the transfer multiple times.

Related reference:

"fteDeleteScheduledTransfer (delete a scheduled file transfer)" on page 490

fteListTemplates (list WebSphere MQ Managed File Transfer templates)

Use the **fteListTemplates** command to list the available WebSphere MQ Managed File Transfer transfer templates on a coordination queue manager.

Purpose

This command lists either all template names or a filtered selection of template names. The output format of the list can be any of the following:

- Template names only (default behavior)
- Template names with a summary of the templates (verbose mode)
- Complete XML message describing the templates (**-x** and **-o** parameters)

This command uses the `coordination.properties` file to connect to the coordination queue manager. For more information, see "The coordination.properties file" on page 540.

Specify the optional **-p** parameter for this command only if you want to use a set of configuration options different from your default set. See Configuration options for more information.

Syntax

fteListTemplates

►►—fteListTemplates———►
                  └─-p—*configuration_options*─┘ └─-v─┘ └─-x─┘ └─-o—*directory_name*─┘ └─-f─┘

►—*template_names*——►◄

Parameters

-p Optional. This parameter determines the set of configuration options to use to delete the template. By convention use the name of a non-default coordination queue manager as the input for this parameter. The command then uses the set of properties files associated with this non-default coordination queue manager.

If you do not specify this parameter, the set of configuration options based on the default coordination queue manager is used.

-v Optional. Specifies verbose mode and provides a short summary of each matching template. This parameter is ignored if you have also specified the **-x** parameter.

The **-v** parameter includes a summary of each template. For example:

```
Template Name: STANDBY
    Source Agent Name: AGENT1
    Source QMgr: QM_JUPITER
    Destination Agent Name: AGENT2
    Destination QMgr: QM_NEPTUNE
    Transfer Priority: 0
```

```
            Transfer file specification
            File Item Details
                Mode: binary
                Checksum: MD5
                Source File:
                    C:\payroll_reports\*.xls
                    Recursive: false
                    Disposition: leave
                Destination File:
                    C:\payroll_backup\*.xls
                    Type: file
                    Exist: error
```

If you do not specify the **-v** parameter, the default output mode is to list the matching templates names.

-x Optional. Provides an XML-formatted message for each matching template. This parameter is ignored unless you also specify the **-o** parameter.

-o (*directory_name***)**
Optional. Sends the XML formatted-message to files in the named directory. One file for each template is created and each file has the same name as the template with an **.xml** suffix. This parameter is ignored unless you also specify the **-x** parameter.

-f Optional. Forces any existing output file to be overwritten. This parameter is ignored unless you also specify the **-o** parameter. If you do not specify **-f** but you do specify the name of an existing output file, the default behavior is to report an error and continue.

(*template_names***)**
Optional. A list of one or more template names to be listed. A template name can include an asterisk as a wildcard that matches zero or more characters. Depending on your operating system, you might need to enclose any template names that include wildcard character in quotation marks (" ") or single quotation marks (' ') to avoid shell expansion. Shell expansion can cause unexpected behavior.

If you do not specify anything for *template_names*, the default is to list all templates.

-? or -h
Optional. Displays command syntax.

Example

In this example, all the templates with names starting with ST are listed:
```
fteListTemplates "ST*"
```

This example creates the template STANDBY as an XML-formatted message to the file STANDBY.xml in the current directory:
```
fteListTemplates -x -o . STANDBY
```

This command creates the following output in STANDBY.xml:
```
<?xml version="1.0" encoding="UTF-8" ?>
- <transferTemplate id="1864c1dd-ba02-4b34-bda9-dc6862448418" version="3.00">
    <name>STANDBY</name>
    <sourceAgentName>AGENT1</sourceAgentName>
    <sourceAgentQMgr>QM_JUPITER</sourceAgentQMgr>
    <sourceAgentQMgrHost>null</sourceAgentQMgrHost>
    <sourceAgentQMgrPort>-1</sourceAgentQMgrPort>
    <sourceAgentQMgrChannel>null</sourceAgentQMgrChannel>
    <destinationAgentName>AGENT2</destinationAgentName>
    <destinationAgentQMgr>QM_NEPTUNE</destinationAgentQMgr>
  - <fileSpecs>
    - <item checksumMethod="MD5" mode="binary">
      - <source disposition="leave" recursive="false">
          <file>C:\payroll_reports\*.xls</file>
        </source>
      - <destination exist="error" type="file">
```

```
      <file>C:\payroll_backup\*.xls</file>
    </destination>
  </item>
</fileSpecs>
<priority>0</priority>
</transferTemplate>
```

Return codes

0   Command completed successfully.

1   Command ended unsuccessfully.

fteMigrateAgent (migrate a WebSphere MQ File Transfer Edition V7.0 agent to WebSphere MQ V7.5

If you want to migrate an existing agent and its associated configuration from any version of WebSphere MQ File Transfer Edition to WebSphere MQ V7.5, use the **fteMigrateAgent** command to migrate. This command can be used to migrate a standard agent, a Connect:Direct agent, a protocol bridge agent, or a web agent. The command can also be used to migrate multiple agents in a single request.

Important: Only users who are WebSphere MQ administrators (and members of the mqm group) can run this command. If you try to run this command as a user who is not a WebSphere MQ administrator, you will receive an error message and the command will not run.

If your agent is configured to run as a Windows service, use the **fteModifyAgent** command to reconfigure the agent so that it is no longer a Windows service. After the migration is complete, use the **fteModifyAgent** command again to configure the new agent to be a Windows service. Alternatively, if you include the -f parameter, the command completes but produces a warning.

Before you can run the **fteMigrateAgent** command, you must stop the agent you want to migrate using the fteStopAgent command.

If you run the command with the -f parameter, only the information about the agent is refreshed. If a required file is missing, the command fails.

Specifically, the following properties files, XML files, and directory associated with the agent are migrated:

Table 27. Agent files migrated by the fteMigrateAgent command

Name of the file migrated by the fteMigrateAgent command for each agent	Information
wmqfte.properties	The wmqfte.properties file is renamed to installation.properties in WebSphere MQ V7.5
command.properties	
coordination.properties	
coordination_queue_manager.mqsc	
*agent_name*_create.mqsc	
*agent_name*_delete.mqsc	
exits directory	The command copies all files in the exits directory.
Applies to standard and web agents only:	
UserSandboxes.xml	
Applies to Connect:Direct bridge agents only:	
ConnectDirectCredentials.xml	
ConnectDirectNodeProperties.xml	

Table 27. Agent files migrated by the fteMigrateAgent command (continued)

Name of the file migrated by the fteMigrateAgent command for each agent	Information
`ConnectDirectProcessDefinitions.xml`	
Applies to protocol bridge agents only:	
`ProtocolBridgeCredentials.xml`	
`ProtocolBridgeProperties.xml`	This file exists on WebSphere MQ File Transfer Edition V7.0.4.1, or later only.

The **fteMigrateAgent** command migrates the files for the installation, coordination, and command queue managers and copies them to WebSphere MQ V7.5 if the files do not already exist on V7.5. If the files already exist, they are not copied as part of the command.

Syntax

fteMigrateAgent

```
►►──fteMigrateAgent──-agentName────agent_name──-config────configuration_directory──────────────►

►──-credentialPath────credentials_path─────────────────────────────────────────────────────►◄
                                        └─ -f ─┘  └──-p────configuration_options─┘
```

Parameters

-agentName *agent_name*
    Required. The name of the agent that you want to migrate to WebSphere MQ V7.5.

-config *configuration_directory*
    Required. The path to the configuration directory for the installation that you are migrating the agent from. For example, `C:\Documents and Settings\All Users\Application Data\IBM\WMQFTE\config`

-credentialPath *credentials_path*
    Required. The path to the location to migrate the credentials information to. For example, `/home/user1/AGENT3`

-f Optional. Forces the agent to migrate even if some of the configuration files that are typically migrated conflict with the existing configuration. For example, if there is a mismatch a between the properties files on WebSphere MQ File Transfer Edition and the properties files on WebSphere MQ V7.5, specifying the **-f** parameter means that this mismatch is ignored.

-p *configuration_options*
    Optional. This parameter determines the set of configuration options that is used to locate the configuration to migrate. Use the name of a set of configuration options as the value of the **-p** parameter. By convention this is the name of a coordination queue manager. If you do not specify this parameter, the default set of configuration options is used. For more information, see "Configuration options" on page 124.

-? or -h
    Optional. Displays command syntax.

Examples

In this example, AGENT3 and its configuration in `/var/ibm/WMQFTE/config` is migrated to WebSphere MQ V7.5:

```
fteMigrateAgent -agentName AGENT3 -config /var/ibm/WMQFTE/config -credentialPath /home/user1/AGENT3
```

In this example, all agents and their configurations in C:\Documents and Settings\All Users\Application Data\IBM\WMQFTE\config are migrated to WebSphere MQ V7.5. The Windows file path is enclosed in double quotation marks (" "). The **-f** parameter is specified to force migration and ignore any property file mismatches:

```
fteMigrateAgent -agentName "*" -config "C:\Documents and Settings\All Users\Application Data\IBM\WMQFTE\config"
-credentialPath "C:\Documents and Settings\user1\AGENT3" -f
```

Return codes

0   Command completed successfully.

1   Command ended unsuccessfully.

For more information about return codes, see "Return codes for WebSphere MQ Managed File Transfer" on page 373.

fteMigrateConfigurationOptions (migrate a WebSphere MQ File Transfer Edition V7.0 configuration to WebSphere MQ V7.5)

The **fteMigrateConfigurationOptions** command migrates a set of configuration options from WebSphere MQ File Transfer Edition V7.0 and copies them to WebSphere MQ V7.5 provided that the files do not already exist on V7.5. If the files already exist, a message is output and the command does not continue.

Important: Only users who are WebSphere MQ administrators (and members of the mqm group) can run this command. If you try to run this command as a user who is not a WebSphere MQ administrator, you will receive an error message and the command will not run.

Syntax

fteMigrateConfigurationOptions

►►──fteMigrateConfigurationOptions──-config────*configuration_directory*──────────────────────────────►

►──-credentialPath────*credentials_path*──-configurationOptionsName────*configuration_options_name*──────►◄

Parameters

-config (*configuration_directory***)**
　　Required. The path to the configuration directory for the installation that you are migrating from. For example, C:\Documents and Settings\All Users\Application Data\IBM\WMQFTE\config

-credentialPath (*credentials_path***)**
　　Required. The path to the location to migrate the credentials information to. For example, C:\Documents and Settings\user1\configurationoptions

-configurationOptionsName (*configuration_options_name***)**
　　Required. The name of the set of configuration options that you want to migrate. You can migrate multiple sets of configuration options by using the asterisk character (*) to represent zero or more characters. You can use an asterisk with a string. For example, to migrate all sets of configuration options with names beginning with IBM, use this parameter as follows: -configurationOptionsName IBM*.

Examples

In this example, all configurations in the directory C:\Documents and Settings\All Users\Application Data\IBM\WMQFTE\config are migrated. The directory path is enclosed in double quotation marks:

```
fteMigrateConfigurationOptions -config "C:\Documents and Settings\All Users\Application Data\IBM\WMQFTE\config"
-credentialPath "C:\Documents and Settings\user1\configurationoptions" -configurationOptionsName *
```

Return codes

0   Command completed successfully.

1   Command ended unsuccessfully.

Related reference:

"Changes between WebSphere MQ File Transfer Edition V7.0.4 or earlier and WebSphere MQ V7.5" on page 19
If you are planning to move from WebSphere MQ File Transfer Edition V7.0.4, or earlier to WebSphere MQ V7.5, review the following information that summarizes the changes between versions.

"fteMigrateAgent (migrate a WebSphere MQ File Transfer Edition V7.0 agent to WebSphere MQ V7.5" on page 502
If you want to migrate an existing agent and its associated configuration from any version of WebSphere MQ File Transfer Edition to WebSphere MQ V7.5, use the **fteMigrateAgent** command to migrate. This command can be used to migrate a standard agent, a Connect:Direct agent, a protocol bridge agent, or a web agent. The command can also be used to migrate multiple agents in a single request.

fteMigrateLogger (migrate a WebSphere MQ File Transfer Edition V7.0 database logger to WebSphere MQ V7.5)

If you want to migrate the configuration of an existing stand-alone database logger from WebSphere MQ File Transfer Edition V7.0.1 or later to WebSphere MQ V7.5, use the **fteMigrateLogger** command.

You cannot use this command to migrate a JEE database logger: instead use the information in "Migrating a WebSphere Application Server V7 JEE database logger from WebSphere MQ File Transfer Edition V7.0 to WebSphere MQ V7.5" on page 26.

Important: Only users who are WebSphere MQ administrators (and members of the mqm group) can run this command. If you try to run this command as a user who is not a WebSphere MQ administrator, you will receive an error message and the command will not run.

Syntax

fteMigrateLogger

▶▶──fteMigrateLogger──-loggerName────*logger_name*──-config────*configuration_directory*────────────────────▶

▶──-credentialPath────*credentials_path*──┬──────────────────────┬──┬──────┬──────────────────────────▶
                                          └─ -file *properties_file* ─┘ └─ -f ─┘

▶──┬─────────────────────────────┬──▶◀
   └─ -p────*configuration_options*─┘

Parameters

-loggerName *logger_name*
   Required. The name that you want to give to the migrated logger configuration in WebSphere MQ V7.5. For more information about logger names, which are new for V7.5, see logger_name parameter.

-config *configuration_directory*
   Required. The path to the configuration directory for the installation that the logger configuration is being migrated from.

-credentialPath *credentials_path*
   Required. The path to the location to migrate the credentials information to. For example, /home/user1/FTELOGGER1

-file *properties_file*
　　Optional. Specifies the database logger properties file to migrate. This parameter is required only if the properties file does not use the following default name and path: *configuration_directory/coordination_qmgr_name*/databaselogger.properties

-f　Optional. Forces migration even if some of the configuration files that are typically migrated conflict with the existing configuration. For example, if there is a mismatch a between the database logger properties files on WebSphere MQ File Transfer Edition and the properties files on WebSphere MQ V7.5, specifying the **-f** parameter means this mismatch is ignored.

-p *configuration_options*
　　Optional. This parameter determines the set of configuration options that is used to locate the logger configuration to migrate. Use the name of a set of configuration options as the value of the **-p** parameter. By convention, this is the name of a coordination queue manager. If you do not specify this parameter, the default set of configuration options is used. For more information, see "Configuration options" on page 124.

-? or -h
　　Optional. Displays command syntax.

Example

In this example, the configuration of a stand-alone database logger located in /var/ibm/WMQFTE/config is migrated to WebSphere MQ V7.5 and is named FTELOGGER1:

```
fteMigrateLogger -loggerName FTELOGGER1 -config /var/ibm/WMQFTE/config
-credentialPath /home/user1/FTELOGGER1
```

Return codes

0　　　Command completed successfully.

1　　　Command ended unsuccessfully.

For more information about return codes, see "Return codes for WebSphere MQ Managed File Transfer" on page 373.

After running the `fteMigrateLogger` command

To verify the migration, after you have successfully run the **fteMigrateLogger** command, start the database logger whose configuration you have migrated on WebSphere MQ V7.5 using the "fteStartLogger (start a logger)" on page 531 command.

Related reference:

"Changes between WebSphere MQ File Transfer Edition V7.0.4 or earlier and WebSphere MQ V7.5" on page 19
If you are planning to move from WebSphere MQ File Transfer Edition V7.0.4, or earlier to WebSphere MQ V7.5, review the following information that summarizes the changes between versions.

"fteMigrateAgent (migrate a WebSphere MQ File Transfer Edition V7.0 agent to WebSphere MQ V7.5" on page 502
If you want to migrate an existing agent and its associated configuration from any version of WebSphere MQ File Transfer Edition to WebSphere MQ V7.5, use the **fteMigrateAgent** command to migrate. This command can be used to migrate a standard agent, a Connect:Direct agent, a protocol bridge agent, or a web agent. The command can also be used to migrate multiple agents in a single request.

"fteMigrateConfigurationOptions (migrate a WebSphere MQ File Transfer Edition V7.0 configuration to WebSphere MQ V7.5)" on page 504
The **fteMigrateConfigurationOptions** command migrates a set of configuration options from WebSphere MQ File Transfer Edition V7.0 and copies them to WebSphere MQ V7.5 provided that the files do not already exist on V7.5. If the files already exist, a message is output and the command does not continue.

fteModifyAgent (modify a WebSphere MQ Managed File Transfer agent)

The **fteModifyAgent** command modifies an existing agent so that it can be run as a Windows service. This command is only available on Windows.

Important: Only users who are WebSphere MQ administrators (and members of the mqm group) can run this command. If you try to run this command as a user who is not a WebSphere MQ administrator, you will receive an error message and the command will not run.

Syntax

fteModifyAgent
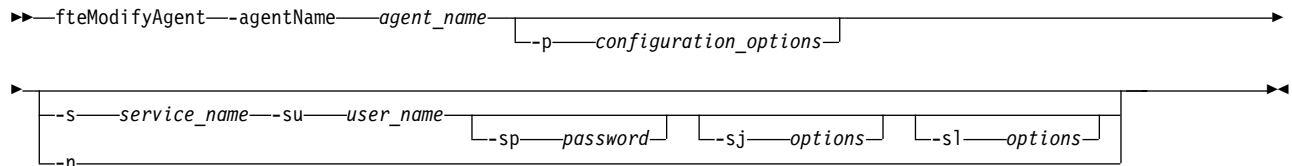
```
▶▶──fteModifyAgent──-agentName────agent_name─────────────────────────────────────────────▶
                                          └─-p────configuration_options─┘

▶───────────────────────────────────────────────────────────────────────────────────────▶◀
   ┌─-s────service_name──-su────user_name───────────────────────────────────────────┐
   │                                  └─-sp────password─┘  └─-sj────options─┘  └─-sl────options─┘ │
   └─-n────────────────────────────────────────────────────────────────────────────┘
```

Parameters

-agentName *agent_name*
  Required. The name of the agent you want to modify.

-p *configuration_options*
  Optional. This parameter determines the set of configuration options that is used to modify the agent. By convention use the name of a non-default coordination queue manager as the input for this parameter. The **fteModifyAgent** command then uses the set of properties files associated with this non-default coordination queue manager.

  Specify the optional **-p** parameter only if you want to use configuration options different from your defaults. If you do not specify this parameter, the set of configuration options based on the default coordination queue manager is used.

-s *service_name*
  Optional. Indicates that the agent is to run as a Windows service. If you do not specify *service_name*, the service is named mqmftAgent*<AGENT><QMGR>*, where *<AGENT>* is the agent name and *<QMGR>* is your agent queue manager name.

  The display name for the service, which is shown in the Windows **Services** window in the **Name** column, is always **WebSphere MQ Managed File Transfer agent *<AGENT>@<QMGR>***.

-su *user_name*
  Optional. When the agent is to run as a Windows service, this parameter specifies the name of the account under which the service should run. To run the agent using a Windows domain user account specify the value in the form DomainName\UserName. To run the service using an account from the local built-in domain specify the value in the form UserName.

  The Windows user account that you specify using the **-su** parameter must have the **Log on as a service** right. For information about how to grant this right, see "Guidance for running an agent or logger as a Windows service" on page 366.

  This parameter is required when **-s** is specified. Equivalent to **-serviceUser**.

-sp *password*
  Optional. Password for the user account set by the **-su** or **-serviceUser** parameter.

This parameter is only valid when **-s** is specified. Equivalent to **-servicePassword**. If you do not specify this parameter when you specify the **-s** parameter, a warning message is produced. This message warns you that you must set the password using the Windows Services tool before the service will start successfully.

-sj *options*

Optional. When the agent is started as a Windows service, defines a list of options in the form of -D or -X that will be passed to the JVM. The options are separated using the number sign (#) or semicolon (;) character. If you need to embed any # or ; characters, put them inside single quotation marks.

This parameter is only valid when **-s** is specified. Equivalent to **-serviceJVMOptions**.

-sl *options*

Optional. Sets the Windows service log level. Valid options are: error, info, warn, debug. The default is info. This option can be useful if you are having problems with the Windows service. Setting it to debug gives more detailed information in the service log file.

This parameter is only valid when **-s** is specified. Equivalent to **-serviceLogLevel**.

-n Optional. Indicates that the agent is to be run as a normal process. This is mutually exclusive with the **-s** option. If neither the **-s** nor the **-n** option is specified, then the agent is configured as a normal Windows process.

Equivalent to **-normal**.

-? or -h

Optional. Displays command syntax.

Example

In this example, AGENT1 is modified to run as a Windows service:

```
fteModifyAgent -agentName AGENT1 -s -su fteuser -sp ftepassword
```

In this example, AGENT1 is modified to remove the Windows service:

```
fteModifyAgent -agentName AGENT1
```

You must stop the agent you want to modify, using the fteStopAgent command, before you can run the fteModifyAgent command.

Return codes

0   Command completed successfully.

1   Command ended unsuccessfully.

Related concepts:

"Guidance for running an agent or logger as a Windows service" on page 366
You can run a WebSphere MQ Managed File Transfer agent, a stand-alone database logger, and a stand-alone file logger, as Windows services. If you are having a problem with these Windows services, you can use the service log files and the information in this topic to diagnose the issue.

Related tasks:

"Starting an agent as a Windows service" on page 201
You can start an agent as a Windows service so that when you log off Windows, your agent continues running and can receive file transfers.

Related reference:

"fteCreateAgent (create a WebSphere MQ Managed File Transfer agent)" on page 429
The **fteCreateAgent** command creates an agent and its associated configuration.

fteModifyLogger (run a WebSphere MQ Managed File Transfer logging application as a Windows service)

Use the **fteModifyLogger** command to modify a logger so that it can be run as a Windows service. You can use this command only on Windows platforms, and you must first stop the logger by using the **fteStopLogger** command.

Important: Only users who are WebSphere MQ administrators (and members of the mqm group) can run this command. If you try to run this command as a user who is not a WebSphere MQ administrator, you will receive an error message and the command will not run.

Purpose

A stand-alone logger, whether for a file or for a database, is shown as "WebSphere MQ Managed File Transfer logger for property set *logger_name@logger_qm*" in the **Name** column of the Services application. The value of *logger_qm* is the name of the command queue manager of the logger.

Syntax

fteModifyLogger

```
►►──fteModifyLogger──-loggerName────(logger_name)──────────────────────────────►
                                              └─ -p──configuration_ options─┘

       ┌─-n────────────────────────────────────────────┐
►──────┤                                                ├──────────────────────►◄
       └─-s──── service_name──┤ Windows systems options ├─┘
```

Windows system options:

```
├──-su──── user_name──────────────────────────────────────────────────────────┤
                     └─-sp──── password─┘  └─-sj──── options─┘  └─-sl──── options─┘
```

Parameters

-loggerName **(***logger_name***)**
    Required. The name of the WebSphere MQ Managed File Transfer logger you want to modify.

-p *configuration_options*
    Optional. This parameter determines the set of configuration options that is used to modify the logger. By convention use the name of a non-default coordination queue manager as the input for this parameter. The **fteModifyLogger** command then uses the set of properties files associated with this non-default coordination queue manager.

    Specify the optional **-p** parameter only if you want to use configuration options different from your defaults. If you do not specify this parameter, the set of configuration options based on the default coordination queue manager is used.

-s *service_name*

Optional. Indicates that the logger is to run as a Windows service. If you do not specify *service_name*, the service is named mqmftLogger<*LOGGER*><*QMGR*>, where <*LOGGER*> is the logger name and <*QMGR*> is your logger queue manager name.

The display name for the service, which is shown in the Windows **Services** window in the **Name** column, is always **WebSphere MQ Managed File Transfer logger <*LOGGER*>@<*QMGR*>**.

-su or -serviceUser *user_name*

Required when **-s** is specified. Specifies the name of the account under which the Windows service should run. To run the agent using a Windows domain user account, specify the value in the form DomainName\UserName. To run the service using an account from the local built-in domain, specify the value in the form UserName.

The Windows user account that you specify using the **-su** parameter must have the permission to log on as a service. For information about how to grant this permission, see "Guidance for running an agent or logger as a Windows service" on page 366.

-sp or -servicePassword *password*

Optional. Only valid when **-s** is specified. Password for the user account set by the **-su** or **-serviceUser** parameter.

If you do not specify this parameter when you specify the **-s** parameter, you are warned that you must set the password by using the Windows Services tool before the service can start successfully.

-sj or -serviceJVMOptions *options*

Optional. Only valid when **-s** is specified. When the logger is started as a Windows service, defines a list of options in the form of -D or -X that will be passed to the JVM. The options are separated using the number sign (#) or semicolon (;) character. If you need to embed any # or ; characters, put them inside single quotation marks (').

-sl or -serviceLogLevel *options*

Optional. Only valid when **-s** is specified. Sets the Windows service log level. Valid options are: error, info, warn, debug. The default is info. This option can be useful if you are having problems with the Windows service. Setting it to debug gives more detailed information in the service log file.

-n or -normal

Optional. Indicates that the logger is to be run as a normal process. This is mutually exclusive with the **-s** option. If neither the **-s** nor the **-n** option is specified, then the logger is configured as a normal Windows process.

-? or -h

Optional. Displays command syntax.

Example

You must stop the logger by using the fteStopLogger command, before running the **fteModifyLogger** command.

In this example, a logger named logger1 has previously been created. This command shows how the logger can be changed to run as a Windows service:

```
fteModifyLogger -loggerName logger1 -s -su fteuser -sp ftepassword
```

Return codes

0   Command completed successfully.

1   Command ended unsuccessfully.

Related concepts:

"Guidance for running an agent or logger as a Windows service" on page 366
You can run a WebSphere MQ Managed File Transfer agent, a stand-alone database logger, and a stand-alone file logger, as Windows services. If you are having a problem with these Windows services, you can use the service log files and the information in this topic to diagnose the issue.

Related tasks:

"Starting an agent as a Windows service" on page 201
You can start an agent as a Windows service so that when you log off Windows, your agent continues running and can receive file transfers.

Related reference:

"fteStartLogger (start a logger)" on page 531
The **fteStartLogger** command starts a WebSphere MQ Managed File Transfer logging application.

"fteStopLogger (stop a logger)" on page 536
The **fteStopLogger** command stops a logger.

fteObfuscate (encrypt sensitive data)

The **fteObfuscate** command encrypts sensitive data in credentials files. This stops the contents of credentials files being read by someone who gains access to the file.

Purpose

User name and password properties in credentials files can be obfuscated. These properties are transformed to a new related property, with a Cipher suffix. For example:

```
<!--
  MQMFTCredentials properties before
-->
<tns:logger name="logger1" user="user1" password="passw0rd" />
<tns:file path="$HOME/trust.jks" password="passw0rd" />

<!--
  MQMFTCredentials properties after
-->
<tns:logger name="logger1" userCipher="e71vKCg2pf" passwordCipher="se71vKCg" />
<tns:file path="$HOME/trust.jks" passwordCipher="e71vKCg2pf" />

<!--
  ProtocolBridgeCredentials Properties before
-->
<tns:user name="Fred" serverUserId="fred" serverPassword="passw0rd" />

<!--
  ProtocolBridgeCredentials properties after
-->
<tns:user name="Fred" serverUserIdCipher="e51vVCg2pf" serverPasswordCipher="se51vBCg" />

<!--
  ConnectDirectCredentials properties before
-->
<tns:user name="fteuser" ignorecase="true" pattern="wildcard"
        cdUserId="cdUser" cdPassword="cdPassword" pnodeUserId="pnodeUser"
        pnodePassword="pnodePassword">
  <tns:snode name="snode1" pattern="wildcard" userId="snodeUser" password="snodePassword"/>
</tns:user>

<!--
  ConnectDirectCredentials properties after
-->
<tns:user name="fteuser" ignorecase="true" pattern="wildcard"
        cdUserIdCipher="e71vKCg2pf" cdPasswordCipher="se71vKCg"
        pnodeUserIdCipher="2f1vgCg6df" pnodePasswordCipher="e71vKCg2pf">
  <tns:snode name="snode1" pattern="wildcard" userIdCipher="e51vVCg2pf" passwordCipher="se51vBCg"/>
</tns:user>
```

Syntax

fteObfuscate

►►—fteObfuscate—-credentialsFile— *credentials_file_name*————————————————◄◄

Parameter

-credentialsFile
   Required. Name of the credentials file whose contents will be obfuscated.

-? or -h
   Optional. Displays command syntax.

Example

In this example, the MQMFTCredentials.xml contents are obfuscated.

fteObfuscate -credentialsFile /home/fteuser/MQMFTCredentials.xml

Return codes

0   Command completed successfully.

1   Command ended unsuccessfully.

ftePingAgent (checks whether a WebSphere MQ Managed File Transfer agent is active)

The **ftePingAgent** command pings a WebSphere MQ Managed File Transfer agent to determine whether the agent is active and able to process transfers.

Purpose

Use the **ftePingAgent** command to check the status of a WebSphere MQ Managed File Transfer agent. Specify the optional **-p** parameter for this command only if you want to use a set of configuration options different from your default set. See Configuration options for more information.

Syntax

ftePingAgent

►►—ftePingAgent——►
                  └─-p——*configuration options*─┘ └─-m——*agent queue manager*─┘

►——————————————————————————*agent_name*—————————————————————————————————————◄◄
   └─-w─┬──────────┬─┘
        └─*timeout*─┘

Parameters

-p (*configuration options***)**
   Optional. This parameter determines the set of configuration options that is used to issue the request to ping an agent. Use the name of a non-default coordination queue manager as the input for this parameter. The command then uses the set of properties files associated with this non-default coordination queue manager. If you do not specify this parameter, the set of configuration options based on the default coordination queue manager is used. See Configuration options for more information.

-m (*queue manager*)

Optional. The name of the queue manager that the agent you want to ping is connected to. If you do not specify the -m parameter the queue manager used is determined from the set of configuration options in use.

-w (*timeout*)

Optional. Specifies that the command should wait for up to *timeout* seconds for the agent to respond. If you do not specify a timeout, or specify a timeout value of -1, then the command waits indefinitely until the agent responds. If you do not specify this option then the default is to wait up to five seconds for the agent to respond.

If *timeout* has been specified, **ftePingAgent** command messages will time out after double the value of *timeout* rather than going to the designated dead letter queue. The command messages will not time out if the command has been set to wait indefinitely.

(*agent name***)**

Required. The name of the WebSphere MQ Managed File Transfer agent that you want to ping.

-? or -h

Optional. Displays command syntax.

Example

In this example, the command pings the agent AGENT1, which is connected to QM_MERCURY. The command waits for up to 40 seconds for AGENT1 to respond before returning.

```
ftePingAgent -m QM_MERCURY -w 40 AGENT1
```

Return codes

0   Command completed successfully. The agent is active and able to process transfers.

1   Command ended unsuccessfully. The command was not able to send a message to the agent.

2   Command ended with a timeout. The command sent a message to the agent, but the agent did not respond within the time.

Related reference:

"fteListAgents (list the WebSphere MQ Managed File Transfer agents for a coordination queue manager)" on page 494
Use the **fteListAgents** command to list all of the WebSphere MQ Managed File Transfer agents that are registered with a particular coordination queue manager from the command line.

"fteShowAgentDetails (display WebSphere MQ Managed File Transfer agent details)" on page 522
Use the **fteShowAgentDetails** command to display the details of a particular WebSphere MQ Managed File Transfer agent. These are the details that are stored by its WebSphere MQ Managed File Transfer coordination queue manager.

"What to do if you think that your transfer is stuck" on page 356
On a heavily loaded system or when there are network problems between the source and destination agents, transfers can occasionally appear to be stuck in a queued or recovering state. There are a number of factors that can cause this.

fteCommon and ftePlatform scripts

The fteCommon and ftePlatform are scripts that are provided by WebSphere MQ Managed File Transfer in the *MQ_INSTALLATION_PATH*/bin directory as helper scripts. Not all of these scripts are present on every platform.

fteCommon

fteCommon is a helper script started by the other WebSphere MQ Managed File Transfer command scripts to perform common setup processing before starting Java.

ftePlatform

ftePlatform is a helper script started by the fteCommon script to perform platform-specific setup processing.

fteSetAgentTraceLevel (set WebSphere MQ Managed File Transfer agent trace level)

Use the **fteSetAgentTraceLevel** command to modify the current trace level for an agent dynamically.

Purpose

Use this command to switch agent trace on and off or to change the level of agent trace that is set. When you use the **fteSetAgentTraceLevel** command, you do not have to shut down and restart an agent to modify the trace level. The trace files produced are located in *MQ_DATA_PATH*/mqft/logs/ *coordination_qmgr_name*/agents/*agent_name*/logs/trace%PID%/trace%PID%.txt, where *%PID%* is the process ID for the agent instance.

Attention: For WebSphere MQ V7.5, only the user that the agent process is running under can run the **fteSetAgentTraceLevel** command.

In WebSphere MQ Managed File Transfer Version 7.5, the **fteSetAgentTraceLevel** command also writes a trace for the agent process controller. The trace files produced are located in *MQ_DATA_PATH*/mqft/logs/ *coordination_qmgr_name*/agents/*agent_name*/logs/pctrace%PID%/pctrace%PID%.txt, where *%PID%* is the process ID for the agent instance.

You can also use the command to cause the agent process to generate a Javacore. The agent generates a Javacore file in the following directory: *MQ_DATA_PATH*/mqft/logs/*coordination_qmgr_name*/agents/ *agent_name*.

Because running trace can affect your performance significantly and can produce a large amount of trace data, run trace with care and only when necessary. Typically, enable trace only when asked to do so by your IBM service representative.

You can set further trace properties, for example trace file size and the number of trace files to keep, in the agent.properties file. These properties are described in Advanced agent properties.

Specify the optional -p parameter for this command only if you want to use a set of configuration options different from your default set. See "The agent.properties file" on page 548 for more information.

Syntax

fteSetAgentTraceLevel

►►—fteSetAgentTraceLevel—-traceAgent——*classes=level*──────────────────────────────►
                                 └—-disableOnAnyFFDC—————────┘
                                 └—-disableOnFFDC——*FFDC_specification*┘

►—└—-jc—┘—└—-p——*configuration_options*—┘—*agent_name*————————————————————►◄

Parameters

-traceAgent *classes=level*
    Required. Level to set the agent trace and which classes to apply the trace to. Specify the following format:

```
classes=level
```

For example:
```
com.ibm.wmqfte=all
```

Specify a comma-separated list of class specifications that you want the level of trace to apply to. If you do not specify this parameter, the trace level is applied to all agent classes.

You can substitute *classes* with an MQMFT package name to trace a specific package only. However, because this option captures just a subset of the agent's behavior, you are generally not recommended to use package filtering.

If (*classes*) start with a plus sign (+), the list of trace classes following the plus sign are added to any existing trace classes currently being traced.

The valid trace level options are as follows and are listed in ascending order of trace file size and detail:

off Switches the agent trace off but continues to write information to the log files. This is the default option.

flow Captures data for trace points associated with processing flow in the agent.

moderate
Captures a moderate amount of diagnostic information in the trace.

verbose
Captures a verbose amount of diagnostic information in the trace.

all Sets agent trace to run on all agent classes.

To start full tracing for the agent, run the following command:
```
fteSetAgentTraceLevel -traceAgent =all AGENT_NAME
```

To stop full tracing for the agent, run the following command:
```
fteSetAgentTraceLevel -traceAgent =off AGENT_NAME
```

-disableOnAnyFFDC
Optional. If this parameter is specified, trace is disabled on the agent when it generates a First Failure Data Capture (FFDC) file.

You can specify only one of the **-disableOnAnyFFDC** and **-disableOnFFDC** parameters.

-disableOnFFDC *FFDC_specification*
Optional. If this parameter is specified, trace is disabled on the agent when it generates a First Failure Data Capture (FFDC) file that matches the *FFDC_specification*. *FFDC_specification* is a comma-separated list of values. The format of the values can be either:

class_name
The name of the class where the FFDC originated. For example, `com.ibm.wmqfte.classA`.

class_name:probe_ID
The name of the class and the probe ID of the location in the class that the FFDC originated from. For example, `com.ibm.wmqfte.classB:1`.

You can specify only one of the **-disableOnAnyFFDC** and **-disableOnFFDC** parameters.

-jc
Optional. Requests that the agent generates a Javacore file. The IBM service team may request that you run the command with this parameter to assist with problem diagnosis. This parameter cannot be used with any other parameter.

-p *configuration_options*
Optional. This parameter determines the set of configuration options that is used to set the agent

trace level. Use the name of a non-default coordination queue manager as the input for this parameter. The command then uses the set of properties files associated with this non-default coordination queue manager.

If you do not specify this parameter, the set of configuration options based on the default coordination queue manager is used.

agent_name
    Required. The name of the WebSphere MQ Managed File Transfer agent that you want to set the trace level for.

-? or -h
    Optional. Displays command syntax.

Example

In this example, the trace level is set to `all` for all classes for AGENT1:

```
fteSetAgentTraceLevel -traceAgent com.ibm.wmqfte=all AGENT1
```

In this example, the trace level is set to `all` for the classes `com.ibm.wmqfte.agent.Agent` and `com.ibm.wmqfte.cmdhandler` for AGENT1:

```
fteSetAgentTraceLevel -traceAgent com.ibm.wmqfte.agent.Agent,com.ibm.wmqfte.cmdhandler=moderate
AGENT1
```

In this example, subclasses are excluded from the trace because the **-traceLevel** parameter is set to off. All classes starting with com.ibm.outer are traced at verbose level except classes starting with com.ibm.outer.inner:

```
fteSetAgentTraceLevel -traceAgent com.ibm.outer=verbose AGENT1
fteSetAgentTraceLevel -traceAgent +com.ibm.outer.inner=off AGENT1
```

Return codes

0   Command completed successfully.

1   Command ended unsuccessfully.

fteSetLoggerTraceLevel

Use the **fteSetLoggerTraceLevel** command to modify the current trace level for a WebSphere MQ Managed File Transfer logger dynamically.

Purpose

Use this command to switch logger trace on and off or change the level of logger trace that is set. When you use the **fteSetLoggerTraceLevel** command, you do not have to shut down and restart a logger to modify the trace level. The trace files produced are located in *MQ_DATA_PATH*/mqft/logs/ *coordination_qmgr_name*/loggers/*logger_name*/logs/trace%PID%/trace%PID%.txt, where *%PID%* is the process ID for the logger instance.

In WebSphere MQ Managed File Transfer Version 7.5 and later, the **fteSetLoggerTraceLevel** command also writes a trace for the logger process controller. The trace files produced are located in *MQ_DATA_PATH*/mqft/logs/*coordination_qmgr_name*/loggers/*logger_name*/logs/pctrace%PID%/pctrace%PID %.txt, where *%PID%* is the process ID for the logger instance.

The command can also be used to cause the logger process to generate a Javacore. The logger generates a Javacore file in the following directory: *MQ_DATA_PATH*/mqft/logs/*coordination_qmgr_name*/loggers/ *logger_name*.

Because running trace can affect your performance significantly and can produce a large amount of trace data, run trace with care and only when necessary. Typically, enable trace only when asked to do so by your IBM service representative.

You can set further trace properties, for example trace file size and the number of trace files to keep, in the `logger.properties` file. These properties are described in Logger properties.

Specify the optional -p parameter for this command only if you want to use a set of configuration options different from your default set. See "Logger configuration properties for WebSphere MQ Managed File Transfer" on page 144 for more information.

Syntax

fteSetLoggerTraceLevel

```
►►─fteSetLoggerTraceLevel──-traceLogger─ classes=level ────────────────────────────────────►
                                              ┌──-disableOnAnyFFDC──────────────────┐
                                              └──-disableOnFFDC── FFDC_specification─┘

►──┬────┬──┬───────────────────────────┬──logger_name─────────────────────────────►◄
   └-jc─┘  └-p─ configuration_options──┘
```
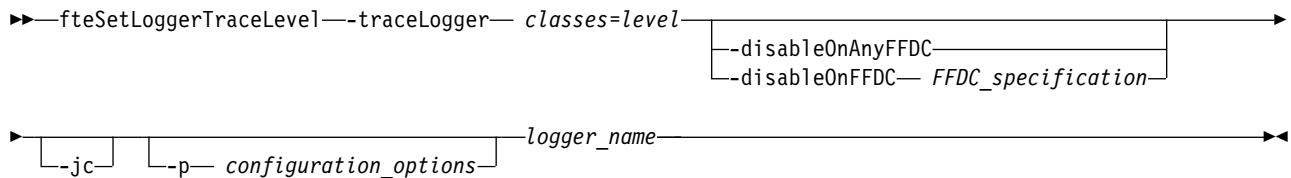
Parameters

-traceLogger *classes=level*
    Required. Level to set the logger trace and which classes to apply the trace to. Specify the following format:

    classes=level

    For example:

    com.ibm.wmqfte=all

    Specify a comma-separated list of class specifications that you want the level of trace to apply to. If you do not specify this parameter, the trace level is applied to all logger classes.

    If (*classes*) start with a plus sign (+), the list of trace classes following the plus sign are added to any existing trace classes currently being traced.

    The valid trace level options are as follows and are listed in ascending order of trace file size and detail:

    off   Switches the logger trace off but continues to write information to the log files. This is the default option.

    flow   Captures data for trace points associated with processing flow in the logger.

    moderate
            Captures a moderate amount of diagnostic information in the trace.

    verbose
            Captures a verbose amount of diagnostic information in the trace.

    all   Sets logger trace to run on all logger classes.

-disableOnAnyFFDC
    Optional. If this parameter is specified, trace is disabled on the logger when it generates a First Failure Data Capture (FFDC) file.

    You can specify only one of the **-disableOnAnyFFDC** and **-disableOnFFDC** parameters.

-disableOnFFDC *FFDC_specification*

Optional. If this parameter is specified, trace is disabled on the logger when it generates a First Failure Data Capture (FFDC) file that matches the *FFDC_specification*. *FFDC_specification* is a comma-separated list of values. The value can be one of the following formats:

class_name

The name of the class where the FFDC originated. For example, `com.ibm.wmqfte.classA`.

class_name:probe_ID

The name of the class and the probe ID of the location in the class that the FFDC originated from. For example, `com.ibm.wmqfte.classB:1`.

You can specify only one of the **-disableOnAnyFFDC** and **-disableOnFFDC** parameters.

-jc

Optional. Requests that the logger generates a Javacore file. The IBM service team might request that you run the command with this parameter to assist with problem diagnosis. You cannot use the **-jc** parameter with any other parameter.

-p *configuration_options*

Optional. This parameter determines the set of configuration options that is used to set the logger trace level. Use the name of a non-default coordination queue manager as the input for this parameter. The command then uses the set of properties files associated with this non-default coordination queue manager.

If you do not specify this parameter, the set of configuration options based on the default coordination queue manager is used.

logger_name

Required. The name of theWebSphere MQ Managed File Transfer logger that you want to set the trace level for.

-? or -h

Optional. Displays command syntax.

Example

In this example, the trace level is set to `all` for all classes for LOGGER1:

```
fteSetLoggerTraceLevel -traceLogger com.ibm.wmqfte=all LOGGER1
```

In this example, the trace level is set to `all` for the classes `com.ibm.wmqfte.logger.logger` and `com.ibm.wmqfte.cmdhandler` for LOGGER1:

```
fteSetLoggerTraceLevel -traceLogger com.ibm.wmqfte.logger.logger,com.ibm.wmqfte.cmdhandler=moderate
LOGGER1
```

In this example, subclasses are excluded from the trace because the **-traceLevel** parameter is set to off. All classes starting with com.ibm.outer are traced at verbose level except classes starting with com.ibm.outer.inner:

```
fteSetLoggerTraceLevel -traceLogger com.ibm.outer=verbose LOGGER1
fteSetLoggerTraceLevel -traceLogger +com.ibm.outer.inner=off LOGGER1
```

Return codes

0   Command completed successfully.

1   Command ended unsuccessfully.

fteSetupCommands (create the command.properties file)

The **fteSetupCommands** command creates the command.properties file. This properties file specifies the details of the queue manager that connects to the WebSphere MQ network when you issue commands.

Important: Only users who are WebSphere MQ administrators (and members of the mqm group) can run this command. If you try to run this command as a user who is not a WebSphere MQ administrator, you will receive an error message and the command will not run.

Purpose

Use the **fteSetupCommands** command to create a command.properties file in the coordination queue manager configuration directory. The command uses the install.properties and installation.properties files to determine where to locate the command.properties file. Ensure that you have already created and configured a coordination queue manager before you issue the **fteSetupCommands** command.

For more information about properties files, see "The command.properties file" on page 545.

Syntax

fteSetupCommands

```
►►──fteSetupCommands──-connectionQMgr────connection_qmgr_name───────────────────────►

►──┬──────────────────────────────────────────────────┬──┬───────────────────────────────────────────────────┬──►
   └──-connectionQMgrHost────connection_qmgr_host──┘  └──-connectionQMgrPort────connection_qmgr_port──┘

►──┬───────────────────────────────────────────────────────────┬──┬──────────────────────────────────┬──┬──────┬──►◄
   └──-connectionQMgrChannel────connection_qmgr_channel──┘  └──-p────configuration_options──┘  └──-f──┘
```
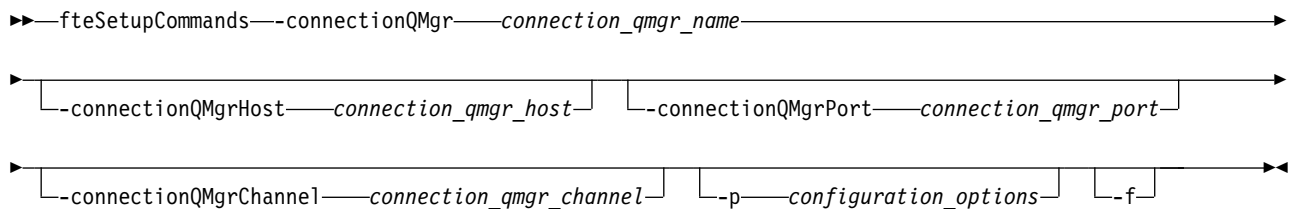
Parameters

-connectionQMgr (*connection_qmgr_name***)**
    Required. The name of the queue manager used to connect to the WebSphere MQ network to issue commands.

-connectionQMgrHost (*connection_qmgr_host***)**
    Optional. The host name or IP address of the connection queue manager.

    If you do not specify the **-connectionQMgrHost** parameter, a bindings mode connection is assumed. Therefore, this parameter is required if you are using a client mode connection.

    If you specify a value for the -connectionQMgrHost parameter but do not specify values for the **-connectionQMgrPort** and **-connectionQMgrChannel** properties, a port number of 1414 and a channel of SYSTEM.DEF.SVRCONN are used by default.

-connectionQMgrPort (*connection_qmgr_port***)**
    Optional. The port number used to connect to the connection queue manager in client mode. If you specify the **-connectionQMgrPort** parameter, you must also specify the **-connectionQMgrHost** parameter.

-connectionQMgrChannel (*connection_qmgr_channel***)**
    Optional. The channel name used to connect to the connection queue manager. If you specify the **-connectionQMgrChannel** parameter, you must also specify the **-connectionQMgrHost** parameter.

-p (*configuration_options***)**
    Optional. This parameter determines the set of configuration options that is used to set up a

command queue manager. Use the name of a non-default coordination queue manager as the input for this parameter. The **fteSetupCommands** command then uses the set of properties files associated with this non-default coordination queue manager.

If you do not specify this parameter, the set of configuration options based on the default coordination queue manager is used.

-f Optional. Forces an overwrite of the existing `command.properties` file with the details specified in this command.

-? or -h
Optional. Displays command syntax.

Example

```
fteSetupCommands -connectionQMgr QM_NEPTUNE -connectionQMgrHost 9.146.157.241
-connectionQMgrPort 1414 -connectionQMgrChannel SYSTEM.DEF.SVRCONN
```

Return codes

0   Command completed successfully.

1   Command ended unsuccessfully.

Related reference:

"The command.properties file" on page 545
The `command.properties` file specifies the command queue manager to connect to when you issue commands and the information that WebSphere MQ Managed File Transfer requires to contact that queue manager.

"fteSetupCoordination (set up coordination details)"
The **fteSetupCoordination** command creates properties files and the coordination queue manager directory for WebSphere MQ Managed File Transfer.

fteSetupCoordination (set up coordination details)

The **fteSetupCoordination** command creates properties files and the coordination queue manager directory for WebSphere MQ Managed File Transfer.

Important: Only users who are WebSphere MQ administrators (and members of the mqm group) can run this command. If you try to run this command as a user who is not a WebSphere MQ administrator, you will receive an error message and the command will not run.

Purpose

Use the **fteSetupCoordination** command to create the following WebSphere MQ Managed File Transfer objects:
- Coordination queue manager directory
- Data directory (if this does not exist)
- `installation.properties` file
- `coordination.properties` file

This command also provides you with the following MQSC commands that you must run against your coordination queue manager to configure WebSphere MQ Managed File Transfer. The MQSC commands create a topic, a topic string, the SYSTEM.FTE queue, and the default database logger queues. These commands also update a namelist and set the PSMODE attribute of the coordination queue manager to ENABLED.

```
DEFINE TOPIC('SYSTEM.FTE') TOPICSTR('SYSTEM.FTE') REPLACE
ALTER TOPIC('SYSTEM.FTE') NPMSGDLV(ALLAVAIL) PMSGDLV(ALLAVAIL)
DEFINE QLOCAL(SYSTEM.FTE) LIKE(SYSTEM.BROKER.DEFAULT.STREAM) REPLACE
ALTER QLOCAL(SYSTEM.FTE) DESCR('Stream for WMQFTE Pub/Sub interface')
```

```
* Altering namelist: SYSTEM.QPUBSUB.QUEUE.NAMELIST
* Value prior to alteration:
DISPLAY NAMELIST(SYSTEM.QPUBSUB.QUEUE.NAMELIST)
ALTER NAMELIST(SYSTEM.QPUBSUB.QUEUE.NAMELIST) +
 NAMES(SYSTEM.BROKER.DEFAULT.STREAM+
 ,SYSTEM.BROKER.ADMIN.STREAM,SYSTEM.FTE)
* Altering PSMODE.  Value prior to alteration:
DISPLAY QMGR PSMODE
ALTER QMGR PSMODE(ENABLED)
```

For more information about properties files, see Configuration options.

Syntax

fteSetupCoordination

►►──fteSetupCoordination──-coordinationQMgr──── *coordination_qmgr_name*──────────────────────────►

►───►
    └──-coordinationQMgrHost── *coordination_qmgr_host*─┘

►───►
    └──-coordinationQMgrPort── *coordination_qmgr_port*─┘

►───►◄
    └──-coordinationQMgrChannel── *coordination_qmgr_channel*─┘ └──-f─┘ └──-default─┘
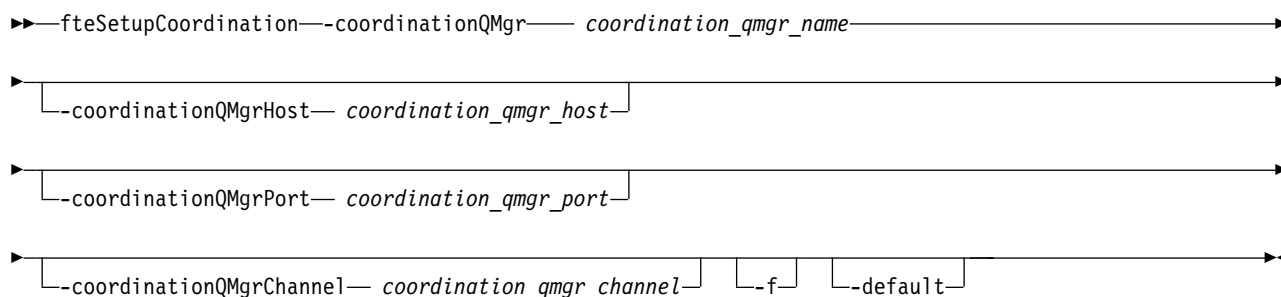
Parameters

-coordinationQMgr (*coordination_qmgr_name* **)**
> Required. The name of the coordination queue manager. This queue manager must be a WebSphere MQ Version 7.0 or later queue manager.

-coordinationQMgrHost (*coordination_qmgr_host***)**
> Optional. The host name or IP address of the coordination queue manager.
>
> If you do not specify the **-coordinationQMgrHost** parameter, a bindings mode connection is assumed.
>
> If you specify a value for the **-coordinationQMgrHost** parameter but do not specify values for the **-coordinationQMgrPort** and **-coordinationQMgrChannel** parameters, a port number of 1414 and a channel of SYSTEM.DEF.SVRCONN are used by default.

-coordinationQMgrPort (*coordination_qmgr_port***)**
> Optional. The port number used for client connections to the coordination queue manager. If you specify the **-coordinationQMgrPort** parameter, you must also specify the **-coordinationQMgrHost** parameter.

-coordinationQMgrChannel (*coordination_qmgr_channel***)**
> Optional. The channel name used to connect to the coordination queue manager. If you specify the **-coordinationQMgrChannel** parameter, you must also specify the **-coordinationQMgrHost** parameter.

-f Optional. Forces an overwrite of the existing coordination queue manager configuration with the details specified in this command.

-default
> Optional. Updates the default configuration options to the options associated with the coordination queue manager specified in this command.

-? or -h
> Optional. Displays command syntax.

Example

In this example, the required objects are set up for a coordination queue manager called QM_SATURN, which is connected to in client mode:

```
fteSetupCoordination -coordinationQMgr QM_SATURN
-coordinationQMgrHost myhost.ibm.com -coordinationQMgrPort 1415
-coordinationQMgrChannel SYSTEM.DEF.SVRCONN
```

Return codes

0   Command completed successfully.

1   Command ended unsuccessfully.

Related concepts:

"Configuration options" on page 124
WebSphere MQ Managed File Transfer provides a set of properties files that contain key information about your setup and are required for operation. These properties files are in the configuration directory that you defined when you installed the product.

Related tasks:

"Configuring the coordination queue manager" on page 128
After running the **fteSetupCoordination** command, run the *coordination_qmgr_name*.mqsc script in the *MQ_DATA_PATH*/mqft/config/*coordination_qmgr_name* directory to perform the necessary configuration for the coordination queue manager. However, if you want to do this configuration manually, complete the following steps on the coordination queue manager.

Related reference:

"The agent.properties file" on page 548
Each agent has its own properties file, `agent.properties`, that must contain the information that an agent uses to connect to its queue manager. The `agent.properties` file can also contain properties that alter the behavior of the agent.

fteShowAgentDetails (display WebSphere MQ Managed File Transfer agent details)

Use the **fteShowAgentDetails** command to display the details of a particular WebSphere MQ Managed File Transfer agent. These are the details that are stored by its WebSphere MQ Managed File Transfer coordination queue manager.

Purpose

You can run the **fteShowAgentDetails** command from any system that can connect to the coordination queue manager. This command uses the `coordination.properties` file to connect to the coordination queue manager.

Specify the optional **-p** parameter for this command only if you want to use a set of configuration options different from your default set. For more information, see "Configuration options" on page 124.

The agent status information produced by this command is generated from the status messages that the agent publishes to the SYSTEM.FTE topic. These messages are described in "Agent status message format" on page 611. The status information produced by the **fteShowAgentDetails** command gives the agent status at the time when the last status message was published. The frequency of these status messages depends on the value of the agentStatusPublishRateLimit property. For more details about this property, see "The agent.properties file" on page 548.

V 7.5.0.1 For IBM WebSphere MQ Version 7.5.0.1 or later, specify the optional **-d** parameter for this command if you want to see diagnostic information about a local agent. This information includes current transfers, scheduled transfers, monitors, and agent queue depths. You can use this information to determine the health and status of a local agent.

For IBM WebSphere MQ Version 7.5, the status of the agent process controller and queue manager is available if you run the command on the same system as the agent. You can use this information to help with problem determination. Also, when you run the command on the same system as the agent, more detailed agent status information is available for the case where the agent has ended unexpectedly.

For a list of the possible agent status values and their meanings, see "Agent status values" on page 665.

For a list of the possible status values for the agent process controller and their meanings, see "Agent process controller status values" on page 666.

For a list of agent trace values and FFDC specifications and their meanings, see "fteSetAgentTraceLevel (set WebSphere MQ Managed File Transfer agent trace level)" on page 345

Syntax

fteShowAgentDetails

```
►►─fteShowAgentDetails─┬──────┬─┬─────────────────────────────┬─┬────┬─┬────┬──agent_name──────────────────►◄
                       └─-bl─┘ └─-p──configuration_options──┘ └─-d─┘ └─-v─┘
```

Parameter

-bl
Optional. Additionally outputs the product build level for the agent.

-p (*configuration_options***)**
Optional. This parameter determines the set of configuration options that is used to issue the request to display the details of an agent. Use the name of a non-default coordination queue manager as the input for this parameter. The command then uses the set of properties files associated with this non-default coordination queue manager.

If you do not specify this parameter, the set of configuration options based on the default coordination queue manager is used.

V 7.5.0.1 **-d**
Optional. Specifies that diagnostic information is displayed for *agent_name*. You can use this parameter only when the agent is running, and on the local system.

V 7.5.0.6 Before IBM WebSphere MQ Version 7.5.0, Fix Pack 6, in order to use the **-d** parameter, the enableFunctionalFixPack property must be set for the appropriate fix pack in the `installation.properties` file. From Version 7.5.0, Fix Pack 6, it is no longer necessary for the enableFunctionalFixPack property to be set when using the **-d** parameter on the **fteShowAgentDetails** command.

-v Optional. Specifies verbose mode, which generates additional output for the agent. These include host name, product version, product build level, trace level, and First Failure Data Capture (FFDC) specification, and a list of transfer states for each of the current source and destination transfers.

The current transfer information is obtained from the agent status publication, which is described in "Agent status message format" on page 611. Therefore this transfer information is only accurate to within the value of the agentStatusPublishRateLimit property. For more details about this property, see "The agent.properties file" on page 548.

agent_name
> Required. The name of the WebSphere MQ Managed File Transfer agent that you want to display.

-? or -h
> Optional. Displays command syntax.

Example

In the following example, running bindings agent, issuing the **fteShowAgentDetails** command locally to the agent:

```
fteShowAgentDetails -v AGENT1

5724-H72 Copyright IBM Corp.  2008, 2012.  ALL RIGHTS RESERVED
Agent Information:
    Name:                       AGENT1
    Type:                       Standard
    Description:
    Operating System:           Windows Server 2003
    Time Zone:                  Greenwich Mean Time
    Product Version:            7.5
    Build Level:                f000-20120312-0957
    Trace Level:                com.ibm.wmqfte.Agent=all
                                com.ibm.wmqfte.common=all
    Trace FFDC:                 com.ibm.wmqfte.common:Any
                                com.ibm.wmqfte.Agent:1


Agent Controller Information:
    Controller type:        MQMFT Process Controller
    Status:                     STARTED
    Status Details:             The agent process controller has
                                started the agent process.
    Agent Restarts within Interval:    0
    Total Agent Restart Count:         0


Agent Availability Information:
    Status:                     READY
    Status Details:             The agent is running and is publishing
                                its status at regular intervals. The
                                last update was received within the
                                expected time period. The agent is
                                ready to process transfers, but none
                                are currently in progress.


Queue Manager Information:
    Name:                       QM1
    Transport:                  Bindings
    Last Status Reported:       AVAILABLE (Last Error MQRC: 0)
    Status Details:             The queue manager is available.

Maximum Number of Running Source Transfers: 25
Maximum Number of Queued Source Transfers: 1000
Source Transfer States:
    No current transfers

Maximum Number of Running Destination Transfers: 25
Destination Transfer States:
    TransferId                                      State
    414d51204d49414f5720202020202020820822c5b4a648c0b20   progress
    414d51204d49414f5720202020202020820822c5b4a346c0b20   progress
```

`▶V 7.5.0.1` In the following example, QMGR1 is the non-default coordination queue manager used as input for the configuration options, and diagnostic information is requested with the **-d** parameter. The **fteShowAgentDetails** command is issued on an IBM WebSphere MQ Version 7.5.0.1 system with a local agent:`▶V 7.5.0.1`

```
fteShowAgentDetails -p QMGR1 -d AGENT1
```

`▶V7.5.0.1`

```
5724-H72 Copyright IBM Corp.  2008, 2012.  ALL RIGHTS RESERVED
Agent Information:
    Name:                        AGENT1
    Type:                        Standard
    Description:
    Operating System:            Linux
    Time Zone:                   Greenwich Mean Time

Agent Controller Information:
    Controller type:             MQMFT Process Controller
    Status:                      STARTED
    Status Details:              The agent process controller has started
                                 the agent process.
    Agent Restarts within Interval: 0
    Total Agent Restart Count:   0

Agent Availability Information:
    Status:                      ACTIVE
    Status Details:              The agent is running and is publishing
                                 its status at regular intervals. The last
                                 update was received within the expected
                                 time period. The agent is currently
                                 processing one or more transfers.

Queue Manager Information:
    Name:                        QMGR1
    Transport:                   Client
    Host:                        host1.hursley.ibm.com
    Port:                        2021
    Channel:                     SYSTEM.DEF.SVRCONN
    Last Status Reported:        UNKNOWN
    Status Details:              Information about the queue manager is
                                 not available because the agent has a
                                 client connection to the queue  manager.

Agent Diagnostic Information:

Command Handler Diagnostics:
    Last Command Queue Read Time:  2012-07-30T15:23:10.705Z
    Pending Command Queue Size:    0
    Last Internal Command Type:    Resync Request (from sender) - 414d5120514d43414e4445202020202079e20f5064230010
    Last Internal Command Time:    2012-07-30T14:17:10.506Z
    Last External Command Type:    New Monitor Request
    Last External Command Time:    2012-07-30T14:10:57.751Z
    Diagnostic Properties File name: C:\Program Files (x86)\IBM\WebSphere
                                 MQ\mqft\logs\MUNGEE\agents\MUNGEE\logs\di
                                 agnostics.20121031.083420.0477.1.properti
                                 Es

Command Handler Worker Thread 0 Diagnostics:
    Status:                      Waiting

Command Handler Worker Thread 1 Diagnostics:
    Status:                      Waiting

Command Handler Worker Thread 2 Diagnostics:
    Status:                      Waiting

Command Handler Worker Thread 3 Diagnostics:
    Status:                      Waiting

Command Handler Worker Thread 4 Diagnostics:
    Status:                      Waiting
```

```
File Transfer Diagnostics:
    Source Transfers:               1
    Destination Transfers:          2

File Transfer 0 Diagnostics:
    Transfer Id:                    414d5120514d43414e4445202020202079e20f5064230010
    Role:                           SOURCE
    State:                          ReSynchronisingTransfer
    Status:                         INACTIVE
    Start Time:                     Not started
    Retry Count:                    0
    CheckPoint Index:               0
    CheckPoint Position:            0

File Transfer 1 Diagnostics:
    Transfer Id:                    414d5120514d43414e44452020202020c8fbd54f144f0d20
    Role:                           DESTINATION
    State:                          RunningTransfer
    CheckPoint Index:               0
    CheckPoint Position:            0
    Write Index:                    0
    Write Position:                 0

File Transfer 2 Diagnostics:
    Transfer Id:                    414d5120514d43414e4445202020202079e20f5086020010
    Role:                           DESTINATION
    State:                          RunningTransfer
    CheckPoint Index:               9
    CheckPoint Position:            0
    Write Index:                    3
    Write Position:                 140923

Monitor 0 Diagnostics:
    Name:                           MONITOR1
    Status:                         STARTED
    Resource Type:                  directory
    Resource:                       /tmp/monitor
    Poll Interval:                  1 minutes
    Batch Size:                     2
    Condition                       Match
    Pattern:                        * (wildcard)
    Executing:                      false
    Last Execute Start Time:        2012-04-04T16:19:01.852Z
    Last Execute End Time:          2012-04-04T16:19:01.852Z
    Last Execute Match Count:       0

Schedule 1 Diagnostics:
    Id:                             1
    Next Trigger Time:              2012-07-17T16:00+0100
    Occurrences So Far:             14
    Repeat Interval:                hours
    Repeat Frequency:               5
    Source Agent:                   AGCANDE
    Destination Agent:              AGCANDE
    Source File:                    /tmp/source/a.txt, ...
    Destination File:               /tmp/dest/a.txt, ...
```

In the following example, stopped bindings agent, issuing the **fteShowAgentDetails** command remotely from the agent:

```
fteShowAgentDetails AGENT2

5724-H72 Copyright IBM Corp.  2008, 2012.  ALL RIGHTS RESERVED
Agent Information:
    Name:                           AGENT2
    Type:                           Standard
    Description:
```

```
    Operating System:          Linux
    Time Zone:                 Greenwich Mean Time

Agent Controller Information:
    Controller type:           MQMFT Process Controller
    Status:                    UNKNOWN
    Status Details:            Information about the agent controller
                               is not available, either because the
                               agent is not running or the agent is
                               running on a different system.
    Agent Restarts within Interval:  0
    Total Agent Restart Count:       0

Agent Availability Information:
    Status:                    STOPPED
    Status Details:            The agent has been stopped. It was shut
                               down in a controlled manner.

Queue Manager Information:
    Name:                      QM2
    Transport:                 Bindings
    Last Status Reported:      UNKNOWN
    Status Details:            Information about the queue manager is
                               not available, either because the agent
                               is not running or the agent is running
                               on a different system.
```

In the following example, bindings agent is waiting to restart with the agent queue manager stopped. The agent has already been restarted once before Total Agent Restart Count: 1, possibly due to a previous agent queue manager restart:

Note: The Last Error MQRC against the Last Status Reported for the queue manager information; this information will remain even when the queue manager becomes available.

```
fteShowAgentDetails AGENT1

5724-H72 Copyright IBM Corp.  2008, 2012.  ALL RIGHTS RESERVED
Agent Information:
    Name:                      AGENT1
    Type:                      Standard
    Description:
    Operating System:          Windows Server 2003
    Time Zone:                 Greenwich Mean Time

Agent Controller Information:
    Controller type:           MQMFT Process Controller
    Status:                    WAITING
    Status Details:            The agent process controller is waiting
                               for the queue manager to become
                               available before starting the agent.
    Agent Restarts within Interval:  0
    Total Agent Restart Count:       1

Agent Availability Information:
    Status:                    STOPPED
    Status Details:            The agent has been stopped. It was shut
                               down in a controlled manner.

Queue Manager Information:
    Name:                      QM1
    Transport:                 Bindings
    Last Status Reported:      UNAVAILABLE (Last Error MQRC: 2059)
    Status Details:            The queue manager is unavailable. It
                               might be that the queue manager has not
                               been started or an incorrect  queue
```

manager name has been configured. Look
up the MQ reason code reported against
the status to understand the problem.

In the following example, the client mode agent has just ended unexpectedly and the agent process
controller tries to recover the situation by restarting it after a delay, specified by the maxRestartDelay
agent property value. The default maxRestartDelay agent property value is -1, and this causes the agent
process controller to terminate; hence in this example the maxRestartDelay property value must have
been set to a value greater than 0. The Current Agent Restart Count: 4 implies that there have been 4
restarts within the maxRestartInterval agent property time period. If the maxRestartCount agent property
is 4 then after 4 restarts within the maxRestartInterval, the agent process controller will wait for
maxRestartDelay seconds before restarting the agent, which is the case here. The Total Agent restart
Count: 8 suggests that this has occurred before. This example is not typical and you would only expect
to see the agent ending unexpectedly if the agent runs out of memory, or a custom user exit has caused
some sort of runtime error. Full details as to why the agent ended unexpectedly are in the agent's
output0.log file:

```
fteShowAgentDetails AGENT3

5724-H72 Copyright IBM Corp.  2008, 2012.  ALL RIGHTS RESERVED
Agent Information:
    Name:                           AGENT3
    Type:                           Standard
    Description:
    Operating System:               Windows Server 2003
    Time Zone:                      Greenwich Mean Time

Agent Controller Information:
    Controller type:                MQMFT Process Controller
    Status:                         RECOVERING
    Status Details:                 The agent process unexpectedly stopped
                                    and the process controller will attempt
                                    to restart it.
    Current Agent Restart Count:    4
    Total Agent Restart Count:      8

Agent Availability Information:
    Status:                         ENDED UNEXPECTEDLY
    Status Details:                 The agent has ended unexpectedly due to
                                    an unrecoverable problem. The agent
                                    will be automatically restarted.

Queue Manager Information:
    Name:                           QM3
    Transport:                      Client
    Host:                           host3.hursley.ibm.com
    Port:                           3031
    Channel:                        SYSTEM.DEF.SVRCONN
```

In the following example, the results for a Connect:Direct bridge agent are displayed:

```
fteShowAgentDetails AG_CD1

5724-H72 Copyright IBM Corp.  2008, 2012.  ALL RIGHTS RESERVED
Agent Information:
    Name:                           AG_CD1
    Type:                           Connect:Direct bridge
    Description:
    Connect:Direct Node Name:       CDNODE
    Connect:Direct Node Host:       localhost:1363
    Operating System:               Windows Server 2003
    Time Zone:                      Greenwich Mean Time

Agent Controller Information:
    Controller type:                MQMFT Process Controller
    Status:                         UNKNOWN
    Status Details:                 Information about the agent controller
```

```
                                     is not available, either because the
                                     agent is not running or the agent is
                                     running on a different system.
        Agent Restarts within Interval:   0
        Total Agent Restart Count:        0

Agent Availability Information:
        Status:                      STOPPED
        Status Details:              The agent has been stopped. It was shut
                                     down in a controlled manner.


Queue Manager Information:
        Name:                        QM_JUPITER
        Transport:                   Bindings
        Last Status Reported:        UNKNOWN
        Status Details:              Information about the queue manager is
                                     not available, either because the agent
                                     is not running or the agent is running
                                     on a different system.
```

Return codes

0   Command completed successfully.

1   Command ended unsuccessfully.

Related reference:

"fteListAgents (list the WebSphere MQ Managed File Transfer agents for a coordination queue manager)" on page 494
Use the **fteListAgents** command to list all of the WebSphere MQ Managed File Transfer agents that are registered with a particular coordination queue manager from the command line.

"Agent status values" on page 665
The **fteListAgents** and **fteShowAgentDetails** commands produce agent status information. There are several possible values for this status.

"Agent process controller status values" on page 666
The **fteShowAgentDetails** command produces agent process controller status information. There are several possible values for this status.

fteStartAgent (start a WebSphere MQ Managed File Transfer agent)

The **fteStartAgent** command starts a WebSphere MQ Managed File Transfer agent from the command line.

Purpose

Use the **fteStartAgent** command to start a WebSphere MQ Managed File Transfer agent. You must start an agent before you can use it to perform file transfers. The **fteStartAgent** command starts an agent on the system where you issue the command: you cannot start an agent on a remote system.

For WebSphere MQ V7.5, the agent process controller manages starting the agent. However, the agent process controller may wait for a period of time, for example where there have been a high rate of agent failures, before attempting to start the agent again. As a WebSphere MQ administrator you can use **fteStartAgent** command to override this wait and initiate a start of the agent. If the agent process controller was waiting for the queue manager to become available this command will also initiate the agent process controller attempting to reconnect to the queue manager.

If you have configured the agent to run as a Windows service by using the fteCreateAgent or fteModifyAgent command, running the **fteStartAgent** command starts the Windows service.

This command returns an error if the agent does not start or is already started. The agent communicates with its queue manager based on the values defined in the agent.properties file.

Specify the optional **-p** parameter for this command only if you want to use a set of configuration options different from your default set. See "The agent.properties file" on page 548 for more information.

The **fteStartAgent** command is not applicable to the IBM 4690 environment. For more information on using WebSphere MQ Managed File Transfer in the IBM 4690 environment, see "Using WebSphere MQ Managed File Transfer in a retail environment" on page 35

Syntax

fteStartAgent

```
▶▶──fteStartAgent──────────────────────────────────────────agent_name────────────────────────────▶◀
                    └─-F─┘  └─-p── configuration_options─┘
```

Parameter

-F Optional. This parameter runs the agent daemon as a foreground process. The default is for the agent daemon to run in the background.

If you are running on Windows, and you have configured the agent to run as a Windows service by using the **fteCreateAgent** or **fteModifyAgent** commands, the **-F** parameter overrides this configuration.

-p *configuration_options*
Optional. This parameter determines the set of configuration options that is used to issue the request to start an agent. Use the name of a non-default coordination queue manager as the input for this parameter. The command then uses the set of properties files associated with this non-default coordination queue manager.

If you do not specify this parameter, the set of configuration options based on the default coordination queue manager is used.

agent_name
Required. The name of the WebSphere MQ Managed File Transfer agent that you want to start.

-? or -h
Optional. Displays command syntax.

Example

In this example, AGENT2 is started and runs in the foreground.
```
fteStartAgent -F AGENT2
```

In the following example (for UNIX and Linux systems), AGENT2 is started with a non-default coordination queue manager, QM_SATURN:
```
./fteStartAgent -p QM_SATURN AGENT2
```

You can also run the command by specifying the path to **fteStartAgent** as follows:
```
<path>/fteStartAgent agentname
```

Return codes

0 Command completed successfully.

1 Command ended unsuccessfully.

Responses

In some circumstances, you might see error messages after running the **fteStartAgent** command:

- If you run the **fteStartAgent** command and see the following error message, your environment probably has additional library paths that conflict with WebSphere MQ Managed File Transfer:

```
BFGCL0001E: An internal error has occurred. The exception was: 'CC=2;RC=2495;AMQ8568:
 The native JNI library 'mqjbnd' was not found. [3=mqjbnd]
```

  If the LD_LIBRARY_PATH or LIBPATH environment variable is set to reference a 64-bit version of the library before the 32-bit version when the agent is running with a 32-bit version of Java (as is currently the case for most platforms), this error occurs.

  To resolve this issue, set the WebSphere MQ Managed File Transfer agent property javaLibraryPath to reference the correct location for the library. For example, for mqjbnd on AIX, set to: /usr/mqm/java/lib. For mqjbnd on Linux, set to: /opt/mqm/java/lib

Related tasks:

"Starting an agent as a Windows service" on page 201
You can start an agent as a Windows service so that when you log off Windows, your agent continues running and can receive file transfers.

"Listing WebSphere MQ Managed File Transfer agents" on page 247
You can list the agents registered with a particular queue manager using the command line or the WebSphere MQ Explorer.

"Stopping a WebSphere MQ Managed File Transfer agent" on page 248
You can stop an agent from the command line. When you stop an agent, you are quiescing the agent and allowing the agent to complete its current file transfer before stopping. You can also specify the **-i** parameter at the command line to stop an agent immediately. When the agent has stopped, you cannot use that agent to transfer files until you restart it.

fteStartLogger (start a logger)

The **fteStartLogger** command starts a WebSphere MQ Managed File Transfer logging application.

Purpose

Use the **fteStartLogger** command to start a logger. The logger can be either a file or database application that runs on the same system as the coordination queue manager. For more information, see the topic "Configuring a WebSphere MQ Managed File Transfer logger" on page 133. For WebSphere MQ V7.5, the logger process controller manages starting the logger. However, the logger process controller may wait for a period of time, for example where there have been a high rate of logger failures, before attempting to start the logger again. As a WebSphere MQ administrator you can use the **fteStartLogger** command to override this wait and initiate a start of the logger. If the logger process controller was waiting for the queue manager to become available this command will also initiate the logger process controller attempting to reconnect to the queue manager.

If you have configured a logger to run as a Windows service by using the fteModifyLogger command, running the **fteStartLogger** command starts the Windows service.

This command returns an error if the logger does not start or is already started. The logger communicates with its queue manager based on the values defined in the logger.properties file.

Specify the **-p** parameter for this command only if you want to use a set of configuration options different from the default. For more information on logger properties, see "Logger configuration properties for WebSphere MQ Managed File Transfer" on page 144

Syntax

fteStartLogger

```
►►──fteStartLogger──────────────────────────────────── logger_name ────────────────────────►◄
                   └─p─configuration_options─┘     └─-F─┘
```

Parameters

logger_name
Required. The name of the WebSphere MQ Managed File Transfer logger you want to start.

-p *configuration_options*
Optional. This parameter determines the set of configuration options that is used to issue the request to start a logger. Use the name of a non-default coordination queue manager as the input for this parameter. **fteStartLogger** then uses the set of properties files associated with this non-default coordination queue manager.

If you do not specify this parameter, the set of configuration options based on the default coordination queue manager is used.

-F Optional. Runs the logger as a foreground process (rather than as the default background process). If you have configured the logger to run as a Windows service by using the **fteModifyLogger** command, the **-F** parameter overrides this configuration.

-? or -h
Optional. Displays command syntax.

Example

In this example, a logger has previously been created named logger1. This command shows how the logger can be started as a foreground process:

```
fteStartLogger -F logger1
```

Return codes

0   Command completed successfully.

1   Command ended unsuccessfully.

Related concepts:

"Configuring a WebSphere MQ Managed File Transfer logger" on page 133

Related reference:

"fteModifyLogger (run a WebSphere MQ Managed File Transfer logging application as a Windows service)" on page 509
Use the **fteModifyLogger** command to modify a logger so that it can be run as a Windows service. You can use this command only on Windows platforms, and you must first stop the logger by using the **fteStopLogger** command.

"fteStopLogger (stop a logger)" on page 536
The **fteStopLogger** command stops a logger.

"Logger error handling and rejection" on page 370
The logger identifies two types of error: per-message errors and general errors.

fteStopAgent (stop a WebSphere MQ Managed File Transfer agent)

Use the **fteStopAgent** command to either stop a WebSphere MQ Managed File Transfer agent in a controlled way or to stop an agent immediately if necessary using the **-i** parameter.

Purpose

When you stop an agent by using the **fteStopAgent** command, you can either allow the agent to complete its current file transfer before stopping, or stop the agent immediately even if the agent is currently transferring a file. When the agent has stopped, you cannot use that agent to transfer files until you restart the agent.

If the agent you wish to stop is connected to the WebSphere MQ network, you can run the **fteStopAgent** command from any system that can connect to the WebSphere MQ network and route to the agent queue manager. Specifically for the command to run, you must have installed and configured a WebSphere MQ Managed File Transfer component (either Service or Agent) on this system to communicate with the WebSphere MQ network. If no connectivity details are available, a bindings mode connection is made to the default queue manager on the local system. If `command.properties` does not exist then an error is generated.

If the agent you wish to stop is not connected to the WebSphere MQ network, for example if the WebSphere MQ network is not currently available, you can only run the **fteStopAgent** command from the system that the agent is running on. In order to stop an agent that is not connected to the WebSphere MQ network you must run the **fteStopAgent** command from the same user the agent is running as. Alternatively, if the agent is running on a Windows system you can run the command as an administrator.

Specify the optional **-p** parameter for this command only if you want to use a set of configuration options different from your default set. See "The agent.properties file" on page 548 for more information.

If your agent is running as a Windows service, running the **fteStopAgent** command stops the Windows service. For more information, see "Starting an agent as a Windows service" on page 201.

The **fteStopAgent** command is not applicable to the IBM 4690 environment. For more information on using WebSphere MQ Managed File Transfer in the IBM 4690 environment, see "Using WebSphere MQ Managed File Transfer in a retail environment" on page 35

Syntax

fteStopAgent

▶▶──fteStopAgent──┬─────────────────────────┬──┬───────────────────────────────┬──┬──────┬──*agent_name*──────────────▶◀
               └─-m──*agent_qmgr_name*─┘ └─-p──*configuration_options*─┘ └─-i─┘

Parameters

-m (*agent_qmgr_name***)**
    Optional. The name of the queue manager that the agent that you want to stop is connected to.

    If the agent is on a remote system, or if the agent is on the local system but you are not the user that started it, you must use the **-m** parameter, and have the appropriate authorities. For more information about authorities, see "Group authorities for resources specific to WebSphere MQ Managed File Transfer" on page 405.

-p (*configuration_options***)**
    Optional. This parameter determines the set of configuration options that is used to issue the request to stop an agent. Use the name of a non-default coordination queue manager as the input for this parameter. The command then uses the set of properties files associated with this non-default coordination queue manager.

    If you do not specify this parameter, the set of configuration options based on the default coordination queue manager is used.

-i
>Optional. Immediately stops the agent. The agent does not complete any transfers that are currently in progress.
>
>If you do not specify the **-i** parameter, the agent completes any transfers currently in progress but the agent does not start any new transfers.

agent_name
>Required. The name of the WebSphere MQ Managed File Transfer agent that you want to stop.

-? or -h
>Optional. Displays command syntax.

Example

In this example the agent AGENT2 on queue manager QM_JUPITER is stopped. The -m parameter is used because this queue manager that AGENT2 is connected to differs from the queue manager specified by the set of configuration options.

```
fteStopAgent -m QM_JUPITER AGENT2
```

Return codes

0   Command completed successfully.

1   Command ended unsuccessfully.

Related tasks:

"Stopping a WebSphere MQ Managed File Transfer agent" on page 248
You can stop an agent from the command line. When you stop an agent, you are quiescing the agent and allowing the agent to complete its current file transfer before stopping. You can also specify the **-i** parameter at the command line to stop an agent immediately. When the agent has stopped, you cannot use that agent to transfer files until you restart it.

Related reference:

"**fteStartAgent** (start a WebSphere MQ Managed File Transfer agent)" on page 529
The **fteStartAgent** command starts a WebSphere MQ Managed File Transfer agent from the command line.

fteStopDatabaseLogger (stop the stand-alone database logger)

The **fteStopDatabaseLogger** command stops the stand-alone database logger.

Purpose

The **fteStopDatabaseLogger** command is supported on WebSphere MQ Managed File Transfer Version 7.0.1 and later.

Use the **fteStopDatabaseLogger** command to stop the stand-alone database logger. The stand-alone database logger is a stand-alone Java application that runs on the same system as the coordination queue manager and the database.

Additional notes about stopping the stand-alone database logger

The **fteStopDatabaseLogger** command sends a message to the command queue used by the stand-alone database logger. If you run **fteStopDatabaseLogger** while the stand-alone database logger is not running, a command message is still placed on the queue. When the stand-alone database logger is next started, the logger immediately receives this command message, and shuts down. If you have issued many stop commands to a stand-alone database logger that is not running, you must repeatedly start the logger until all the stop commands have been consumed. Alternatively, you can clear the command queue to remove all pending commands.

If your stand-alone database logger is running as a Windows service, running the **fteStopDatabaseLogger** command stops the Windows service.

Some error conditions, typically accompanied by message BFGDB0038E, prevent the stand-alone database logger from reading commands. To stop a stand-alone database logger in this state, use your operating system facilities to end the process (for example, the UNIX **kill** command or the Windows Task Manager). The XA transaction protocol used by the stand-alone database logger ensures that no messages are lost when the process is ended.

Syntax

fteStopDatabaseLogger
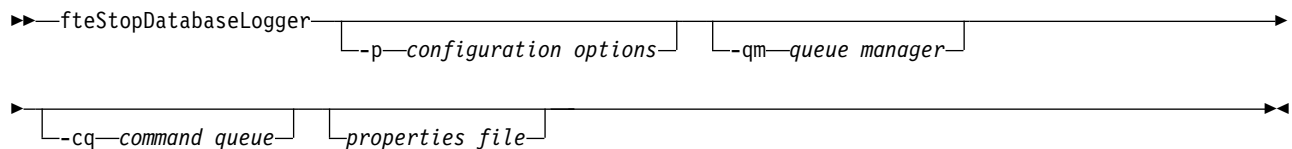
```
►►──fteStopDatabaseLogger─────────────────────────────────────────────────────────►
                          └─-p──configuration options─┘  └─-qm──queue manager─┘

►──────────────────────────────────────────────────────────────────────────────────►◄
  └─-cq──command queue─┘  └─properties file─┘
```

Parameters

-p *(configuration options)*
Optional. Determines the set of configuration options that is used to stop the stand-alone database logger. Use the name of a set of configuration options as the value for the **-p** parameter. By convention this value is the name of a coordination queue manager. If you do not specify this parameter, the default set of configuration options is used.

-qm *(queue manager)*
Optional. By default, the command queue used by the stand-alone database logger is assumed to be on the coordination queue manager specified by the **-p** parameter (or its default). If it is necessary to send stand-alone database logger commands to a command queue located elsewhere, the **-qm** parameter can be used to specify an alternative destination. In all cases, note that the command connects to the command queue manager implied by the **-p** parameter, regardless of the ultimate destination of the message.

-cq *(command queue)*
Optional. Specifies the command queue to which the stop message is sent. In most cases, stand-alone database loggers use the default queue name and this parameter is not necessary.

properties file
Optional. By default, the stand-alone database logger's properties file is assumed to be located in the coordination queue manager's directory. You can optionally supply your own fully qualified path to a properties file containing the required properties for the stand-alone database logger to run. If you specified a properties file for the **fteStartDatabaseLogger** command, specify the same properties file for this command.

-? or -h
Optional. Displays command syntax.

Example

In this example, a stand-alone database logger with command queue, FTE.LOGGER2.COMMAND on queue manager PLUTO, is stopped.
```
fteStopDatabaseLogger -qm PLUTO -cq FTE.LOGGER2.COMMAND
```

Return codes

0   Command completed successfully.

1   Command ended unsuccessfully.

Related concepts:

"Configuring a WebSphere MQ Managed File Transfer logger" on page 133

fteStopLogger (stop a logger)

The **fteStopLogger** command stops a logger.

Purpose

Use the **fteStopLogger** command to stop a logger. The logger can be either a file logger, which records a history of managed file transfer activity to a file, or a database logger which records the history to a database.

Additional notes about stopping a stand-alone logger process

If your logger is running as a Windows service, running the **fteStopLogger** command stops the Windows service.

Syntax

fteStopLogger
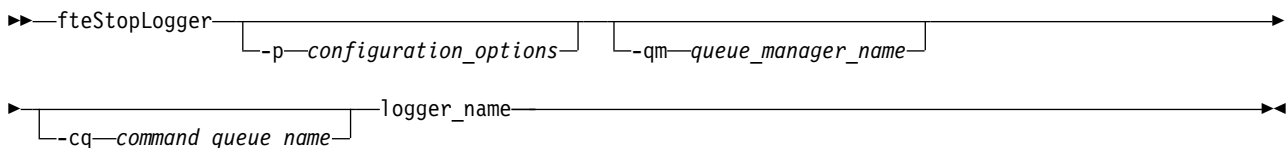
```
►►─fteStopLogger─────────────────────────────────────────────────────────────────────────►
                   └─-p─configuration_options─┘  └─-qm─queue_manager_name─┘

►────────────────────────────────logger_name────────────────────────────────────────────►◄
     └─-cq─command_queue_name─┘
```

Parameters

-p (configuration_options**)**
    Optional. Determines the set of configuration options that is used to stop the logger. Use the name of a set of configuration options as the value for the **-p** parameter. By convention this value is the name of a coordination queue manager. If you do not specify this parameter, the default set of configuration options is used.

-qm (queue_manager_name**)**
    Optional. By default, the logger's command queue is assumed to be on the coordination queue manager specified by the **-p** parameter (or its default). If you want to send logger commands to a command queue located elsewhere, use the **-qm** parameter to specify an alternative destination. In all cases, this command connects to the command queue manager indicated by the **-p** parameter, regardless of the message's ultimate destination.

-cq (command_queue_name**)**
    Optional. Specifies the command queue to send the stop message to. In most cases, loggers use the default queue name meaning this parameter is not necessary.

logger_name
    Required. The name of the WebSphere MQ Managed File Transfer logger you want to stop.

-? or -h
    Optional. Displays command syntax.

Example

In this example, a logger has previously been created named logger1 and is currently running. This command shows how the logger can be stopped:

```
fteStopLogger logger1
```

Return codes

0   Command completed successfully.

1   Command ended unsuccessfully.

Related concepts:

"Configuring a WebSphere MQ Managed File Transfer logger" on page 133

Related reference:

"fteModifyLogger (run a WebSphere MQ Managed File Transfer logging application as a Windows service)" on page 509

Use the **fteModifyLogger** command to modify a logger so that it can be run as a Windows service. You can use this command only on Windows platforms, and you must first stop the logger by using the **fteStopLogger** command.

"fteStartLogger (start a logger)" on page 531

The **fteStartLogger** command starts a WebSphere MQ Managed File Transfer logging application.

Configuring

Environment variables in WebSphere MQ Managed File Transfer properties

In WebSphere MQ V7.5, it is possible for environment variables to be used in Managed File Transfer properties that represent file or directory locations. This allows the locations of files or directories used when running parts of the product, to vary depending on environment changes. For example, which user is running the process.

The following properties accept file or directory locations and can therefore contain environment variables:

- agentSslKeyStore
- agentSslKeyStoreCredentialsFile
- agentSslTrustStore
- agentSslTrustStoreCredentialsFile
- cdNodeKeystoreCredentialsFile
- cdNodeTruststoreCredentialsFile
- cdTmpDir
- cdNodeKeystore
- cdNodeTruststore
- commandPath
- connectionSslKeyStore
- connectionSslKeyStoreCredentialsFile
- connectionSslTrustStore
- connectionSslTrustStoreCredentialsFile
- coordinationSslKeyStore
- coordinationSslKeyStoreCredentialsFile
- coordinationSslTrustStore
- coordinationSslTrustStoreCredentialsFile
- exitClassPath
- exitNativeLibraryPath

- `javaCoreTriggerFile`
- `sandboxRoot`
- `transferRoot`
- `wmqfte.database.credentials.file`

Example

In this example on a Windows 7 system, a user `fteuser` using an environment variable of USERPROFILE:

`wmqfte.database.credentials.file=%USERPROFILE%\\logger\\mqmftcredentials.xml`

Resolves to the following file path:

`C:\Users\fteuser\logger\mqmftcredentials.xml`

In this example on a UNIX system, a user `fteuser` using an environment variable of HOME:

`transferRoot=$HOME/fte/mqmftcredentials.xml`

Resolves to the following file path:

`/home/fteuser/fte/mqmftcredentials.xml`

The installation.properties file

The `installation.properties` file specifies the name of your default set of configuration options. This entry points WebSphere MQ Managed File Transfer to a structured set of directories and property files that contain the configuration to use. Typically the name of a set of configuration options is the name of the associated coordination queue manager.

This file is created by the installer, and can be changed by using the **fteChangeDefaultConfigurationOptions** command.

The `installation.properties` file is located in your *MQ_DATA_PATH* directory. For example on Windows, the default file location is *MQ_DATA_PATH*\mqft\installations*installation_name* and on UNIX and Linux systems, the default file location is /var/mqm/mqft/installations/*installation_name*.

The `installation.properties` file contains the following values:

Table 28. Basic properties

Property name	Description	Default value
`V7.5.0.2` commandMessagePriority	Sets the priority of both internal messages and command messages for the **fteStopAgent**, **fteCancelTransfer** and **ftePingAgent** commands. If you submit a large number of transfer requests to transfer many small files in quick succession, for example, the new transfer requests can become queued on the source agent's command queue. The external and internal messages have the default IBM WebSphere MQ message priority so the internal messages are blocked by the new transfer requests. This can cause the transfer negotiation time to be exceeded and for the transfers to go into recovery. You can also use the commandMessagePriority property to set the priority of internal acknowledgement and acknowledgement-expected messages. To prioritize the internal WebSphere MQ Managed File Transfer messages above new transfer requests, set this property to a value between 1 (the lowest) and 9 (the highest). `V7.5.0.6` From IBM WebSphere MQ Version 7.5.0, Fix Pack 6, the default value is changed to 8. This means that, if the IBM WebSphere MQ attribute DEFPRTY (default priority) on an agent command queue is less than or equal to 7, internal negotiation messages are prioritized ahead of new transfer requests. If the value of the DEFPRTY attribute is set to either 8 or 9, to maintain the effectiveness of the commandMessagePriority property, you must change either DEFPRTY or the commandMessagePriority property.	Before IBM WebSphere MQ Version 7.5.0, Fix Pack 6, the default value is the MQPRI_PRIORITY_AS_Q_DEF constant, which has a value of -1. `V7.5.0.6` From IBM WebSphere MQ Version 7.5.0, Fix Pack 6, the default value is 8.
defaultProperties	The name of the default set of configuration options. This value is the name of a directory located in the configuration directory, which contains directories and properties files that specify configuration information.	No default

Table 28. Basic properties (continued)

Property name	Description	Default value
V 7.5.0.1 enableFunctionalFixPack	The fix pack function level to enable. By default, any new function included with a fix pack is not enabled. Set this property to a version identifier to enable the new features available with that version. For details of the new function associated with each version, start with "What's new in Version 7.5?" on page 11. You can specify the version identifier with or without period characters (.). **V 7.5.0.2** For example, to use the function available with IBM WebSphere MQ Version 7.5.0, Fix Pack 2, set this property to 7502 or 7.5.0.2. For example, to use the function available with IBM WebSphere MQ Version 7.5.0, Fix Pack 1, set this property to 7501 or 7.5.0.1. **V 7.5.0.6** From IBM WebSphere MQ Version 7.5.0, Fix Pack 6, it is no longer necessary for enableFunctionalFixPack to be set when using the **-d** parameter on the **fteShowAgentDetails** command.	No default

The following text is an example of the contents of a `installation.properties` file.

```
defaultProperties=ERIS
```

ERIS is the name of a directory that is located in the same directory as the `installation.properties` file. The directory ERIS contains directories and properties files that describe a set of configuration options.

Related concepts:

"Configuration options" on page 124
WebSphere MQ Managed File Transfer provides a set of properties files that contain key information about your setup and are required for operation. These properties files are in the configuration directory that you defined when you installed the product.

Related reference:

"fteChangeDefaultConfigurationOptions (change the default configuration options)" on page 424
Use the **fteChangeDefaultConfigurationOptions** command to change the default configuration options that you want WebSphere MQ Managed File Transfer to use. The value of the configuration options defines the group of properties files that WebSphere MQ Managed File Transfer uses.

The coordination.properties file

The `coordination.properties` file specifies the connection details to the coordination queue manager. Because several WebSphere MQ Managed File Transfer installations might share the same coordination queue manager, you can use a symbolic link to a common `coordination.properties` file on a shared drive.

The `coordination.properties` file is created by the installer or by the **fteSetupCoordination** command. You can use the **fteSetupCoordination** command with the **-f** flag to change the basic coordination queue manager properties in this file. To change or add advanced coordination queue manager properties you must edit the file in a text editor.

The `coordination.properties` file is located in your *MQ_DATA_PATH*/mqft/config/*coordination_qmgr_name* directory.

The `coordination.properties` file contains the following values:

Table 29. Coordination queue manager properties

Property name	Description	Default value
coordinationQMgr	The name of the coordination queue manager.	No default
coordinationQMgrHost	The host name or IP address of the coordination queue manager.	No default
coordinationQMgrPort	The port number used for client connections to the coordination queue manager.	1414
coordinationQMgrChannel	The SVRCONN channel name used to connect to the coordination queue manager.	SYSTEM.DEF.SVRCONN

If you do not specify a value for the coordinationQMgrHost property, bindings mode is used by default.

If you specify a value for the coordinationQMgrHost property but do not specify values for the coordinationQMgrPort and coordinationQMgrChannel properties, a port number of 1414 and a channel of SYSTEM.DEF.SVRCONN are used by default.

Table 30. Advanced coordination queue manager properties

Property name	Description	Default value
Agent properties:		

Table 30. Advanced coordination queue manager properties (continued)

Property name	Description	Default value
agentStatusJitterTolerance	The maximum amount of time an agent status message publication can be delayed by before the message is considered as overdue. This value is measured in milliseconds. The age of a status message is based on the time at which it was published at the coordination queue manager. However, the message is emitted by the agent some time before it is received at the coordination queue manager to allow for the time required to travel across the IBM WebSphere MQ network. If this transit always takes the same amount of time, messages created 60 seconds apart are published 60 seconds apart, regardless of the actual time in transit. However, if the transit time varies between messages, they might be created at 60-second intervals but published at intervals of, for example, 61, 59, 58, and 62 seconds. The maximum deviation from 60, 2 seconds in this example, is the jitter. This property determines the maximum delay due to jitter before the message is treated as overdue.	3000
Code page properties:		
coordinationCcsid	The code page the commands connect to the coordination queue manager with. Also any publications to the coordination queue manager made by the agent are performed with this code page. If you specify a value for coordinationCcsid you must also specify a value for coordinationCcsidName.	1208
coordinationCcsidName	The Java representation of the coordinationCcsid. If you specify a value for coordinationCcsidName you must also specify a value for coordinationCcsid.	UTF8
Connection properties:		

Table 30. Advanced coordination queue manager properties (continued)

Property name	Description	Default value
javaLibraryPath	When connecting to a queue manager in bindings mode WebSphere MQ Managed File Transfer must have access to the IBM WebSphere MQ Java bindings libraries. By default WebSphere MQ Managed File Transfer looks for the bindings libraries in the default location defined by IBM WebSphere MQ. If the bindings libraries are in a different location use this property to specify the location of the bindings libraries.	`MQ_INSTALLATION_PATH`/java/lib
Multi-instance queue manager properties:		
coordinationQMgrStandby	The host name and the port number used for client connections, in IBM WebSphere MQ CONNAME format, for the standby instance of a multi-instance coordination queue manager defined by the coordinationQMgr property. For example, *host_name*(*port_number*)	No default
Queue properties:		
dynamicQueuePrefix	This property defines the IBM WebSphere MQ prefix to use for generating a temporary queue name. The format of the dynamicQueuePrefix property follows the format of the `DynamicQName` field of the IBM WebSphere MQ MQOD structure. For more information, see Creating dynamic queues. You can also define this property in the `command.properties` file if you want to use a specific IBM WebSphere MQ prefix for temporary reply queues that are generated by commands that require a response from the agent.	WMQFTE.*
modelQueueName	This property defines the IBM WebSphere MQ model queue to use for generating a temporary queue. You can also define this property in the command.properties file if you want to use a specific IBM WebSphere MQ model queue for temporary reply queues that are generated by commands that require a response from the agent. For more information, see "The command.properties file" on page 545.	SYSTEM.DEFAULT.MODEL.QUEUE

Table 30. Advanced coordination queue manager properties (continued)

Property name	Description	Default value
Security properties:		
userIdForClientConnect	The user ID that gets flowed through the client connections to IBM WebSphere MQ. If *java* is specified the user name reported by the JVM is flowed as part of the IBM WebSphere MQ connection request. The value of this property can be None or java.	None
Subscription properties:		
coordinationSubscriptionTopic	Use this property to specify a topic other than SYSTEM.FTE to subscribe to in order to obtain publications about the status of the Managed File Transfer network. All tooling still publishes to the SYSTEM.FTE topic, but you can change your IBM WebSphere MQ topology to distribute these publications to different topics based on their content. You can then use this function to force the tooling to subscribe to one of these other topics. For IBM WebSphere MQ Version 7.5 and later fix packs, you require an interim fix for APAR IC96850 for the property to be recognized by the IBM WebSphere MQ Explorer plug-in and the **fteListMonitors** command.	SYSTEM.FTE

The following text is an example of the contents of a `coordination.properties` file.

```
coordinationQMgr=ERIS
coordinationQMgrHost=kuiper.example.com
coordinationQMgrPort=2005
coordinationQMgrChannel=SYSTEM.DEF.SVRCONN
```

ERIS is the name of a WebSphere MQ queue manager that is located on the system `kuiper.example.com`. The queue manager ERIS is the queue manager that WebSphere MQ Managed File Transfer sends log information to.

Related concepts:

"Configuration options" on page 124
WebSphere MQ Managed File Transfer provides a set of properties files that contain key information about your setup and are required for operation. These properties files are in the configuration directory that you defined when you installed the product.

Related reference:

"fteSetupCoordination (set up coordination details)" on page 520
The **fteSetupCoordination** command creates properties files and the coordination queue manager directory for WebSphere MQ Managed File Transfer.

The command.properties file

The command.properties file specifies the command queue manager to connect to when you issue commands and the information that WebSphere MQ Managed File Transfer requires to contact that queue manager.

The command.properties file is created by the installer or by the **fteSetupCommands** command. You can use the **fteSetupCommands** command with the **-f** flag to change the basic command queue manager properties in this file. To change or add advanced command queue manager properties you must edit the file in a text editor.

Some WebSphere MQ Managed File Transfer commands connect to the agent queue manager or coordination queue manager instead of the command queue manager. For information about which commands connect to which queue manager, see "Which WebSphere MQ Managed File Transfer command connects to which queue manager" on page 419.

The command.properties file is located in your *MQ_DATA_PATH*/mqft/config/*coordination_qmgr_name* directory.

The command.properties file contains the following values:

Table 31. Basic command queue manager properties

Property name	Description	Default value
connectionQMgr	The name of the queue manager used to connect to the WebSphere MQ network.	No default
connectionQMgrHost	The host name or IP address of the connection queue manager.	No default
connectionQMgrPort	The port number used to connect to the connection queue manager in client mode.	1414
connectionQMgrChannel	The SVRCONN channel name used to connect to the connection queue manager.	SYSTEM.DEF.SVRCONN

If you do not specify a value for the connectionQMgrHost property, bindings mode is used by default.

If you specify a value for the connectionQMgrHost property but do not specify values for the connectionQMgrPort and connectionQMgrChannel properties, a port number of 1414 and a channel of SYSTEM.DEF.SVRCONN are used by default.

Table 32. Advanced command queue manager properties

Property name	Description	Default value
Code page properties:		
connectionCcsid	The code page the commands connect to the command queue manager with. If you specify a value for connectionCcsid you must also specify a value for connectionCcsidName.	1208

Table 32. Advanced command queue manager properties (continued)

Property name	Description	Default value
connectionCcsidName	The Java representation of the connectionCcsid. If you specify a value for connectionCcsidName you must also specify a value for connectionCcsid.	UTF8
Multi-instance queue manager properties:		
connectionQMgrStandby	The host name and the port number used for client connections, in WebSphere MQ CONNAME format, for the standby instance of a multi-instance command queue manager defined by the connectionQMgr property. For example, *host_name*(*port_number*)	No default
Security properties:		
userIdForClientConnect	The user ID that gets flowed through the client connections to WebSphere MQ. If *java* is specified the user name reported by the JVM is flowed as part of the WebSphere MQ connection request. The value of this property can be None or java.	None
Queue properties:		
dynamicQueuePrefix	For commands that require a response from the agent, this property defines the WebSphere MQ prefix to use for generating the temporary reply queue name. The format of the dynamicQueuePrefix property follows the format of the `DynamicQName` field of the WebSphere MQ MQOD structure. For more information, see Creating dynamic queues. You can also define this property in the `coordination.properties` file if you want to use a specific WebSphere MQ prefix for temporary queues that are generated by WMQFTE.	WMQFTE.*

Table 32. Advanced command queue manager properties (continued)

Property name	Description	Default value
modelQueueName	For commands that require a response from the agent, this property defines the WebSphere MQ model queue to use for generating the temporary reply queue. You can also define this property in the coordination.properties file if you want to use a specific WebSphere MQ model queue for temporary queues that are generated by WMQFTE. For more information, see "The coordination.properties file" on page 540.	SYSTEM.DEFAULT.MODEL.QUEUE
Connection properties:		
javaLibraryPath	When connecting to a queue manager in bindings mode WebSphere MQ Managed File Transfer must have access to the WebSphere MQ Java bindings libraries. By default WebSphere MQ Managed File Transfer looks for the bindings libraries in the default location defined by WebSphere MQ. If the bindings libraries are in a different location use this property to specify the location of the bindings libraries.	/opt/mqm/java/lib
failCleanAgentWithNoArguments	By default, the value of this property is true, which means that the `fteCleanAgent` command fails to run if only the agent name parameter is specified. Setting the property to false means that, if only the agent name parameter is set, the behavior of the `fteCleanAgent` command is equivalent to specifying the `-all` parameter.	true

The following text is an example of the contents of a `command.properties` file.

```
connectionQMgr=PLUTO
connectionQMgrHost=kuiper.example.com
connectionQMgrPort=1930
connectionQMgrChannel=SYSTEM.DEF.SVRCONN
```

PLUTO is the name of a WebSphere MQ queue manager that is located on the system `kuiper.example.com`. The queue manager PLUTO is the queue manager that the WebSphere MQ Managed File Transfer commands connect to.

Related concepts:

"Configuration options" on page 124
WebSphere MQ Managed File Transfer provides a set of properties files that contain key information about your setup and are required for operation. These properties files are in the configuration directory that you defined when you installed the product.

Related reference:

"Java system properties" on page 597
A number of WebSphere MQ Managed File Transfer command and agent properties must be defined as Java system properties, because they define configuration for early function that is unable to use the command or agent properties mechanism.

"SSL properties" on page 598
Use SSL withWebSphere MQ and WebSphere MQ Managed File Transfer to prevent unauthorized connections between agents and queue managers, and to encrypt message traffic between agents and queue managers.

"fteSetupCommands (create the command.properties file)" on page 519
The **fteSetupCommands** command creates the `command.properties` file. This properties file specifies the details of the queue manager that connects to the WebSphere MQ network when you issue commands.

The agent.properties file

Each agent has its own properties file, `agent.properties`, that must contain the information that an agent uses to connect to its queue manager. The `agent.properties` file can also contain properties that alter the behavior of the agent.

The `agent.properties` file is created by the installer or by the **fteCreateAgent**, **fteCreateWebAgent**, **fteCreateBridgeAgent** or **fteCreateCDAgent** command. You can use any of these commands with the **-f** flag to change the basic agent queue manager properties and those advanced agent properties that are associated with the type of agent that you are creating. To change or add advanced agent properties, you must edit the file in a text editor.

The `agent.properties` file for an agent is in your *MQ_DATA_PATH*/mqft/config/*coordination_qmgr_name*/agents/*agent_name* directory.

If you change the `agent.properties` file you must restart the agent to pick up the changes.

For WebSphere MQ V7.5, you can use environment variables in some WebSphere MQ Managed File Transfer properties that represent file or directory locations. This allows you to use the locations of files or directories when running parts of the product to vary depending on environment changes, such as which user is running the process. For more information, see "Environment variables in WebSphere MQ Managed File Transfer properties" on page 537.

Each `agent.properties` file contains the following values:

Table 33. Agent properties

Property name	Description	Default value
agentName	The name of the agent. The name of the agent must conform to the WebSphere MQ object naming conventions. For more information, see "Object naming conventions for WebSphere MQ Managed File Transfer" on page 664.	No default
agentDesc	The description of the agent - if you choose to create a description.	No default
agentQMgr	The agent queue manager name.	No default
agentQMgrHost	The host name or IP address of the agent queue manager.	No default

Table 33. Agent properties (continued)

Property name	Description	Default value
agentQMgrPort	The port number that is used for client connections to the agent queue manager.	1414
agentQMgrChannel	The SVRCONN channel name that is used to connect to the agent queue manager.	SYSTEM.DEF.SVRCONN
agentType	The type of agent: • Standard non-bridge agent (STANDARD) • Protocol bridge agent (BRIDGE) • Connect:Direct bridge agent (CD_BRIDGE) • Web Gateway agent (WEB_GATEWAY) • Embedded agent as used by IBM Integration Bus (EMBEDDED) • Sterling File Gateway embedded agent (SFG)	STANDARD

If you do not specify a value for the agentQMgrHost property, bindings mode is used by default.

If you specify a value for the agentQMgrHost property but do not specify values for the agentQMgrPort and agentQMgrChannel properties, a port number of 1414 and a channel of SYSTEM.DEF.SVRCONN are used by default.

Advanced agent properties

WebSphere MQ Managed File Transfer also provides more advanced agent properties that help you configure agents. If you want to use any of the following properties, manually edit the `agent.properties` file to add the required advanced properties. When you specify file paths on Windows, ensure the separator character backslash (\) is entered as double backslashes (\\), that is, escaped backslash (\). Alternatively, you can use a single forward slash (/) character as a separator. For more information about character escaping in Java properties files, see the Oracle documentation Javadoc for the Properties class.

- Agent size properties
- Code page properties
- Command properties
- Connection properties
- Connect:Direct bridge properties
- File to message and message to file agent properties
- General agent properties
- Input/output properties
- Multi-channel support properties
- Multi-instance properties
- Process controller properties
- Protocol bridge properties
- Queue properties
- Resource monitoring properties
- Root directory properties
- Security properties
- Timeout properties
- Trace and logging properties

- Transfer limit properties
- User exit routine properties
- WebSphere MQ client compression properties

Table 34. Advanced agent properties

Property name	Description	Default value
Agent size properties:		
agentCheckpointInterval	The interval in complete frames of data between which a checkpoint is taken for recovery purposes. This is an advanced property and for most WebSphere MQ Managed File Transfer configurations it is not necessary to modify its value. If there is a problem which causes the transfer to go into recovery, the transfer can recover only to a checkpoint boundary. Hence, the larger this value (with large agentChunkSize, agentWindowSize, and agentFrameSize values), the longer the time that is needed for the agent to recover transfers. For reliable WebSphere MQ Managed File Transfer networks where transfers rarely enter a recovery state, it may be beneficial to increase this value to increase overall performance.	1
agentChunkSize	The size of each transfer chunk for the transport of file data. Hence, denotes the maximum size of the WebSphere MQ messages that are transferred between the source and the destination agents. This is an advanced property and for most WebSphere MQ Managed File Transfer configurations it is not necessary to modify its value. This value is negotiated between the source agent and the destination agent, and the larger of the two values is used. If you want to change the value of this property, change the value at both the source agent and at the destination agent. agentChunkSize is an integer value. For example: agentChunkSize = 10,240 sets the chunk size to 10 KB.	262144-byte (which is equivalent to 256 KB)
agentFrameSize	The number of windows for the transfer frame. This is an advanced property and for most WebSphere MQ Managed File Transfer configurations it is not necessary to modify its value. For networks that have high latency, increasing this value may improve overall performance as it causes the agent to have more message chunks active concurrently. The value of this property, multiplied by agentWindowSize, multiplied by agentChunkSize, denotes the upper limit of the memory consumption of the agent for each transfer. For example, 262144-byte chunks x 10 x 5 = 12.5 MB for each transfer. Note: If the size of the files that is transferred in a single transfer is less than 12.5 MB increasing this property has no effect on the performance of the transfer.	5
agentWindowSize	The number of chunks for each window. This is an advanced property and for most WebSphere MQ Managed File Transfer configurations it is not necessary to modify its value. For networks that have high latency, increasing this value may improve overall performance. This is because it causes the agent to have more message chunks active concurrently and reduces the frequency that acknowledgement messages are sent back to the source agent. The value of this property, multiplied by agentFrameSize, multiplied by agentChunkSize, denotes the upper limit of the memory consumption of the agent for each transfer, and denotes the upper limit of the WebSphere MQ message data on the command queue of the destination agent. For example, 262144-byte chunks x 10 x 5 = an upper limit of 12.5 MB, for each transfer. Note: If the size of the files that is transferred in a single transfer is less than 12.5 MB increasing the value of this property has no effect on the performance of the transfer.	10
Code page properties:		
agentCcsid	The code page the agent connects to its agent queue manager with. If you specify a value for agentCcsid, you must also specify a value for agentCcsidName. For information on how to view the known code pages for the JVM, see the -hsc parameter in the **fteCreateBridgeAgent** command.	1208

Table 34. Advanced agent properties (continued)

Property name	Description	Default value
agentCcsidName	The Java representation of the agentCcsid. If you specify a value for agentCcsidName, you must also specify a value for agentCcsid.	UTF8
Command properties:		
maxCommandHandlerThreads	Controls the number of threads available for the initial parsing and processing of transfer command messages. When active, the threads require a connection to the queue manager but the threads release the connection when idle.	5
maxCommandOutput	The maximum number of bytes stored for command output. This property applies to commands specified for a managed call and preSource, postSource, preDestination, and postDestination commands for a managed transfer. This limits the length of command output that is written to the transfer log on the SYSTEM.FTE topic.	10 KB
maxCommandRetries	The maximum number of retries for a command that the agent permits. This property applies to commands specified for a managed call and the preSource, postSource, preDestination, and postDestination commands for a managed transfer.	9
maxCommandWait	The maximum wait, in seconds, between retries that the agent permits. This property applies to commands specified for a managed call and the preSource, postSource, preDestination, and postDestination commands for a managed transfer.	60
immediateShutdownTimeout	For an immediate shutdown of an agent, you can use this property to specify the maximum amount of time in seconds an agent waits for its transfers to complete before forcing a shutdown. **Note:** Do not change the value of this property to less than the default of 10 seconds. An immediate shutdown of an agent requires sufficient time to end any external processes. If the value of this property is too low, processes might be left running. If the value 0 is specified for this property, the agent waits for all outstanding transfers to stop. If an invalid value is specified for this property, the default value is used.	10
Connection properties:		
javaLibraryPath	When connecting to a queue manager in bindings mode, WebSphere MQ Managed File Transfer must have access to the WebSphere MQ Java bindings libraries. By default WebSphere MQ Managed File Transfer looks for the bindings libraries in the default location that is defined by WebSphere MQ. If the bindings libraries are in a different location, use this property to specify the location of the bindings libraries.	None
Connect:Direct bridge properties:		
cdNode	Required property if you want to use the Connect:Direct bridge. The name of the Connect:Direct node to use to transfer messages from the Connect:Direct bridge agent to destination Connect:Direct nodes. This node is part of the Connect:Direct bridge, not the remote node that is the source or destination of the transfer. For more information, see "The Connect:Direct bridge" on page 263.	No default
cdNodeHost	The host name or IP address of the Connect:Direct node to use to transfer files from the Connect:Direct bridge agent to destination nodes (the Connect:Direct bridge node). In most cases, the Connect:Direct bridge node is on the same system as the Connect:Direct bridge agent. In these cases, the default value of this property, which is the IP address of the local system, is correct. If your system has multiple IP addresses, or your Connect:Direct bridge node is on a different system to your Connect:Direct bridge agent and their systems share a file system, use this property to specify the correct host name for the Connect:Direct bridge node. If you have not set the cdNode property, this property is ignored.	The host name or IP address of the local system
cdNodePort	The port number of the Connect:Direct bridge node that client applications use to communicate with the node. In Connect:Direct product documentation, this port is referred to as the API port. If you have not set the cdNode property, this property is ignored.	1363

Table 34. Advanced agent properties (continued)

Property name	Description	Default value
cdTmpDir	The location to store files temporarily on the system where the Connect:Direct bridge agent is running before they are transferred to the destination Connect:Direct node. This property specifies the full path of the directory where files are temporarily stored. For example, if cdTmpDir is set to /tmp then the files are temporarily placed in the /tmp directory. The Connect:Direct bridge agent and the Connect:Direct bridge node must be able to access the directory specified by this parameter using the same path name. Consider this when planning the installation of your Connect:Direct bridge. If possible, create the agent on the system where the Connect:Direct node that is part of the Connect:Direct bridge is located. If your agent and node are on separate systems, the directory must be on a shared file system and be accessible from both systems using the same path name. For more information about the supported configurations, see "The Connect:Direct bridge" on page 263. If you have not set the cdNode property, this property is ignored. For WebSphere MQ V7.5, the value of this property can contain environment variables.	*value_of_java.io.tmpdir* /cdbridge-*agentName* On Windows, *value_of_java.io.tmpdir* \cdbridge-*agentName*
cdTrace	Whether the agent traces data that is sent between the Connect:Direct bridge agent and its Connect:Direct node. The value of this property can be true or false.	false
cdMaxConnectionRetries	The maximum number of Connect:Direct connection attempts, for a file transfer where a successful connection has not yet been made, before the transfer fails.	-1 (an infinite number of attempts)
cdMaxPartialWorkConnectionRetries	The maximum number of Connect:Direct connection attempts, for a file transfer where a previous connection attempt has been successful and transfer work has completed, before the transfer fails.	-1 (an infinite number of attempts)
cdMaxWaitForProcessEndStats	The maximum time in milliseconds to wait for Connect:Direct process completion information to become available within the Connect:Direct node statistics information, after the process has ended, before the file transfer is judged to have failed. Typically the information is available immediately, but under certain failure conditions the information is not published. In these conditions the file transfer fails after waiting for the amount of time that is specified by this property.	60000
cdAppName	The application name that the Connect:Direct bridge agent uses to connect to the Connect:Direct node that is part of the bridge.	WebSphere MQ Managed File Transfer *current version*, where *current version* is the version number of the product.
cdNodeLocalPortRange	The range of local ports to use for socket connections between the Connect:Direct bridge agent and the Connect:Direct node that is part of the bridge. The format of this value is a comma-separated list of values or ranges. By default, the operating system selects the local port numbers.	None
cdNodeProtocol	The protocol that the Connect:Direct bridge agent uses to connect to the Connect:Direct node that is part of the bridge. The following values are valid: • TCPIP • SSL • TLS	TCPIP
cdNodeKeystore	The path to the keystore that is used for secure communications between the Connect:Direct bridge agent and the Connect:Direct node that is part of the bridge. If you have not set the cdNodeProtocol property to SSL or TLS, this property is ignored. For WebSphere MQ V7.5, the value of this property can contain environment variables.	None
cdNodeKeystoreType	The file format of the keystore that is specified by the cdNodeKeystore property. The following values are valid: jks and pkcs12. If you have not set the cdNodeProtocol property to SSL or TLS, this property is ignored.	jks

Table 34. Advanced agent properties (continued)

Property name	Description	Default value
cdNodeKeystoreCredentialsFile	The path to the file that contains the cdNodeKeystore credentials. For WebSphere MQ V7.5, the value of this property can contain environment variables.	The default value for this property is %HOMEDRIVE%%HOMEPATH%\\ mqmftcredentials.xml on Windows and $HOME/MQMFTCredentials.xml on other platforms.
cdNodeTruststore	The path to the truststore that is used for secure communications between the Connect:Direct bridge agent and the Connect:Direct node that is part of the bridge. If you have not set the cdNodeProtocol property to SSL or TLS, this property is ignored. For WebSphere MQ V7.5, the value of this property can contain environment variables.	None
cdNodeTruststoreType	The file format of the truststore that is specified by the cdNodeTruststore property. The following values are valid: jks and pkcs12. If you have not set the cdNodeProtocol property to SSL or TLS, this property is ignored.	jks
cdNodeTruststoreCredentialsFile	The path to the file that contains the cdNodeTruststore credentials. For WebSphere MQ V7.5, the value of this property can contain environment variables.	The default value for this property is %HOMEDRIVE%%HOMEPATH%\\ mqmftcredentials.xml on Windows and $HOME/MQMFTCredentials.xml on other platforms.
logCDProcess	The level of Connect:Direct process logging that is recorded in the agent event log in the output0.log file. The values that this property can have are None or Failures or All.	None
File to message and message to file agent properties:		
deleteTmpFileAfterRenameFailure	Setting this property to a value of false ensures that temporary files are not deleted from the destination if the rename operation fails. In this case, the transferred data remains at the destination in a temporary (.part) file. You can manually rename this file later. By default this property has the value of true. This property applies to both message-to-file and file-to-file transfers.	true
enableQueueInputOutput	By default, the agent cannot read data from a source queue or write data to a destination queue as part of a transfer. Setting this value to true enables the agent to perform file to message, and message to file transfers. The value of this property can be true or false.	false
enableSystemQueueInputOutput	Specifies whether the agent can read from or write to WebSphere MQ system queues. System queues are prefixed with the qualifier SYSTEM. **Note:** System queues are used by WebSphere MQ, WebSphere MQ Managed File Transfer, and other applications to transmit important information. Changing this property enables the agent to access these queues. If you enable this property, use user sandboxing to limit the queues that the agent can access.	false
maxDelimiterMatchLength	The maximum number of characters that can be matched by the Java regular expression that is used to split a text file into multiple messages as part of a file-to-message transfer.	5
maxInputOutputMessageLength	The maximum length, in bytes, of a message that is read from a source queue or written to a destination queue by an agent. The maxInputOutputMessageLength property of the source agent in a transfer determines how many bytes can be read from a message on the source queue. The maxInputOutputMessageLength property of the destination agent in a transfer determines how many bytes can be written to a message on the destination queue. If the length of the message exceeds the value of this property the transfer fails with an error. This property does not affect the WebSphere MQ Managed File Transfer internal queues. For information about changing this property, see "Guidance for setting WebSphere MQ attributes and WebSphere MQ Managed File Transfer properties associated with message size" on page 363.	1048576

Table 34. Advanced agent properties (continued)

Property name	Description	Default value
monitorGroupRetryLimit	The maximum number of times that a monitor triggers a message-to-file transfer again if the message group still exists on the queue. The number of times that the message-to-file transfer triggers is determined from the MQMD backout count of the first message in the group. If the agent is restarted the monitor triggers a transfer again even if the number of times the transfer triggers exceeds the value of monitorGroupRetryLimit. If this behavior causes the number of times that the transfer triggers to exceed the value of monitorGroupRetryLimit, the agent writes an error to its event log. If the value -1 is specified for this property, the monitor triggers the transfer again an unlimited number of times until the trigger condition is not satisfied.	10
General agent properties:		
agentStatusPublishRateLimit	The maximum rate in seconds that the agent republishes its status because of a change in file transfer status. If you set this property to too small a value, the performance of the WebSphere MQ network might be negatively affected.	30
agentStatusPublishRateMin	The minimum rate in seconds that the agent publishes its status. This value must be greater than or equal to the value of the agentStatusPublishRateLimit property.	300
▶ V 7.5.0.2 ◀ enableMemoryAllocationChecking	Determines whether the MQMFT agent checks that there is sufficient memory available to run a transfer before a transfer is started. The check is made on both the source and destination agents. If there is insufficient memory available, the transfer is put into recovery, which prevents the agent from failing with an out-of-memory error. When calculating the memory required for a transfer, the maximum memory that is required by the transfer is used. Therefore, the value might be greater than the actual memory that is used by the transfer. For this reason, the number of concurrent transfers that can run might be reduced if the enableMemoryAllocationChecking property is set to true. You are recommended to set the property to true only if you are experiencing problems with MQMFT failing with out-of-memory errors. The transfers that are likely to consume large amounts of memory are file-to-message and message-to-file transfers where the sizes of the messages are large.	false
▶ V 7.5.0.1 ◀ enableUserMetadataOptions	Determines whether you can use known keys for user-defined metadata in transfer requests to provide more transfer options. These known keys always start with the following prefix com.ibm.wmqfte.. As a consequence when the enableUserMetadataOptions property is set true, keys that use this prefix are not supported for user-defined use. When the enableUserMetadataOptions property is set to true, the keys that are supported currently are as follows: • com.ibm.wmqfte.insertRecordLineSeparator • com.ibm.wmqfte.newRecordOnLineSeparator • com.ibm.wmqfte.convertLineSeparators For information about what these keys mean, see "**fteCreateTransfer** (create new file transfer)" on page 461. The value of this property can be true or false.	false

Table 34. Advanced agent properties (continued)

Property name	Description	Default value
V7.5.0.6 failTransferOnFirstFailure	Allows an agent to be configured to fail a managed transfer as soon as a transfer item within that managed transfer fails. To enable this feature, APAR IT03450 must be applied for both the source agent and the destination agent, and the failTransferOnFirstFailure property must be set to `true` in the source agent's `agent.properties` file. Setting the property to true `true` on the destination agent is optional. When the failTransferOnFirstFailure property is set to `true`, the agent starts processing managed transfer requests as normal. However, as soon as a transfer item fails, then the managed transfer is marked as failed and no further transfer items are processed. Transfer items that were successfully processed before the managed transfer failed are handled in the following way: • The source disposition for those transfer items is honored. For example, if the source disposition for the transfer item was set to `delete`, then the source file is deleted. • The destination files that were written remain on the destination file system and are not deleted. If the failTransferOnFirstFailure property is not set to `true` and a managed file transfer contains multiple files and one of these files fails to transfer, for example because the destination file already exists and the overwrite property is set to `error`, the source agent continues and attempts to transfer any remaining files in the request.	false
itemsPerProgressMessage	The number of files that are transferred before an agent publishes its next progress log message. Use this property to control the rate that progress log messages are published to the coordination queue manager during a transfer. The maximum value this property can be set to is 1000. **Note:** Progress messages include information about every file that is transferred since the last progress message was published. Increasing this value increases the size of the progress messages, which might affect performance.	50
V7.5.0.2 maxInlineFileSize	For single file-to-file, or file-to-message transfers, the maximum file size (in bytes) that can be automatically included in the initial transfer request message. You can use this property to improve the speed of your transfers, but if you set the file size to too large a value, this might degrade performance. A suggested initial size for this property is 100 KB but you are recommended to thoroughly test different values until you find the best file size for your system.	0
Input/output properties:		
doNotUseTempOutputFile	By default, the agent writes to a temporary file at the destination and renames this temporary file to the required file name after the file transfer is complete. Setting this value to true causes the agent to write directly to the final destination file. On IBM 4690, do not set this property to true: writing directly to the final file at the destination is not supported on 4690 OS. The value of this property for a transfer is defined by the destination agent.	false
enableMandatoryLocking	When accessing normal files WebSphere MQ Managed File Transfer takes a shared lock for reading and an exclusive lock for writing. On UNIX type platforms, file locking is fulfilled across processes. However, on Windows file locking is advisory only. When this property is set to true, WebSphere MQ Managed File Transfer enforces file locking. On Windows this means that if another application has a file open, monitoring of that file does not trigger until the file is closed. WebSphere MQ Managed File Transfer transfers involving that file fail. For UNIX type platforms, setting this property has no effect. The value of this property can be `true` or `false`.	false
ioIdleThreadTimeout	Time in milliseconds for a file system input/output thread to remain idle before the thread shuts down.	10000
ioQueueDepth	The maximum number of input/output requests to queue up.	10

Table 34. Advanced agent properties (continued)

Property name	Description	Default value
ioThreadPoolSize	Maximum number of file system input/output threads available. Typically each transfer uses its own file system input/output thread, but if the number of concurrent transfers exceeds this limit, the file system input/output threads are shared between transfers. If you think you are likely to regularly have more concurrent transfers in progress than the ioThreadPoolSize value, you might see an improvement by increasing this value, so that each transfer has its own file system input/output thread.	10
textReplacementCharacterSequence	For text mode transfer, if any of the data bytes cannot be converted from the source code page to the destination code page, the default behavior is for the file transfer to fail. Set this property to allow the transfer to complete successfully by inserting the specified character value. This property value is a single character. Typically, a question mark (?) is used for any unmappable characters. For example, use this format textReplacementCharacterSequence=? where the question mark (?) is the replacement character. You cannot use a white space character as a replacement character.	None
Multi-channel support:		
agentMultipleChannelsEnabled	Setting this property to true enables a WebSphere MQ Managed File Transfer agent to send transfer data messages across multiple IBM WebSphere MQ channels. In some scenarios, setting this property might improve performance. However, only enable multi-channel support only if there is a demonstrable performance benefit. Only messages that are put to the SYSTEM.FTE.DATA.*destinationAgentName* queue are sent across multiple channels. The behavior for all other messages remains unchanged. When you set this property to true, you must also complete the IBM WebSphere MQ configuration steps in one of the following topics to enable multi-channel support: • "Configuring multiple IBM WebSphere MQ channels in a cluster" on page 564 • "Configuring multiple IBM WebSphere MQ channels in a non-clustered configuration" on page 564 Additionally, you must also complete the standard IBM WebSphere MQ configuration steps that are required for a WebSphere MQ Managed File Transfer agent, which are detailed in "Configuring WebSphere MQ Managed File Transfer for first use" on page 125. The value of this property can be true or false.	false
agentMessageBatchSize	When configured with multiple channels, a source agent sends data messages for a transfer across each channel on a round-robin basis. This property controls the number of messages that are sent down each channel at a time.	5
Multi-instance queue manager properties:		
agentQMgrStandby	The host name and the port number that are used for client connections, in WebSphere MQ CONNAME format, for the standby instance of a multi-instance agent queue manager that is defined by agentQMgr. For example, *host_name(port_number)*	No default
Process controller properties:		
agentQMgrRetryInterval	The interval, in seconds, between checks on the availability of the queue manager by the agent's process controller.	30
maxRestartCount	The maximum number of restarts that can happen within the time interval that is specified by the value of the maxRestartInterval property. When this value is exceeded the agent's process controller stops restarting the agent, and instead makes an action that is based on the value of the maxRestartDelay property.	4
maxRestartInterval	The interval, in seconds, that the agent's process controller measures agent restarts over. If the number of restarts in this interval exceeds the value of the maxRestartCount property, the agent's process controller stops restarting the agent. Instead the agent's process controller makes an action that is based on the value of the maxRestartDelay property.	120

Table 34. Advanced agent properties (continued)

Property name	Description	Default value
maxRestartDelay	Determines the behavior of the agent's process controller when the rate of agent restarts exceeds the value of the maxRestartCount and maxRestartInterval properties. If you specify a value less than or equal to zero, the agent's process controller is stopped. If you specify a value greater than zero, it is the number of seconds to wait before the restart history information held by the agent's process controller is reset and the agent is restarted.	-1
processControllerPollingInterval	The interval, in seconds, between each attempt that an IBM 4690 agent controller makes to check the status that is provided by its associated agent. This interval begins after the last successful status message is received from the agent.	30
processControllerPollingTimeout	The time, in seconds, that an IBM 4690 agent controller waits for an expected status update from an agent before it assumes the agent is no longer responsive. When the time defined by the processControllerPollingInterval property elapses, the agent controller starts to check for a status update. If a status update does not arrive in the time period that is defined by this property the agent is assumed to be unresponsive.	10
useProcessController	By default an IBM 4690 agent attempts to start a process controller to monitor its execution. It is recommended this value is set to true, but if the cost of memory usage is too high it can be set to false. The agent will not attempt to start a process controller, but this may reduce the agent reliability.	true
Protocol bridge properties:		
protocolBridgeCredentialConfiguration	The value of this property is passed in as a string to the initialize() method of the exit classes that are specified by protocolBridgeCredentialExitClasses.	null
protocolBridgeCredentialExitClasses	Specifies a comma-separated list of classes that implement a protocol bridge credential user exit routine. For more information, see "Mapping credentials for a file server using exit classes" on page 259.	No default.
protocolBridgeDataTimeout	The timeout in milliseconds that the protocol bridge agent waits to either establish a data connection to an FTP server or to receive data from an FTP server over a connection that is already established. If you set this property to a value of 0, the protocol bridge agent waits indefinitely. If the timeout elapses, the protocol bridge agent closes any existing data connections to the FTP server and attempts to establish a new data connection before resuming the current transfer. If the attempt to establish the new data connection fails, the current transfer also fails.	0
V7.5.0.2 protocolBridgeLogoutBeforeDisconnect	Specifies whether the protocol bridge agent logs the user out of the file server before closing the FTP session and disconnecting. If you set this property to true, the protocol bridge agent issues an FTP **QUIT** command to the file server.	false
protocolBridgePropertiesConfiguration	Passed as one of the bridge properties to the initialize() method of the exit classes that are specified by the protocolBridgeServerPropertiesExitClasses property.	No default
protocolBridgePropertiesExitClasses	Specifies a comma-separated list of classes that implement a protocol bridge server properties user exit routine. For more information, see "Looking up protocol file server properties by using exit classes (ProtocolBridgePropertiesExit2)" on page 253.	No default
Queue properties:		
publicationMDUser	The MQMD user ID to associate with messages sent to be published by the coordination queue manager. If you do not set this property, the MQMD user ID is set based on the WebSphere MQ rules for setting MQMD user IDs.	No default
Resource monitoring properties:		
monitorFilepathPlatformSeparator	Specifies whether to use platform-specific path separators within the $FILEPATH variable. A value of true uses platform-specific path separators. A value of false uses a UNIX style forward slash (/) path separator on all platforms.	true
monitorMaxResourcesInPoll	Specifies the maximum number of monitored resources to be triggered in each poll interval. For example, if you specify a monitor pattern of *.txt, a poll interval of 10 seconds, and set the monitorMaxResourcesInPoll property to 10, the monitorMaxResourcesInPoll property limits the agent to trigger on a maximum of 10 matches for each poll interval. Matching resources beyond the limit of 10 are triggered in later poll intervals. In addition, you can use the monitorMaxResourcesInPoll property in combination with a matching **-bs** parameter on the **fteCreateMonitor** command, for example, to restrict each poll interval to triggering one transfer only. A value less than or equal to zero means that the number of monitor resources that are triggered in a polling interval is unlimited.	-1

Table 34. Advanced agent properties (continued)

Property name	Description	Default value
monitorReportTriggerFail	Specifies whether failure conditions, in the environment and configuration, that are detected in the monitor are reported as a log message to the SYSTEM.FTE topic. A value of true logs messages. A value of false does not log messages.	true
monitorReportTriggerNotSatisfied	Specifies whether a non-satisfied trigger sends a log message to the SYSTEM.FTE topic that contains the details. A value of true logs messages. A value of false does not log messages.	false
monitorReportTriggerSatisfied	Specifies whether a satisfied trigger sends a log message to the SYSTEM.FTE topic that contains the details. A value of true logs messages. A value of false does not log messages.	false
monitorSilenceOnTriggerFailure	The number of consecutive failures of the resource monitor trigger before the failures are no longer reported.	5
monitorStopOnInternalFailure	The number of consecutive internal FFDC conditions of the resource monitor before the monitor changes its state to stop.	10
Root directory properties:		
commandPath	Specifies the set of paths that commands can be called by, using one of the following methods: • Agent Ant call, filecopy, or filemove tasks • In an XML message passed to an agent, using one of the supported WebSphere MQ Managed File Transfer agent command XML schemas (for example, managedCall or managedTransfer). For information about the valid syntax of the value of the commandPath property, see "The commandPath property" on page 414. Take extreme care when you set this property because any command in one of the specified commandPaths can effectively be called from a remote client system that is able to send commands to the agent. For this reason, by default, when you specify a commandPath, sandboxing is enabled so that all commandPath directories are automatically denied access for a transfer. You can set the sandboxRoot property to override this default behavior, but you are not recommended to do so, because this override effectively enables a client to transfer any command to the agent system and call the command. For WebSphere MQ V7.5, the value of this property can contain environment variables.	None - no commands can be called
additionalWildcardSandboxChecking	Specifies whether additional checks are to be made on wildcard transfers for an agent that has been configured with a user or agent sandbox in order to restrict the locations that the agent can transfer files to and from. When this property is set to true, the additional checking is enabled. If a transfer request attempts to read a location that is outside of the defined sandbox for file matching of the wildcard, the transfer fails. If there are multiple transfers within one transfer request, and one of these transfers fails due to it attempting to read a location outside of the sandbox, the entire transfer fails. If checking fails, the reason for failure is given in an error messages (see "Additional checks for wildcard transfers" on page 108). If the property is omitted or set to false then no additional checks are made on wildcard transfers.	None
sandboxRoot	Specifies the set of root paths to include and exclude when you use sandboxing. See Working in a sandbox for information about this feature. Separate paths with a platform-specific path separator. Prefix paths with an exclamation point (!) character to denote paths as excluded from the sandbox. This feature is useful if you want to exclude a subdirectory under an included root path. The sandboxRoot property is not supported on protocol bridge agents. You cannot specify the sandboxRoot property and the userSandboxes property together. For WebSphere MQ V7.5, the value of this property can contain environment variables.	None - no sandbox

Table 34. Advanced agent properties (continued)

Property name	Description	Default value
transferRoot	Default root directory for relative paths that are specified to the agent. For WebSphere MQ V7.5, the value of this property can contain environment variables. On IBM 4690, there is no default directory for this property. Therefore, if you want to use relative paths, you must set the value of this property to a directory path. If you do not set this property, file transfers to or from a 4690 agent that specify a relative source or destination path, fail with an error message.	The home directory for the user that started the agent process. No default directory for IBM 4690.
transferRootHLQ	Default HLQ (user ID) for non-fully qualified data sets specified to the agent	The user name of the user that started the agent process.
userSandboxes	Restrict the area of the file system that files can be transferred to and from based on the MQMD user name of the user that requests the transfer. For more information, see "Working with user sandboxes" on page 105. The userSandboxes property is not supported on protocol bridge agents. You cannot specify the sandboxRoot property and the userSandboxes property together.	false
Scheduler property:		
maxSchedulerRunDelay	The maximum interval, in minutes, that the agent waits to check for scheduled transfers. Specify a positive integer to enable this property. For more information about why you might want to use this property, see "What to do if your scheduled transfer does not run or is delayed" on page 356. Because the agent might be reading a command from its command queue at the time that scheduled transfers are due to run, there may be an additional delay before the scheduled transfers are started. In this case, the scheduler runs immediately after that command completes.	-1
Security properties:		
authorityChecking	Specifies whether the security features described in "User authorities on WebSphere MQ Managed File Transfer actions" on page 409 are enabled.	false
logAuthorityChecks	The level of authority check logging that is recorded in the agent event log in the `output0.log` file. The values that this property can have are `None` or `Failures` or `All`.	None
userIdForClientConnect	The user ID that gets flowed through the client connections to WebSphere MQ. If *java* is specified, the user name reported by the JVM is flowed as part of the WebSphere MQ connection request. The values that this property can have are `None` or `java`.	None
Timeout properties:		
maxTransferNegotiationTime	The maximum time in milliseconds that a transfer waits for a destination agent to complete negotiation. If negotiation does not complete in this time, the transfer is put into a resynchronization state and allows another transfer, when available, to run. In scenarios where the source or destination agent is under heavy load it is possible that the default value is too low for the agent to respond quickly enough to the negotiation request. This is most likely when a source agent has a large number of resource monitors defined or when its resource monitors are monitoring directories that contain large numbers of files. However, it can also occur when a large number of transfer requests is submitted to an agent. Increasing the value of this property to 200,000 or more may be necessary in such scenarios.	30 000
recoverableTransferRetryInterval	The time to wait in milliseconds between detecting a recoverable transfer error and attempting to resume the transfer.	60 000
senderTransferRetryInterval	The time in milliseconds to wait until a rejected transfer is retried because the destination is already running the maximum number of transfers. Minimum value is 1000.	30 000

Table 34. Advanced agent properties (continued)

Property name	Description	Default value
transferAckTimeout	Timeout in milliseconds that a transfer waits for acknowledgment or data from the other end before a retry is issued. This is an advanced property and for most WebSphere MQ Managed File Transfer configurations it is not necessary to modify its value. Acknowledgments are sent from the receiving agent to the sending agent whenever a complete window of data is received. For bandwidth-constrained or unreliable networks and large agentWindowSize and agentChunkSize settings, it is possible that the default is not long enough. This can cause unnecessary retransfer of data between the agents. Therefore increasing this value might be beneficial and may reduce the likelihood of a transfer going into recovery mode because of a slow network.	60 000
transferAckTimeoutRetries	Maximum number of acknowledgment retries for a transfer without a response before the agent gives up and moves the transfer into a recovery state	5
xmlConfigReloadInterval	The interval in seconds between the agent reloading XML configuration files during runtime. To prevent the agent from reloading XML configuration files during runtime set this property to -1. The following XML configuration files are affected by this property: • ConnectDirectCredentials.xml • ConnectDirectNodeProperties.xml • ConnectDirectProcessDefinitions.xml • ProtocolBridgeCredentials.xml • ProtocolBridgeProperties.xml • UserSandboxes.xml	30
Tracing and logging properties:		
javaCoreTriggerFile	The full path to a file location that the agent monitors. If you create or update a file at this location, the agent triggers a Javacore file. A separate thread polls this file every 30 seconds to check whether the file has been created or updated. If the file has been created or updated since the last poll, the agent generates a Javacore file in one of the following directories: ▶**V 7.5.0.1**◀ • UNIX: *MQ_DATA_PATH*/mqft/logs/*coordination_qmgr_name*/agents/*agent_name* • Linux: *MQ_DATA_PATH*/mqft/logs/*coordination_qmgr_name*/agents/*agent_name* • Windows: *MQ_DATA_PATH*\mqft\logs*coordination_qmgr_name*\agents*agent_name* • IBM 4690: f:\adxetc\java\core When you specify this property, the agent outputs the following message at startup: BFGAG0092I The <insert_0> file will be used to request JVM diagnostic information. For WebSphere MQ V7.5, the value of this property can contain environment variables.	None
trace	Trace specification when agent is to be run with trace enabled at agent start. The trace specification is a comma-separated list of classes, the equals character, and a trace level. For example, com.ibm.wmqfte.agent.Agent,com.ibm.wmqfte.commandhandler=all. You can specify multiple trace specifications in a colon-separated list. For example, com.ibm.wmqfte.agent.Agent=all:com.ibm.wmqfte.commandhandler=moderate.	None
outputLogFiles	The total number of output.log files to keep. This value applies to an agent's process controller and the agent itself.	5
outputLogSize	The maximum size in MB of each output.log file before output wraps onto the next file. This value applies to an agent's process controller and the agent itself.	1
outputLogEncoding	The character encoding that the agent uses when it writes to the output.log file.	The default character encoding of the platform that the agent is running on.
traceFiles	The total number of trace files to keep. This value applies to an agent's process controller as well and the agent itself.	5

Table 34. Advanced agent properties (continued)

Property name	Description	Default value
traceSize	The maximum size in MB of each trace file before trace wraps onto the next file. This value applies to an agent's process controller and the agent itself.	20
traceMaxBytes	The limit to the amount of message data that is output in the trace file.	4096 bytes
logTransferRecovery	When this property is set to a value of true, whenever a transfer enters recovery diagnostic events are reported to the agent's event log.	Before IBM WebSphere MQ Version 7.5.0, Fix Pack 6, the default value is false. ▶V 7.5.0.6◀ From IBM WebSphere MQ Version 7.5.0, Fix Pack 6, the default value is true.
logCapture	Captures transfer request messages that are submitted to this agent and log messages that are published by the agent to the coordination queue manager. These captured messages can be helpful when debugging transfer problems. Captured messages are stored in files in the agent log directory called capture?.log. The ? is a numeric value. The file that contains the number 0 holds the newest captured messages.	false
logCaptureFileSize	Defines the maximum size of a capture file in megabytes.	10
logCaptureFiles	Defines the maximum number of capture files that are retained before the oldest file is discarded.	10
logCaptureFilter	A Java regular expression that the agent uses to match the topic name of the message. Only those messages that match the regular expression are captured.	.* (match all)
Transfer limit properties:		
maxDestinationTransfers	The maximum number of concurrent transfers that the destination agent processes at any point in time. Each transfer request that is submitted to an agent counts against this total regardless of the number of files that are transferred to satisfy the request. This means that a transfer request that transfers a single file counts in the same way as a transfer request that transfers 10 files. The agent queues transfers when the destination agent reaches the limit that is specified by the maxDestinationTransfers property. If the sum of the following agent property values: maxSourceTransfers + maxDestinationTransfers + maxQueuedTransfers exceeds the value of the MAXDEPTH setting of the state store queue (SYSTEM.FTE.STATE.*agent name*), the agent does not start.	25 (for all agents except Connect:Direct bridge agents) 5 (for Connect:Direct bridge agents)
maxFilesForTransfer	The maximum number of transfer items that are allowed for a single managed transfer. If a managed transfer contains more items than the value of maxFilesForTransfer, the managed transfer fails and no transfer items are processed. Setting this property prevents you from accidentally transferring too many files because of a bad transfer request, for example, if a user accidentally specifies the transfer of the root directory / on a UNIX system.	5000
maxSourceTransfers	The maximum number of concurrent transfers that the source agent processes at any point in time. Each transfer request that is submitted to an agent counts against this total regardless of the number of files that are transferred to satisfy the request. This means that a transfer request that transfers a single file counts in the same way as a transfer request that transfers 10 files. The source agent queues transfers when the destination agent reaches the limit that is specified by the maxSourceTransfers property. If the sum of the following agent property values: maxSourceTransfers + maxDestinationTransfers + maxQueuedTransfers exceeds the value of the MAXDEPTH setting of the state store queue (SYSTEM.FTE.STATE.*agent name*), the agent does not start.	25 (for all agents except Connect:Direct bridge agents) 5 (for Connect:Direct bridge agents)

Table 34. Advanced agent properties (continued)

Property name	Description	Default value
maxQueuedTransfers	The maximum number of pending transfers that can be queued by an agent until the agent rejects a new transfer request. You can set this property so that despite of the limits of maxDestinationTransfers and maxSourceTransfers being met or exceeded, any new transfer requests that you make now are accepted, queued and then carried out later. The order that queued transfer requests are processed in is a factor of their priority and how long they have been queued. Old and high priority pending transfers are selected first. Transfers with a low priority that have been on the queue for a long time are selected in preference to newer, higher priority transfers. If the sum of the following agent property values: maxSourceTransfers + maxDestinationTransfers + maxQueuedTransfers exceeds the value of the MAXDEPTH setting of the state store queue (SYSTEM.FTE.STATE.*agent name*), the agent does not start.	1000
User exit routine properties:		
agentForceConsistentPathDelimiters	Force the path delimiter in the source file and destination file information that is provided to the transfer exits to be the UNIX style: forward slash (/). Valid options are `true` and `false`.	false
destinationTransferEndExitClasses	Specifies a comma-separated list of classes that implement a destination transfer end user exit routine.	No default
destinationTransferStartExitClasses	Specifies a comma-separated list of classes that implement a destination transfer start user exit routine.	No default
exitClassPath	Specifies a platform-specific, character-delimited list of directories that act as the class path for user exit routines. The agent exit directory is searched before any entries in this class path.	Agent's `exits` directory
exitNativeLibraryPath	Specifies a platform-specific, character-delimited list of directories that act as the native library path for user exit routines.	Agent's `exits` directory
ioMaxRecordLength	The maximum record length, in bytes, that can be supported for a record-oriented file. WebSphere MQ Managed File Transfer can support writing to record-oriented files with any record length. However, large record lengths might cause out-of-memory errors, so to avoid these errors the maximum record length is restricted by default to 64 K. When reading from record-oriented files an entire record must fit into a single transfer chunk, therefore the record length is additionally limited by the transfer chunk size. This property is used only for I/O user exit record-oriented files.	64 KB
monitorExitClasses	Specifies a comma-separated list of classes that implement a monitor exit routine. For more information, see "Resource monitor user exits" on page 923.	No default
protocolBridgeCredentialExitClasses	Specifies a comma-separated list of classes that implement a protocol bridge credential user exit routine. For more information, see "Mapping credentials for a file server using exit classes" on page 259.	No default.
sourceTransferEndExitClasses	Specifies a comma-separated list of classes that implement a source transfer end exit routine.	No default
sourceTransferStartExitClasses	Specifies a comma-separated list of classes that implement a source transfer start exit routine.	No default
IOExitClasses	Specifies a comma-separated list of classes that implement an I/O user exit routine. List only the classes that implement the IOExit interface, that is, do not list classes that implement the other I/O user exit interfaces, for example IOExitResourcePath and IOExitChannel. For more information, see "Using WebSphere MQ Managed File Transfer transfer I/O user exits" on page 333.	No default.
webGatewayName	Required. The name of the Web Gateway that you are deploying. The name of the Web Gateway is not case-sensitive and must conform to the WebSphere MQ object naming conventions. For more information, see "Object naming conventions for WebSphere MQ Managed File Transfer" on page 664.	No default.
WebSphere MQ client compression:		

Table 34. Advanced agent properties (continued)

Property name	Description	Default value
agentDataCompression	This property is supported for client connections only. A comma-separated list of the compression types for the transfer of file data to negotiate with the remote IBM WebSphere MQ server. You can find information about these compression types in the following topic: Message data compression list. The values are checked for validity and then passed through in order of appearance as properties to the agent client channel. The IBM WebSphere MQ client then handles negotiation between this client channel and the remote server channel to find the matching lowest common denominator between the compression properties on the two channels. If no match is found, MQCOMPRESS_NONE is always selected.	MQCOMPRESS_NONE
agentHeaderCompression	This property is supported for client connections only. A comma-separated list of the compression types for the transfer of header data to negotiate with the remote IBM WebSphere MQ server. Accepted values are MQCOMPRESS_NONE or MQCOMPRESS_SYSTEM. You can find information about these compression types in the following topic: Message header compression list. The values are checked for validity and then passed through in order of appearance as properties to the agent client channel. The IBM WebSphere MQ client then handles negotiation between this client channel and the remote server channel to find the matching lowest common denominator between the compression properties on the two channels. If no match is found, MQCOMPRESS_NONE is always selected.	MQCOMPRESS_NONE

Related concepts:

"Configuration options" on page 124
WebSphere MQ Managed File Transfer provides a set of properties files that contain key information about your setup and are required for operation. These properties files are in the configuration directory that you defined when you installed the product.

Related reference:

"Java system properties" on page 597
A number of WebSphere MQ Managed File Transfer command and agent properties must be defined as Java system properties, because they define configuration for early function that is unable to use the command or agent properties mechanism.

"SSL properties" on page 598
Use SSL withWebSphere MQ and WebSphere MQ Managed File Transfer to prevent unauthorized connections between agents and queue managers, and to encrypt message traffic between agents and queue managers.

"fteCreateAgent (create a WebSphere MQ Managed File Transfer agent)" on page 429
The **fteCreateAgent** command creates an agent and its associated configuration.

"fteCreateBridgeAgent (create and configure WebSphere MQ Managed File Transfer protocol bridge agent)" on page 432
The **fteCreateBridgeAgent** command creates a protocol bridge agent and its associated configuration. Create a protocol bridge agent for each file server that you want to send files to and receive files from.

"fteCreateCDAgent (create a Connect:Direct bridge agent)" on page 437
The fteCreateCDAgent command creates a WebSphere MQ Managed File Transfer agent and its associated configuration for use with the Connect:Direct bridge.

"fteCreateWebAgent (create a WebSphere MQ Managed File Transfer web agent)" on page 480
The **fteCreateWebAgent** command creates an agent and its associated configuration for use with the Web Gateway. This command is provided with WebSphere MQ Managed File Transfer Server.

"Environment variables in WebSphere MQ Managed File Transfer properties" on page 537
In WebSphere MQ V7.5, it is possible for environment variables to be used in Managed File Transfer properties that represent file or directory locations. This allows the locations of files or directories used when running parts of the product, to vary depending on environment changes. For example, which user is running the process.

Configuring multiple IBM WebSphere MQ channels in a cluster

If you want to use IBM WebSphere MQ multi-channel support in a clustered configuration, first set the agentMultipleChannelsEnabled property to `true` and then complete the steps in this topic.

About this task

In a cluster, multi-channel support is enabled by IBM WebSphere MQ definitions on the queue manager of the destination agent only.

You must complete the steps in this topic in addition to the standard IBM WebSphere MQ configuration steps required for a WebSphere MQ Managed File Transfer agent, which are listed in "Configuring WebSphere MQ Managed File Transfer for first use" on page 125.

The following configuration examples use **runmqsc** commands.

Procedure

1. Define a cluster-receiver channel for each channel that you want to use. For example, if you are using two channels:

   ```
   DEFINE CHANNEL(TO.DESTQMGRNAME_1) CHLTYPE(CLUSRCVR) CLUSTER(MFTCLUSTER)
   DEFINE CHANNEL(TO.DESTQMGRNAME_2) CHLTYPE(CLUSRCVR) CLUSTER(MFTCLUSTER)
   ```

   where:
   - *DESTQMGRNAME* is the name of the queue manager of the destination agent.
   - *MFTCLUSTER* is the name of the IBM WebSphere MQ cluster.

   You are recommended to use the *MFTCLUSTER.DESTMGRNAME*_n naming convention for channels, but this convention is not mandatory.

2. Define a queue manager alias corresponding to each channel. For example:

   ```
   DEFINE QREMOTE(SYSTEM.FTE.DESTQMGRNAME_1) RQMNAME(DESTQMGRNAME) CLUSTER(MFTCLUSTER)
   DEFINE QREMOTE(SYSTEM.FTE.DESTQMGRNAME_2) RQMNAME(DESTQMGRNAME) CLUSTER(MFTCLUSTER)
   ```

   You must use the SYSTEM.FTE.*DESTQMGRNAME*_n naming convention for queue manager aliases because the sending agent searches for queue manager aliases of this format. The numbers that you use for *n* must start at 1 and be consecutive. You must make the definitions cluster-wide so that they are available on the source agent's queue manager.

   For both the source agent and destination agent to correctly determine the number of queue manager aliases, do **not** define a default XMITQ for the queue manager.

Related concepts:

"Configuring WebSphere MQ Managed File Transfer for first use" on page 125
You must perform some configuration tasks for WebSphere MQ Managed File Transfer agents and queue managers once, the first time you want to use them.

Related reference:

"The agent.properties file" on page 548
Each agent has its own properties file, `agent.properties`, that must contain the information that an agent uses to connect to its queue manager. The `agent.properties` file can also contain properties that alter the behavior of the agent.

Configuring multiple IBM WebSphere MQ channels in a non-clustered configuration

If you want to use IBM WebSphere MQ multi-channel support in a non-clustered configuration, first set the agentMultipleChannelsEnabled property to `true` and then complete the steps in this topic.

About this task

In a non-clustered configuration, multi-channel support is enabled by IBM WebSphere MQ definitions on the queue manager of both the source agent and the destination agent.

You must complete the steps in this topic in addition to the standard IBM WebSphere MQ configuration steps required for a WebSphere MQ Managed File Transfer agent, which are listed in "Configuring WebSphere MQ Managed File Transfer for first use" on page 125.

The following steps assume that sender-receiver channels are being used to communicate between the source and destination queue managers.

The following configuration examples use **runmqsc** commands.

Procedure

1. On the destination agent's queue manager, define a receiver channel for each channel that you want to use. For example, if you are using two channels:

   ```
   DEFINE CHANNEL(TO.DESTQMGRNAME_1) CHLTYPE(RCVR) TRPTYPE(TCP)
   DEFINE CHANNEL(TO.DESTQMGRNAME_2) CHLTYPE(RCVR) TRPTYPE(TCP)
   ```

   where: DESTQMGRNAME is the name of the queue manager of the destination agent.

   You are recommended to use the TO.DESTMGRNAME_n naming convention for channels, but this convention is not mandatory. The receiver channel names must match the corresponding sender channels on the source agent's queue manager.

2. On the source agent's queue manager, define a transmission queue for each channel that you want to use. For example, if you are using two channels:

   ```
   DEFINE QLOCAL(DESTQMGRNAME_1) USAGE(XMITQ)
   DEFINE QLOCAL(DESTQMGRNAME_2) USAGE(XMITQ)
   ```

   You are recommended to use the DESTMGRNAME_n naming convention for transmission queues, but this convention is not mandatory. The transmission queues you define are referenced from the sender channel definitions and the queue manager alias definitions in the following steps.

3. On the source agent's queue manager, define a sender channel for each channel that you want to use. For example, if you are using two channels:

   ```
   DEFINE CHANNEL(TO.DESTQMGRNAME_1) CHLTYPE(SDR) TRPTYPE(TCP) CONNAME(DESTHOST:port)
   XMITQ(DESTQMGRNAME_1)
   DEFINE CHANNEL(TO.DESTQMGRNAME_2) CHLTYPE(SDR) TRPTYPE(TCP) CONNAME(DESTHOST:port)
   XMITQ(DESTQMGRNAME_2)
   ```

   You are recommended to use the TO.DESTMGRNAME_n naming convention for the channels, but this convention is not mandatory. The sender channel names must match the corresponding receiver channels on the destination agent's queue manager.

4. On the source agent's queue manager, define a queue manager alias corresponding to each channel. For example:

   ```
   DEFINE QREMOTE(SYSTEM.FTE.DESTQMGRNAME_1) RQMNAME(DESTQMGRNAME) XMITQ(DESTQMGRNAME_1)
   DEFINE QREMOTE(SYSTEM.FTE.DESTQMGRNAME_2) RQMNAME(DESTQMGRNAME) XMITQ(DESTQMGRNAME_2)
   ```

   You must use the SYSTEM.FTE.DESTQMGRNAME_n naming convention for the queue manager aliases because the sending agent searches for queues manager aliases of this format. The numbers that you use for *n* must start at 1 and be consecutive.

   For the agent to correctly determine the number of queue manager aliases, do **not** define a default XMITQ for the queue manager.

Related concepts:

"Configuring WebSphere MQ Managed File Transfer for first use" on page 125

You must perform some configuration tasks for WebSphere MQ Managed File Transfer agents and queue managers once, the first time you want to use them.

Related reference:

"The agent.properties file" on page 548

Each agent has its own properties file, `agent.properties`, that must contain the information that an agent uses to connect to its queue manager. The `agent.properties` file can also contain properties that alter the behavior of the agent.

Supported user-defined metadata keys

▶ V 7.5.0.1

When the agent property enableUserMetadataOptions is set to a value of true, the following user-defined metadata keys are supported when specified to a new transfer request.

Table 35. Metadata keys

Key name	Description	Default value
com.ibm.wmqfte.insertRecordLineSeparator	For text transfers. When this key is set to true, specifies that when reading record-oriented files, such as z/OS data sets, line separators are to be inserted between records. When this key is set to false, specifies that when reading record-oriented files, line separators are not to be inserted between records.	true
com.ibm.wmqfte.newRecordOnLineSeparator	For text transfers. When this key is set to true, specifies that when writing to record-oriented files, such as z/OS data sets, that line separators indicate a new record and are not written as part of the data. When this key is set to false, specifies that when writing to record-oriented files that line separators are to be treated like any other character (that is, no record breaks).	true
com.ibm.wmqfte.convertLineSeparators	For text transfers. Specifies whether the line separator sequences CRLF and LF are converted to the required line separator sequence for the destination. This conversion currently only takes effect for the following cases: 1. If the user-defined metadata key com.ibm.wmqfte.newRecordOnLineSeparator is set to false and the transfer is to a record-oriented file. 2. If the user-defined metadata key com.ibm.wmqfte.com.ibm.wmqfte.insertRecordLineSeparator is set to false and the transfer is from a record-oriented file.	true

Related information:

Table 34 on page 550

fteCreateTransfer -md parameter

Additional agent configuration files

In addition to the `agent.properties` file, the agent can have a number of XML configuration files in its configuration directory.

Configuration files

The following XML configuration files can be used to specify additional information used by the agent:

ProtocolBridgeCredentials.xml
>
> If your agent is a protocol bridge agent, you can use this file to specify the credentials to use to log in to the FTP or SFTP server that the agent connects to.

ProtocolBridgeProperties.xml
>
> If your agent is a protocol bridge agent, you can use this file to define the properties of non-default protocol file servers that the agent connects to. The **fteCreateBridgeAgent** command creates a default protocol file server in this file for you.

ConnectDirectCredentials.xml
>
> If your agent is a Connect:Direct bridge agent, you can use this file to specify the credentials to use to connect to the Connect:Direct nodes involved in a transfer.

ConnectDirectNodeProperties.xml
>
> If your agent is a Connect:Direct bridge agent, you can use this file to specify the operating system information about the Connect:Direct nodes involved in a transfer.

ConnectDirectProcessDefinition.xml
>
> If your agent is a Connect:Direct bridge agent, you can use this file to specify the user-defined Connect:Direct processes to call as part of a file transfer.

UserSandboxes.xml
>
> You can use this file to specify which areas of the file system the agent can read from or write to.

Updating the configuration files

Unlike the `agent.properties` file, you can update the XML configuration files and have the agent pick up the changes without having to restart the agent.

When you submit a transfer, if it has been more than 10 second since the last time the agent checked the XML configuration file, the agent checks the last modified time of the XML configuration file. If the XML configuration file has been modified since the last time the agent read the file, the agent reads the file again. If the contents of the file are valid when compared to the XML schema, the agent updates its information. If the contents of the file are not valid, the agent uses the information from the previous version of the file and writes a message to the `outpu0.log` file.

Protocol bridge credentials file format

The `ProtocolBridgeCredentials.xml` file in the agent configuration directory defines the user names and credential information that the protocol bridge agent uses to authorize itself with the protocol server.

The `ProtocolBridgeCredentials.xml` file must conform to the `ProtocolBridgeCredentials.xsd` schema. The `ProtocolBridgeCredentials.xsd` schema document is located in the *MQ_INSTALLATION_PATH*/mqft/samples/schema directory of the MQMFT installation. Users are responsible for manually creating the `ProtocolBridgeCredentials.xml` file, it is no longer created by the **fteCreateBridgeAgent** command. Sample files are available in the *MQ_INSTALLATION_PATH*/mqft/samples directory of the MQMFT installation.

V7.5 introduces a new <agent> element that contains the <server> or <serverHost> element for the named agent.

The `ProtocolBridgeCredentials.xml` file is periodically reloaded by the agent and any valid changes to the file will affect the behavior of the agent. The default reload interval is 30 seconds. This interval can be changed by specifying the agent property `xmlConfigReloadInterval` in the `agent.properties` file.

Schema - V7.5

The following schema describes which elements are valid in the `ProtocolBridgeCredentials.xml` file for V7.5

```
<schema targetNamespace="http://wmqfte.ibm.com/ProtocolBridgeCredentials" elementFormDefault="qualified"
             xmlns="http://www.w3.org/2001/XMLSchema" xmlns:tns="http://wmqfte.ibm.com/ProtocolBridgeCredentials">

<!--
    <?xml version="1.0" encoding="UTF-8"?>
      <tns:credentials xmlns:tns="http://wmqfte.ibm.com/ProtocolBridgeCredentials"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://wmqfte.ibm.com/ProtocolBridgeCredentials
    ProtocolBridgeCredentials.xsd ">
        <tns:agent name="agent1">
          <tns:serverHost name="myserver">
            <tns:user name="fred" serverPassword="pwd" serverUserId="bill"/>
            <tns:user name="jane" serverUserId="june" hostKey="1F:2e:f3">
              <tns:privateKey associationName="test" keyPassword="pwd2">
                .... private key ...
              </tns:privateKey>
            </tns:user>
          </tns:serverHost>
        </tns:agent>

        <tns:agent name="agent2">
          <tns:server name="server*" pattern="wildcard">
            <tns:user name="fred" serverPassword="pwd" serverUserId="bill"/>
            <tns:user name="jane" serverUserId="june" hostKey="1F:2e:f3">
              <tns:privateKey associationName="test" keyPassword="pwd2">
                .... private key ...
              </tns:privateKey>
            </tns:user>
          </tns:server>
        </tns:agent>

        <tns:agent name="agent3">
          <tns:serverHost name="ftpsServer"
                          keyStorePassword="keypass"
                          trustStorePassword="trustpass">
            <tns:user name="fred" serverPassword="pwd" serverUserId="bill"/>
          </tns:serverHost>
        </tns:agent>

      </tns:credentials>
-->

   <element name="credentials" type="tns:credentialsType"/>

   <complexType name="credentialsType">
    <sequence>
     <element name="agent" type="tns:agentType" minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
   </complexType>

   <complexType name="agentType">
    <choice minOccurs="0" maxOccurs="1">
  <element name="serverHost" type="tns:serverHostType" minOccurs="0" maxOccurs="unbounded"/>
  <element name="server" type="tns:serverType" minOccurs="0" maxOccurs="unbounded"/>
 </choice>
 <attribute name="name" type="string" use="required"/>
   </complexType>

   <complexType name="serverHostType">
    <sequence>
     <element ref="tns:user" minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
    <attribute name="name" type="string" use="required"/>
    <attribute name="keyStorePassword" type="string" use="optional"/>
    <attribute name="keyStorePasswordCipher" type="string" use="optional"/>
    <attribute name="trustStorePassword" type="string" use="optional"/>
    <attribute name="trustStorePasswordCipher" type="string" use="optional"/>
   </complexType>

   <complexType name="serverType">
    <sequence>
     <element ref="tns:user" minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
    <attribute name="name" type="string" use="required"/>
    <attribute name="pattern" type="tns:patternType" use="optional" />
    <attribute name="keyStorePassword" type="string" use="optional"/>
    <attribute name="keyStorePasswordCipher" type="string" use="optional"/>
    <attribute name="trustStorePassword" type="string" use="optional"/>
    <attribute name="trustStorePasswordCipher" type="string" use="optional"/>
```

```
      </complexType>

      <element name="user" type="tns:userType"/>

      <complexType name="userType">
       <sequence>
        <element ref="tns:privateKey" minOccurs="0" maxOccurs="unbounded"/>
       </sequence>
       <attribute name="name" type="string" use="required"/>
       <attribute name="serverUserId" type="string" use="optional"/>
       <attribute name="serverUserIdCipher" type="string" use="optional"/>
       <attribute name="serverPassword" type="string" use="optional"/>
       <attribute name="serverPasswordCipher" type="string" use="optional"/>
       <attribute name="hostKey" use="optional">
            <simpleType>
          <restriction base="string">
           <pattern
            value="([a-fA-F0-9]){2}(:([a-fA-F0-9]){2})*">
           </pattern>
          </restriction>
         </simpleType>
        </attribute>
      </complexType>

      <element name="privateKey" type="tns:privateKeyType"/>

      <complexType name="privateKeyType">
       <simpleContent>
            <extension base="string">
          <attribute name="keyPassword" type="string" use="optional"/>
          <attribute name="keyPasswordCipher" type="string" use="optional"/>
          <attribute name="associationName" type="string" use="required"/>
         </extension>
        </simpleContent>
      </complexType>

      <!--
  Determines the type of pattern matching to use.
 -->
 <simpleType name="patternType">
  <restriction base="string">
   <enumeration value="regex" />
   <enumeration value="wildcard" />
  </restriction>
 </simpleType>
</schema>
```

Schema - V7.0.4

The schema for the V7.0.4.1 and later, and the schema for the V7.0.4 and earlier, are available here:
Protocol bridge credentials file format

Understanding the `ProtocolBridgeCredentials.xml` file

The elements and attributes used in the `ProtocolBridgeCredentials.xml` file are described in the
following list.

`<credentials>`
    Group element containing elements that describe the credentials used by a protocol bridge agent to
    connect to a protocol server.

`<agent>`
    Element containing a <server> or <serverHost> definition for a named agent.

`<server>`
    The protocol server that the protocol bridge connects to.

    The <server> element is not supported for V7.0.4 or earlier.

Attribute	Description
name	The name of the protocol server.

Attribute	Description
pattern	If you have used wildcards or regular expressions to specify the pattern of a protocol server name, use either `wildcard` or `regex`.
trustStorePassword or trustStorePasswordCipher	Required when the **<server>** element refers to an FTPS server. The password used to access the truststore. If the **fteObfuscate** command has been used then the cipher version of the attribute must be used.
keyStorePassword or keyStorePasswordCipher	Optional. The password used to access the keystore. If the **fteObfuscate** command has been used then the cipher version of the attribute must be used.

<serverHost>

The host name of the protocol server that the protocol bridge connects to.

The `ProtocolBridgeCredentials.xml` file can either contain <serverHost> elements or <server> elements but you cannot use a mixture of the two different types. When you use <serverHost>, the name is matched against the protocol server's host name. When you use <server>, the name is matched against the protocol server's name (as defined in the `ProtocolBridgeProperties.xml` file).

Attribute	Description
name	The host name or IP address of the protocol server.
trustStorePassword or trustStorePasswordCipher	Required when the **<serverHost>** element refers to an FTPS server. The password used to access the truststore. If the **fteObfuscate** command has been used then the cipher version of the attribute must be used.
keyStorePassword or keyStorePasswordCipher	Optional. The password used to access the keystore. This property is optional unless you set the keyStore attribute, in which case it is required. If the **fteObfuscate** command has been used then the cipher version of the attribute must be used.

<user>

A user mapping from a WebSphere MQ Managed File Transfer user name to a protocol server user name.

Attribute	Description
name	The user name that is used with WebSphere MQ Managed File Transfer.
serverUserId or serverUserIdCipher	The user name that is used with the protocol server. If the **fteObfuscate** command has been used then the cipher version of the attribute must be used.
serverPassword or serverPasswordCipher	The password for the user name used on the protocol server. If the **fteObfuscate** command has been used then the cipher version of the attribute must be used.
hostKey	The server host SSH fingerprint.

<privateKey>

The private key of a user.

Attribute	Description
keyPassword or keyStorePasswordCipher	The password for the private key. If the **fteObfuscate** command has been used then the cipher version of the attribute must be used.
associationName	A name used for trace and logging.

Protocol bridge properties file format

The ProtocolBridgeProperties.xml file in the agent configuration directory defines properties for protocol file servers.

The ProtocolBridgeProperties.xml file must conform to the ProtocolBridgeProperties.xsd schema. The ProtocolBridgeProperties.xsd schema document is located in the *MQ_INSTALLATION_PATH*/mqft/samples/ schema directory of the MQMFT installation. A template file, ProtocolBridgeProperties.xml, is created by the **fteCreateBridgeAgent** command in the agent configuration directory.

The ProtocolBridgeProperties.xml file is periodically reloaded by the agent and any valid changes to the file will affect the behavior of the agent. The default reload interval is 30 seconds. This interval can be changed by specifying the agent property xmlConfigReloadInterval in the agent.properties file.

Schema

The following schema describes the ProtocolBridgeProperties.xml file.

Note: The maxReconnectRetry and reconnectWaitPeriod attributes are not supported on IBM WebSphere MQ V7.5 or on WebSphere MQ Managed File Transfer V7.0.2, or later.

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
@start_non_restricted_prolog@
Version: %Z% %I% %W% %E% %U% [%H% %T%]

Licensed Materials - Property of IBM

5724-H72

Copyright IBM Corp. 2011 2013 All Rights Reserved.

US Government Users Restricted Rights - Use, duplication or
disclosure restricted by GSA ADP Schedule Contract with
IBM Corp.
@end_non_restricted_prolog@
  -->

<!-- This schema defines the format of the ProtocolBridgeProperties.xml file that is located in the agent
 properties directory of a protocol bridge agent. This XML file
 is used to define properties for protocol servers.

 For more information, see the WebSphere MQ product documentation.
  -->

<schema targetNamespace="http://wmqfte.ibm.com/ProtocolBridgeProperties" elementFormDefault="qualified"
 xmlns="http://www.w3.org/2001/XMLSchema" xmlns:tns="http://wmqfte.ibm.com/ProtocolBridgeProperties">

 <!--
  Example: ProtocolBridgeProperties.xml

  <?xml version="1.0" encoding="UTF-8"?>
  <tns:serverProperties xmlns:tns="http://wmqfte.ibm.com/ProtocolBridgeProperties"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://wmqfte.ibm.com/ProtocolBridgeProperties
        ProtocolBridgeProperties.xsd">
    <tns:credentialsFile path="$HOME/ProtocolBridgeCredentials.xml" />
      <tns:defaultServer name="myserver" />
      <tns:ftpServer name="myserver" host="myhost.hursley.ibm.com" port="1234" platform="windows"
             timeZone="Europe/London" locale="en-GB" fileEncoding="UTF-8"
             listFormat="unix" limitedWrite="false" />
    <tns:sftpServer name="server1" host="myhost.hursley.ibm.com" platform="windows"
```

```
|                     fileEncoding="UTF-8" limitedWrite="false">
|           <limits maxListFileNames="10" />
|        </tns:sftpServer>
|    </tns:serverProperties>
|    -->
|
|    <!-- Root element for the document -->
|    <element name="serverProperties" type="tns:serverPropertiesType"></element>
|
|    <!--
|     A container for all protocol bridge server properties
|     -->
|    <complexType name="serverPropertiesType">
|     <sequence>
|      <element name="credentialsFile" type="tns:credentialsFileName" minOccurs="0" maxOccurs="1" />
|      <element name="defaultServer" type="tns:serverName" minOccurs="0" maxOccurs="1" />
|      <choice minOccurs="0" maxOccurs="unbounded">
|       <element name="ftpServer" type="tns:ftpServerType" />
|       <element name="sftpServer" type="tns:sftpServerType" />
|       <element name="ftpsServer" type="tns:ftpsServerType" />
|       <element name="ftpsfgServer" type="tns:ftpsfgServerType" />
|       <element name="ftpssfgServer" type="tns:ftpssfgServerType" />
|      </choice>
|     </sequence>
|    </complexType>
|
|    <!--
|     A container for a server name
|     -->
|    <complexType name="serverName">
|     <attribute name="name" type="tns:serverNameType" use="required" />
|    </complexType>
|
|    <!--
|     A container for a credentials file name
|     -->
|    <complexType name="credentialsFileName">
|     <attribute name="path" type="string" use="required" />
|    </complexType>
|
|    <!--
|     A container for all the information about an FTP server
|     -->
|    <complexType name="ftpServerType">
|     <sequence>
|      <element name="limits" type="tns:generalLimitsType" minOccurs="0" maxOccurs="1"  />
|     </sequence>
|     <attributeGroup ref="tns:ftpServerAttributes"/>
|           <attribute name="passiveMode" type="boolean" use="optional" />
|    </complexType>
|
|    <!--
|     A container for all the information about an SFG FTP server
|     -->
|    <complexType name="ftpsfgServerType">
|     <sequence>
|      <element name="limits" type="tns:generalLimitsType" minOccurs="0" maxOccurs="1"  />
|     </sequence>
|     <attributeGroup ref="tns:ftpServerAttributes"/>
|    </complexType>
|
|    <!--
|     A container for all the information about an SFTP server
|     -->
|    <complexType name="sftpServerType">
|     <sequence>
|      <element name="limits" type="tns:sftpLimitsType" minOccurs="0" maxOccurs="1"  />
|     </sequence>
|     <attributeGroup ref="tns:sftpServerAttributes"/>
|    </complexType>
|
|    <!--
|     A container for all the information about a FTPS server
|     -->
|    <complexType name="ftpsServerType">
|     <sequence>
|      <element name="limits" type="tns:generalLimitsType" minOccurs="0" maxOccurs="1"  />
|     </sequence>
|     <attributeGroup ref="tns:ftpsServerAttributes"/>
```

```
</complexType>

<!--
 A container for all the information about a SFG FTPS server
-->
<complexType name="ftpssfgServerType">
 <sequence>
  <element name="limits" type="tns:generalLimitsType" minOccurs="0" maxOccurs="1"  />
 </sequence>
 <attributeGroup ref="tns:ftpsServerAttributes"/>
</complexType>


<!--
 Attributes common to all server types
-->
<attributeGroup name="generalServerAttributes">
 <attribute name="name" type="tns:serverNameType" use="required" />
 <attribute name="host" type="string" use="required" />
 <attribute name="port" type="nonNegativeInteger" use="optional" />
 <attribute name="platform" type="tns:platformType" use="required" />
 <attribute name="fileEncoding" type="string" use="required" />
 <attribute name="limitedWrite" type="boolean" use="optional" />
 <attribute name="controlEncoding" type="string" use="optional" />
</attributeGroup>

<!--
 Attributes common to ftp and ftps server types
-->
<attributeGroup name="ftpServerAttributes">
 <attributeGroup ref="tns:generalServerAttributes"/>
 <attribute name="timeZone" type="string" use="required" />
 <attribute name="locale" type="tns:localeType" use="required" />
 <attribute name="listFormat" type="tns:listFormatType" use="optional" />
 <attribute name="listFileRecentDateFormat" type="tns:dateFormatType" use="optional" />
 <attribute name="listFileOldDateFormat" type="tns:dateFormatType" use="optional" />
 <attribute name="monthShortNames" type="tns:monthShortNamesType" use="optional" />
</attributeGroup>

<!--
 Attributes common to ftps server types
-->
<attributeGroup name="ftpsServerAttributes">
 <attributeGroup ref="tns:ftpServerAttributes"/>
 <attribute name="ftpsType" type="tns:ftpsTypeType" use="optional" />
 <attribute name="trustStore" type="string" use="required" />
 <attribute name="trustStoreType" type="string" use="optional" />
    <attribute name="keyStore" type="string" use="optional" />
 <attribute name="keyStoreType" type="string" use="optional" />
 <attribute name="ccc" type="boolean" use="optional" />
 <attribute name="protFirst" type="boolean" use="optional" />
 <attribute name="auth" type="string" use="optional" />
 <attribute name="connectTimeout" type="nonNegativeInteger" use="optional"/>
 <attribute name="cipherSuiteList" type="string" use="optional" />
</attributeGroup>


<!--
 A container for limit-type attributes for a server. Limit parameters
 are optional, and if not specified a system default will be used.
-->
<complexType name="generalLimitsType">
 <attributeGroup ref="tns:generalLimitAttributes"/>
</complexType>

<complexType name="sftpLimitsType">
 <attributeGroup ref="tns:generalLimitAttributes"/>
 <attribute name="connectionTimeout" type="nonNegativeInteger" use="optional" />
</complexType>

<!--
 Attributes for limits common to all server types
-->
<attributeGroup name="generalLimitAttributes">
 <attribute name="maxListFileNames" type="positiveInteger" use="optional" />
 <attribute name="maxListDirectoryLevels" type="nonNegativeInteger" use="optional" />
 <attribute name="maxReconnectRetry" type="nonNegativeInteger" use="optional" />
 <attribute name="reconnectWaitPeriod" type="nonNegativeInteger" use="optional" />
 <attribute name="maxSessions" type="positiveInteger" use="optional" />
```

```
         <attribute name="socketTimeout" type="nonNegativeInteger" use="optional" />
       </attributeGroup>

       <!--
        The type for matching valid server names. Server names must be at least 2 characters in length and
        are limited to alphanumeric characters and the following characters: ".", "_", "/" and "%".
       -->
       <simpleType name="serverNameType">
             <restriction base="string">
                 <pattern value="[0-9a-zA-Z\._/%\-]{2,}"/>
             </restriction>
         </simpleType>

       <!--
        The types of platform supported.
       -->
       <simpleType name="platformType">
        <restriction base="string">
        </restriction>
       </simpleType>

       <!--
        The type for matching a locale specification.
       -->
       <simpleType name="localeType">
             <restriction base="string">
                 <pattern value="(..)[-_](..)"/>
             </restriction>
         </simpleType>

       <!--
        The types of list format supported (for FTP servers).
       -->
       <simpleType name="listFormatType">
        <restriction base="string">
        </restriction>
       </simpleType>

       <!--
        Date format for FTP client directory listing on an FTP server. This is
        the format to be passed to methods setDefaultDateFormatStr and
        setRecentDateFormatStr for Java class:
        org.apache.commons.net.ftp.FTPClientConfig
       -->
       <simpleType name="dateFormatType">
        <restriction base="string">
        </restriction>
       </simpleType>

       <!--
        A list of language-defined short month names can be specified. These are
        used for translating the directory listing received from the FTP server.
        The format is a string of three character month names separated by "|"
       -->
       <simpleType name="monthShortNamesType">
             <restriction base="string">
                 <pattern value="(...\|){11}(...)"/>
             </restriction>
         </simpleType>

         <!--
        The enumerations of the allowed FTPS types: "implicit" & "explicit"
        If not specified the default is "explicit"
       -->
       <simpleType name="ftpsTypeType">
             <restriction base="string">
                 <enumeration value="explicit"/>
                 <enumeration value="implicit"/>
             </restriction>
         </simpleType>

         <!--
          Attribute Group for SFTP Servers
          -->
         <attributeGroup name="sftpServerAttributes">
        <attributeGroup ref="tns:generalServerAttributes"/>
        <attribute name="cipherList" type="string" use="optional" />
       </attributeGroup>
     </schema>
```

Understanding the `ProtocolBridgeProperties.xml` file

The elements and attributes that are used in the `ProtocolBridgeProperties.xml` file are described in the following list:

`<serverProperties>`
Root element of the XML document

`<credentialsFile>`
Path to the file containing credentials. For WebSphere MQ V7.5, the value of this property can contain environment variables. For more information, see "Environment variables in WebSphere MQ Managed File Transfer properties" on page 537

`<defaultServer>`
The protocol file server that acts as the default server for file transfers

`<ftpServer>`
An FTP file server

`<sftpServer>`
An SFTP file server

`<ftpsServer>`
An FTPS file server

General server attributes that apply to all types of protocol file server:

Attribute	Description
name	Required. The name of the protocol file server. Protocol server names must be at least two characters in length, are not case-sensitive, and are limited to alphanumeric characters and the following characters: • period (.) • underscore (_) • forward slash (/) • percent sign (%)
host	Required. The host name or IP address of the protocol file server that you want to send files to or receive files from.
port	Optional. The port number of the protocol file server that you want to send files to or receive files from.
platform	Required. The platform of the protocol file server that you want to send files to or receive files from. Specify either `UNIX` or `WINDOWS`.Set this property according to how you enter paths on your FTP, FTPS, or SFTP server. For example, if you are running an FTP server on Windows but when you log in to the server, you must enter UNIX-style paths (that is, with forward slashes), set this value to UNIX and not WINDOWS. Servers running on Windows often present a UNIX-style file system.
fileEncoding	Required. Defines the character encoding that is used by the file server. This property is used when you transfer files in text mode so that the correct encoding sequences are changed when the files are moved between platforms. For example, UTF-8.

Attribute	Description
limitedWrite	Optional. The default mode when writing to a file server is to create a temporary file and then rename that file when the transfer has completed. For a file server that is configured as write only, the file is created directly with its final name. The value of this property can be `true` or `false`. The default is `false`.
controlEncoding	Optional. The control encoding value for control messages being sent to the protocol file server. This property affects the encoding of the file name that is used and must be compatible with the control encoding of the protocol file server. The default is `UTF-8`.

General attributes that apply to FTP and FTPS servers only:

Attribute	Description
timeZone	Required. The time zone of the protocol file server that you want to send files to or receive files from. For example: `America/New_York` or `Asia/Tokyo`.
locale	Required. The language that is used on the protocol file server that you want to send files to or receive files from. For example: `en_US` or `ja_JP`
listFormat	Optional. The listing format that defines the format of the file-listed information that is returned from the protocol file server. Use either `Windows` or `UNIX`. The default is `UNIX`.
listFileRecentDateFormat	Optional. The recent date format (less than a year) for FTP client directory listing on an FTP server. This attribute and the listFileOldDateFormat attribute allow you to redefine the expected date formats that are returned by the protocol file server. The default is as defined by the protocol file server.
listFileOldDateFormat	Optional. The old date format (more than a year) for FTP client directory listing on an FTP server. This attribute and the listFileRecentDateFormat attribute allow you to redefine the expected date formats that are returned by the protocol file server. The default is as defined by the protocol file server.
monthShortNames	Optional. A replacement list of month names that are used to decode date information returned from the protocol file server. This property consists of a list of 12 comma-separated names to override the default locale month values. The default is as defined by the protocol file server.

General attributes that apply to FTP servers only:

Attribute	Description
passiveMode	Optional. Controls whether the connection to the FTP server is passive or active. If you set the value of this property to `false`, the connection is active. If you set the value to `true`, the connection is passive. The default is `false`.

General attributes that apply to FTPS servers only:

Attribute	Description
ftpsType	Optional. Specifies whether the explicit or implicit form of the FTPS protocol is used. The default is `explicit`.
trustStore	Required. The location of the truststore that is used to determine whether the certificate presented by the FTPS server is trusted.
trustStoreType	Optional. The format of the truststore file. The default is JKS.
keyStore	Optional. The location of the keystore that is used to provide certificate information if challenged by the FTPS server. The default is for the protocol bridge to not be able to connect to FTPS servers that are configured to require the authentication of clients.
keyStoreType	Optional. The format of the keystore file. The default is JKS.
ccc	Optional. Selects whether a clear (unencrypted) command channel is used when authentication has completed. The default value is `false`, which means that the command channel remains encrypted for the entire duration of the FTPS session. This attribute is applicable only when the ftpsType is set to `explicit`.
protFirst	Optional. Specifies whether the **USER/PASS** commands are issued to the FTPS server before or after the **PBSZ/PROT** commands. The default value is `false`, which means **USER/PASS** commands are sent first followed by **PBSZ/PROT** commands. This attribute is applicable only when the ftpsType is set to `explicit`.
auth	Optional. Specifies the protocol that is specified as part of the **AUTH** command. A specified protocol will be tried first, then the default is to try TLS, SSL, TLS-C, or TLS-P until the FTPS server does not reject with a 504 reply code. This attribute is applicable only when the ftpsType is set to `explicit`.
cipherSuiteList	Specifies a comma-separated list of one or more cipher suite names. A cipher suite specifies the protocol, hash algorithm, and encryption algorithm that are used, and how many bits are used in the encryption key when data is exchanged between the agent and the FTPS server. The list supplied is used in the negotiation between the agent and the FTPS server. If you do not specify a value, the default set of ciphers enabled by Java is used in the negotiation between the agent and the FTPS server. If you specify a cipher suite that is not supported by the agent, transfers using the FTPS server will fail, even if there are supported cipher suites in the cipher suite list. For a list of valid cipher suite values, see Cipher suites in the IBM SDK and Runtime Environment Java Technology Edition Version 7 Information Center.

`<limits>`

Container element for attributes that are common to all types of server and for attributes that are specific to a type of server:

General limit attributes that apply to all types of protocol file server:

Attribute	Description
maxListFileNames	Optional. The maximum number of names that are collected when scanning a directory on the protocol file server for file names. The default is 999999999.
maxListDirectoryLevels	Optional. The maximum number of directory levels on the protocol server to recursively scan for file names. The default is 1000.
maxReconnectRetry (This attribute is now deprecated.)	**Deprecated**. This attribute is not supported on IBM WebSphere MQ V7.5. or on WebSphere MQ Managed File Transfer V7.0.2, or later. Optional. The maximum number of times a protocol server tries to reconnect before the protocol bridge agent stops trying. The default is 2.
reconnectWaitPeriod (This attribute is now deprecated.)	**Deprecated**. This attribute is not supported on IBM WebSphere MQ V7.5. or on WebSphere MQ Managed File Transfer V7.0.2, or later. Optional. The time period, in seconds, to wait to before attempting to reconnect. The default is 10 seconds.
maxSessions	Optional. The maximum number of sessions for the protocol server. This number must be greater than or equal to the sum of the maximum number of source and destination transfers for the protocol bridge agent. The default is the sum of the values for the agent properties maxSourceTransfers, maxDestinationTransfers, and maxCommandHandlerThreads, plus 1. If these three properties are using their default values of 25, 25, and 5, the maxSessions default is then 56.
socketTimeout	Optional. The socket timeout in seconds. The value of this attribute is used during file streaming. The default is 30 seconds.

Limit attribute that applies to SFTP servers only:

Attribute	Description
connectionTimeout	Optional. The time, in seconds, to wait for a response from the protocol file server to a connection request. A timeout indicates that the protocol file server is not available. The default value is 30 seconds.
cipherList	Optional. Specifies a comma-separated list of ciphers that are used to communicate between the protocol bridge agent and the SFTP server. The ciphers are called in the order that they are specified in this list. The cipher must be available on the server and the client before it can be used. The default is `aes128-cbc,aes192-cbc,aes256-cbc`.

Connect:Direct credentials file format

The ConnectDirectCredentials.xml file in the agent configuration directory defines the user names and credential information that the Connect:Direct agent uses to authorize itself with a Connect:Direct node.

The ConnectDirectCredentials.xml file must conform to the ConnectDirectCredentials.xsd schema. The ConnectDirectCredentials.xsd schema document is located in the *MQ_INSTALLATION_PATH*/mqft/samples/ schema directory of the MQMFT installation. A sample ConnectDirectCredentials.xml file is located in the *MQ_INSTALLATION_PATH*/mqft/samples/credentials directory of the MQMFT installation.

The file ConnectDirectCredentials.xml is periodically reloaded by the agent and any valid changes to the file will affect the behavior of the agent. The default reload interval is 30 seconds. This interval can be changed by specifying the agent property **xmlConfigReloadInterval** in the agent.properties file.

Schema

The following schema describes which elements are valid in the ConnectDirectCredentials.xml file.

```
<?xml version="1.0" encoding="UTF-8"?>

<!--
  This schema defines the format of the XML file that is located in the agent properties
  directory of a Connect:Direct bridge agent. The XML file ConnectDirectCredentials.xml
  is used by the default credential validation of the Connect:Direct bridge.
  For more information, see the WebSphere MQ InfoCenter
-->

<schema targetNamespace="http://wmqfte.ibm.com/ConnectDirectCredentials"
        elementFormDefault="qualified"
        xmlns="http://www.w3.org/2001/XMLSchema"
        xmlns:tns="http://wmqfte.ibm.com/ConnectDirectCredentials"

  <!--
    <?xml version="1.0" encoding="UTF-8"?>

    <tns:credentials xmlns:tns="http://wmqfte.ibm.com/ConnectDirectCredentials"
                     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                     xsi:schemaLocation="http://wmqfte.ibm.com/ConnectDirectCredentials
                     ConnectDirectCredentials.xsd">
      <tns:agent name="CDAGENT01">
        <tns:pnode name="cdnode*" pattern="wildcard">
          <tns:user name="MUSR_.*"
            ignorecase="true"
            pattern="regex"
            cdUserId="bob"
            cdPassword="passw0rd"
            pnodeUserId="bill"
            pnodePassword="alacazam">
          <tns:snode name="cdnode2" pattern="wildcard" userId="sue" password="foo"/>
          </tns:user>
        </tns:pnode>
      </tns:agent>
    </tns:credentials>

  -->

  <element name="credentials" type="tns:credentialsType"/>

  <complexType name="credentialsType">
    <sequence>
      <element name="agent" type="tns:agentType" minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
  </complexType

  <complexType name="agentType">
```

```
    <sequence>
      <element name="pnode" type="tns:pnodeType" minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
    <attribute name="name" type="string" use="required"/>
  </complexType>


  <complexType name="pnodeType">
    <sequence>
      <element name="user" type="tns:userType" minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
    <attribute name="name" type="string" use="required"/>
    <attribute name="pattern" type="tns:patternType" use="optional"/>
  </complexType

  <complexType name="userType">
    <sequence>
      <element name="snode" type="tns:snodeType" minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
    <attribute name="name" type="string" use="required"/>
    <attribute name="ignorecase" type="boolean" use="optional"/>
    <attribute name="pattern" type="tns:patternType" use="optional"/>
    <attribute name="cdUserId" type="string" use="optional"/>
    <attribute name="cdUserIdCipher" type="string" use="optional"/>
    <attribute name="cdPassword" type="string" use="optional"/>
    <attribute name="cdPasswordCipher" type="string" use="optional"/>
    <attribute name="pnodeUserId" type="string" use="optional"/>
    <attribute name="pnodeUserIdCipher" type="string" use="optional"/>
    <attribute name="pnodePassword" type="string" use="optional"/>
    <attribute name="pnodePasswordCipher" type="string" use="optional"/>
  </complexType>

  <complexType name="snodeType"
    <attribute name="name" type="string" use="required"/>
    <attribute name="pattern" type="tns:patternType" use="optional"/>
    <attribute name="userId" type="string" use="optional"/>
    <attribute name="userIdCipher" type="string" use="optional"/>
    <attribute name="password" type="string" use="optional"/>
    <attribute name="passwordCipher" type="string" use="optional"/>
  </complexType>

  <simpleType name="patternType">
    <restriction base="string">
      <enumeration value="regex"/>
      <enumeration value="wildcard"/>
    </restriction>
  </simpleType>
</schema>
```

Understanding the `ConnectDirectCredentials.xml` file

The elements and attributes used in the `ConnectDirectCredentials.xml` file are described in the following list.

`<credentials>`
> Group element containing elements that describe the credentials used by a Connect:Direct bridge agent to connect to a Connect:Direct node.

`<agent>`
> Group element containing elements for <pnode> definitions for a named agent.

`<pnode>`
> The primary node (PNODE) in the Connect:Direct transfer. This node initiates the connection to the secondary node (SNODE).

Attribute	Description
name	The name of the Connect:Direct node. The value of this attribute can be a pattern that matches many node names.
pattern	Specifies the type of pattern that is used for the value of the name attribute. Valid values for the pattern attribute are • wildcard - wildcards are used • regex - Java regular expressions are used

\<user\>
> The WebSphere MQ user that submits the transfer request.

Attribute	Description
name	The user name that is used with WebSphere MQ Managed File Transfer. The value of this attribute can be a pattern that matches many user names.
ignorecase	Specifies whether the case of the name is ignored. Valid values for the ignorecase attribute are • true - the name is not case sensitive • false - the name is case sensitive
pattern	Specifies the type of pattern that is used for the value of the name attribute. Valid values for the pattern attribute are • wildcard - wildcards are used • regex - Java regular expressions are used
cdUserId or cdUserIdCipher	The user name that is used by the Connect:Direct bridge to connect to its associated Connect:Direct node. If the **fteObfuscate** command has been used then the cipher version of the attribute must be used.
cdPassword or cdPasswordCipher	The password associated with the user name specified by the cdUserId attribute. If the **fteObfuscate** command has been used then the cipher version of the attribute must be used.
pnodeUserId or pnodeUserIdCipher	The user name that is used by the Connect:Direct primary node. If the **fteObfuscate** command has been used then the cipher version of the attribute must be used.
pnodePassword or pnodePasswordCipher	The password associated with the user name specified by the pnodeUserId attribute. If the **fteObfuscate** command has been used then the cipher version of the attribute must be used.

\<snode\>
> The Connect:Direct node that performs the role of secondary node (SNODE) during the Connect:Direct file transfer.

Attribute	Description
name	The name of the Connect:Direct node. The value of this attribute can be a pattern that matches many node names.

Attribute	Description
pattern	Specifies the type of pattern that is used for the value of the name attribute. Valid values for the pattern attribute are • wildcard - wildcards are used • regex - Java regular expressions are used
userId or userIdCipher	The user name used to connect to this node during a file transfer. If the **fteObfuscate** command has been used then the cipher version of the attribute must be used.
password or passwordCipher	The password associated with the user name specified by the userId attribute. If the **fteObfuscate** command has been used then the cipher version of the attribute must be used.

Example

In this example, the Connect:Direct bridge agent connects to the Connect:Direct node pnode1. When a WebSphere MQ user with the user name beginning with the prefix fteuser followed by a single character, for example fteuser2, requests a transfer involving the Connect:Direct bridge, the Connect:Direct bridge agent will use the user name cduser and the password passw0rd to connect to the Connect:Direct node pnode1. When the Connect:Direct node pnode1 performs its part of the transfer it uses the user name pnodeuser and the password passw0rd1.

If the secondary node in the Connect:Direct transfer has a name that begins with the prefix FISH, the node pnode1 uses the user name fishuser and the password passw0rd2 to connect to the secondary node. If the secondary node in the Connect:Direct transfer has a name that begins with the prefix CHIPS, the node pnode1 uses the user name chipsuser and the password passw0rd3 to connect to the secondary node.

```xml
<?xml version="1.0" encoding="UTF-8"?>

<tns:credentials xmlns:tns="http://wmqfte.ibm.com/ConnectDirectCredentials"
                 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                 xsi:schemaLocation="http://wmqfte.ibm.com/ConnectDirectCredentials ConnectDirectCredentials.xsd">
  <tns:agent name="CDAGENT01">
    <tns:pnode name="pnode1" pattern="wildcard">
      <tns:user name="fteuser?" pattern="wildcard" ignorecase="true"
        cdUserId="cduser" cdPassword="passw0rd"
        pnodeUserId="pnodeuser" pnodePassword="passw0rd1">
      <tns:snode name="FISH*" pattern="wildcard"
        userId="fishuser" password="passw0rd2"/>
      <tns:snode name="CHIPS*" pattern="wildcard"
        userId="chipsuser" password="passw0rd3"/>
      </tns:user>
    </tns:pnode>
  </tns:agent>
</tns:credentials>
```

Connect:Direct node properties file format

The ConnectDirectNodeProperties.xml file in the Connect:Direct bridge agent configuration directory specifies information about remote Connect:Direct nodes that are involved in a file transfer.

The ConnectDirectNodeProperties.xml file must conform to the ConnectDirectNodeProperties.xsd schema. The ConnectDirectNodeProperties.xsd schema document is located in the *MQ_INSTALLATION_PATH*/mqft/samples/schema directory of the MQMFT installation. A template ConnectDirectNodeProperties.xml file is created by the **fteCreateCDAgent** command in the agent configuration directory.

The file `ConnectDirectNodeProperties.xml` is periodically reloaded by the agent and any valid changes to the file will affect the behavior of the agent. The default reload interval is 30 seconds. This interval can be changed by specifying the agent property xmlConfigReloadInterval in the agent.properties file.

Schema

The following schema describes which elements are valid in the `ConnectDirectNodeProperties.xml` file.

```xml
<?xml version="1.0" encoding="UTF-8"?>

<schema targetNamespace="http://wmqfte.ibm.com/ConnectDirectNodeProperties"
  elementFormDefault="qualified"
  xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:tns="http://wmqfte.ibm.com/ConnectDirectNodeProperties">

  <element name="nodeProperties" type="tns:nodePropertiesType"></element>

  <complexType name="nodePropertiesType">
    <sequence>
      <element name="credentialsFile" type="tns:credentialsFileName" minOccurs="0" maxOccurs="1" />
      <element name="node" type="tns:nodeType" minOccurs="0" maxOccurs="unbounded"></element>
    </sequence>
  </complexType>

  <complexType name="nodeType">
    <attribute name="name" type="string" use="required" />
    <attribute name="pattern" type="tns:patternType" use="optional" />
    <attribute name="type" type="string" use="required" />
  </complexType>

  <simpleType name="patternType">
    <restriction base="string">
      <enumeration value="regex" />
      <enumeration value="wildcard" />
    </restriction>
  </simpleType>

</schema>
```

Understanding the `ConnectDirectNodeProperties.xml` file

The elements and attributes used in the `ConnectDirectNodeProperties.xml` file are described in the following list.

`nodeProperties`
　　Root element of the XML document.

`credentialsFile`
　　Path to the credentials file where sensitive information is stored. For WebSphere MQ V7.5, the value of this property can contain environment variables. For more information, see "Environment variables in WebSphere MQ Managed File Transfer properties" on page 537

`node`
　　Specifies one or more Connect:Direct nodes.

Attribute	Description
name	A pattern that identifies the names of Connect:Direct nodes that use the definitions specified by the node element. Pattern matching is not case sensitive.

Attribute	Description
pattern	Specifies the type of pattern that is used for the value of the name attribute. Valid values for the pattern attribute are: • wildcard - wildcards are used • regex - Java regular expressions are used For information about the types of regular expressions used by MQMFT, see "Regular expressions used by WebSphere MQ Managed File Transfer" on page 685.
type	Specifies the operating system type of the Connect:Direct node or nodes that match the pattern given by the name attribute. Valid values for the type attribute are: • Windows - the node runs on Windows • UNIX - the node runs on UNIX or Linux The value of this attribute is not case sensitive.

Example

In this example, the file specifies that all Connect:Direct nodes that have a name that begins with "cdnodew" run on a Windows platform and all Connect:Direct nodes that have a name that begins with "cdnodeu" run on a UNIX platform. The file specifies that all other Connect:Direct nodes run on a UNIX platform. The Connect:Direct bridge agent searches for matches from the start of the file to the end of the file and uses the first match that it finds. The Connect:Direct credentials file has been specified as ConnectDirectCredentials.xml.

```
<?xml version="1.0" encoding="UTF-8"?>
<tns:nodeProperties xmlns:tns="http://wmqfte.ibm.com/ConnectDirectNodeProperties"
                    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                    xsi:schemaLocation="http://wmqfte.ibm.com/ConnectDirectNodeProperties
                                          ConnectDirectNodeProperties.xsd">

    <tns:credentialsFile path="ConnectDirectCredentials.xml" />
    <tns:node name="cdnodew*" pattern="wildcard" type="windows" />
    <tns:node name="cdnodeu.*" pattern="regex" type="unix" />
    <tns:node name="*" pattern="wildcard" type="unix" />

</tns:nodeProperties>
```

Connect:Direct process definitions file format

The ConnectDirectProcessDefinitions.xml file in the Connect:Direct bridge agent configuration directory specifies the user-defined Connect:Direct process to start as part of the file transfer.

The ConnectDirectProcessDefinitions.xml file must conform to the ConnectDirectProcessDefinitions.xsd schema. The ConnectDirectProcessDefinitions.xsd schema document is located in the *MQ_INSTALLATION_PATH*/mqft/samples/schema directory of the MQMFT installation. A template ConnectDirectProcessDefinitions.xml file is created by the **fteCreateCDAgent** command in the agent configuration directory.

The file ConnectDirectProcessDefinitions.xml is periodically reloaded by the agent and any valid changes to the file will affect the behavior of the agent. The default reload interval is 30 seconds. This interval can be changed by specifying the agent property xmlConfigReloadInterval in the agent.properties file.

Schema

The following schema describes which elements are valid in the ConnectDirectProcessDefinitions.xml file.

```xml
<schema targetNamespace="http://wmqfte.ibm.com/ConnectDirectProcessDefinitions"
        elementFormDefault="qualified"
        xmlns="http://www.w3.org/2001/XMLSchema"
        xmlns:tns="http://wmqfte.ibm.com/ConnectDirectProcessDefinitions">

    <element name="cdprocess" type="tns:cdprocessType"></element>

    <complexType name="cdprocessType">
        <sequence>
            <element name="processSet" type="tns:processSetType"
                    minOccurs="0"     maxOccurs="unbounded"></element>
        </sequence>
    </complexType>

    <complexType name="processSetType">
        <sequence>
            <element name="condition" type="tns:conditionType"
                    minOccurs="0"    maxOccurs="1" />
            <element name="process"    type="tns:processType"
                    minOccurs="1"    maxOccurs="1" />
        </sequence>
    </complexType>

    <complexType name="conditionType">
        <choice minOccurs="0" maxOccurs="unbounded">
            <element name="match" type="tns:matchType" />
            <element name="defined" type="tns:definedType" />
        </choice>
    </complexType>

    <complexType name="matchType">
        <attribute name="variable" type="string" use="required" />
        <attribute name="value" type="string" use="required" />
        <attribute name="pattern" type="tns:patternType" use="optional" />
    </complexType>

    <complexType name="definedType">
        <attribute name="variable" type="string" use="required" />
    </complexType>

    <complexType name="processType">
        <sequence>
            <element name="preTransfer"          type="tns:transferType"
                    minOccurs="0"                maxOccurs="1" />
            <element name="transfer"             type="tns:transferType"
                    minOccurs="0"                maxOccurs="1" />
            <element name="postTransferSuccess" type="tns:transferType"
                    minOccurs="0"                maxOccurs="1" />
            <element name="postTransferFailure" type="tns:transferType"
                    minOccurs="0"                maxOccurs="1" />
        </sequence>
    </complexType>

    <complexType name="transferType">
        <attribute name="process" type="string" use="required" />
    </complexType>

    <simpleType name="patternType">
        <restriction base="string">
            <enumeration value="regex" />
            <enumeration value="wildcard" />
        </restriction>
    </simpleType>

</schema>
```

Understanding the `ConnectDirectProcessDefinitions.xml` file

The elements and attributes used in the `ConnectDirectProcessDefinitions.xml` file are described in the following list.

`cdProcess`
The root element of the XML document.

`processSet`
Group element containing all the information about a set of user-defined processes.

`condition`
Group element containing the conditions that a transfer is tested against to determine whether the set of processes contained in the processSet element are used.

`match`
A condition that tests whether a the value of a variable matches a given value.

Attribute	Description
variable	Specifies a variable. The value of this variable is compared with the value of the value attribute. The variable is an intrinsic symbol. For more information, see "Substitution variables for use with user-defined Connect:Direct processes" on page 686.
value	Specifies a pattern to match against the value of the variable specified by the variable attribute.
pattern	Specifies the type of pattern that is used for the value of the value attribute. Valid values for the pattern attribute are • `wildcard` - wildcards are used • `regex` - Java regular expressions are used This attribute is optional and the default is `wildcard`.

`defined`
A condition that tests whether a variable has been defined.

Attribute	Description
variable	Specifies a variable. If this variable exists, the match condition is satisfied. The variable is an intrinsic symbol. For more information, see "Substitution variables for use with user-defined Connect:Direct processes" on page 686.

`process`
Group element containing the information about where to locate the Connect:Direct processes to call when a match is found.

`transfer`
The Connect:Direct process to call during a transfer request.

Attribute	Description
process	Optional. Specifies the name of a file that contains a Connect:Direct process to call during a transfer request. The file path is relative to the Connect:Direct bridge agent configuration directory. This attribute is optional, the default is to use a process generated by MQMFT. For WebSphere MQ V7.5, the value of this property can contain environment variables. For more information, see "Environment variables in WebSphere MQ Managed File Transfer properties" on page 537

Example

In this example, there are three processSet elements.

The first processSet element specifies that if a transfer request has a **%FTESNODE** variable with a value that matches the pattern `Client*` and a **%FTESUSER** variable with a value of `Admin`, the Connect:Direct bridge agent submits the Connect:Direct process located in the *agent_configuration_directory*/`AdminClient.cdp` as part of the transfer.

The second processSet element specifies that if a transfer request has a **%FTESNODE** variable with a value that matches the pattern `Client*`, the Connect:Direct bridge agent submits the Connect:Direct process located in the *agent_configuration_directory*/`Client.cdp` as part of the transfer. The Connect:Direct bridge agent reads the processSet elements in the order that they are defined, and if it finds a match, it uses the first match and does not look for another match. For transfer requests that match the conditions of both the first and second processSet, the Connect:Direct bridge agent calls only the processes specified by the first processSet.

The third processSet element has no conditions and matches all transfers. If the transfer request does not match the conditions of the first or second processSet, the Connect:Direct bridge agent submits the Connect:Direct process specified by the third condition. This process is located in the *agent_configuration_directory*/`Default.cdp` as part of the transfer.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<tns:cdprocess xmlns:tns="http://wmqfte.ibm.com/ConnectDirectProcessDefinitions"
           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xsi:schemaLocation="http://wmqfte.ibm.com/ConnectDirectProcessDefinitions ConnectDirectProcessDefinitions.xsd">

    <tns:processSet>
        <tns:condition>
            <tns:match variable="%FTESNODE" value="Client*" pattern="wildcard" />
            <tns:match variable="%FTESUSER" value="Admin" pattern="wildcard" />
        </tns:condition>
        <tns:process>
            <tns:transfer process="AdminClient.cdp" />
        </tns:process>
    </tns:processSet>

    <tns:processSet>
        <tns:condition>
            <tns:match variable="%FTESNODE" value="Client*" pattern="wildcard" />
        </tns:condition>
        <tns:process>
            <tns:transfer process="Client.cdp" />
        </tns:process>
    </tns:processSet>

    <tns:processSet>
        <tns:process>
            <tns:transfer process="Default.cdp" />
        </tns:process>
    </tns:processSet>

</tns:cdprocess>
```

Working with user sandboxes

You can restrict the area of the file system that files can be transferred into and out of based on the MQMD user name that requests the transfer.

User sandboxes are not supported when the agent is a protocol bridge agent or a Connect:Direct bridge agent.

To enable user sandboxing, add the following property to the `agent.properties` file for the agent that you want to restrict:

`userSandboxes=true`

When this property is present and set to true the agent uses the information in the *MQ_DATA_PATH*/mqft/config/*coordination_qmgr_name*/agents/*agent_name*/UserSandboxes.xml file to determine which parts of the file system the user who requests the transfer can access.

The `UserSandboxes.xml` XML is composed of an `<agent>` element that contains zero or more `<sandbox>` elements. These elements describe which rules are applied to which users. The `user` attribute of the `<sandbox>` element is a pattern that is used to match against the MQMD user of the request.

The file `UserSandboxes.xml` is periodically reloaded by the agent and any valid changes to the file will affect the behavior of the agent. The default reload interval is 30 seconds. This interval can be changed by specifying the agent property xmlConfigReloadInterval in the `agent.properties` file.

If you specify the `userPattern="regex"` attribute or value, the `user` attribute is interpreted as a Java regular expression. For more information, see "Regular expressions used by WebSphere MQ Managed File Transfer" on page 685.

If you do not specify the `userPattern="regex"` attribute or value the `user` attribute is interpreted as a pattern with the following wildcard characters:
- asterisk (*), which represents zero or more characters
- question mark (?), which represents exactly one character

Matches are performed in the order that the `<sandbox>` elements are listed in the file. Only the first match is used, all following potential matches in the file are ignored. If none of the `<sandbox>` elements specified in the file match the MQMD user associated with the transfer request message, the transfer cannot access the file system. When a match has been found between the MQMD user name and a `user` attribute, the match identifies a set of rules inside a `<sandbox>` element that are applied to the transfer. This set of rules is used to determine which files can be read from or written to as part of the transfer.

Each set of rules can specify a `<read>` element, which identifies which files can be read, and a `<write>` element which identifies which files can be written. If you omit the `<read>` or `<write>` elements from a set of rules, it is assumed that the user associated with that set of rules is not allowed to perform any reads or any writes, as appropriate.

Note: The `<read>` element must be before the `<write>` element, and the `<include>` element must be before the `<exclude>` element, in the `UserSandboxes.xml` file.

Each `<read>` or `<write>` element contains one or more patterns that are used to determine whether a file is in the sandbox and can be transferred. Specify these patterns by using the `<include>` and `<exclude>` elements. The `name` attribute of the `<include>` or `<exclude>` element specifies the pattern to be matched. An optional `type` attribute specifies whether the name value is a file or queue pattern. If the `type` attribute is not specified, the agent treats the pattern as a file or directory path pattern. For example:

```
<tns:read>
 <tns:include name="/home/user/**"/>
 <tns:include name="USER.**" type="queue"/>
 <tns:exclude name="/home/user/private/**"/>
</tns:read>
```

The `<include>` and `<exclude>` name patterns are used by the agent to determine whether files or queues can be read from or written to. An operation is allowed if the canonical file path or queue name matches at least one of the included patterns and exactly zero of the excluded patterns. The patterns specified by using the `name` attribute of the `<include>` and `<exclude>` elements use the path separators and conventions appropriate to the platform that the agent is running on. If you specify relative file paths, the paths are resolved relative to the `transferRoot` property of the agent.

When specifying a queue restriction, a syntax of QUEUE@QUEUEMANAGER is supported, with the following rules:

- If the at character (@) is missing from the entry, the pattern is treated as a queue name that can be accessed on any queue manager. For example, if the pattern is `name` it is treated the same way as `name@**`.
- If the at character (@) is the first character in the entry, the pattern is treated as a queue manager name and all queues on the queue manager can be accessed. For example, if the pattern is `@name` it is treated the same way as `**@name.`.

The following wildcard characters have special meaning when you specify them as part of the `name` attribute of the `<include>` and `<exclude>` elements:

*	A single asterisk matches zero or more characters in a directory name, or in a qualifier of a queue name.
?	A question mark matches exactly one character in a directory name, or in a qualifier of a queue name.
**	Two asterisk characters match zero or more directory names, or zero or more qualifiers in a queue name. Also, paths that end with a path separator have an implicit "**" added to the end of the path. So /home/user/ is the same as /home/user/**.

For example:

- `/**/test/**` matches any file that has a `test` directory in its path
- `/test/file?` matches any file inside the `/test` directory that starts with the string `file` followed by any single character
- `c:\test\*.txt` matches any file inside the `c:\test` directory with a `.txt` extension
- `c:\test\**\*.txt` matches any file inside the `'c:\test` directory, or one of its subdirectories that has a `.txt` extension
- `TEST.*.QUEUE@QM1` matches any queue on the queue manager QM1 that has the first qualifier of `TEST`, has any second qualifier, and a third qualifier of `QUEUE`.

Symbolic links

You must fully resolve any symbolic links that you use in file paths in the `UserSandboxes.xml` file by specifying hard links in the `<include>` and `<exclude>` elements. For example, if you have a symbolic link where /var maps to /SYSTEM/var, you must specify this path as `<tns:include name="/SYSTEM/var"/>`, otherwise the intended transfer fails with a user sandbox security error.

Paths on IBM 4690 systems

For information about how paths specified in the `UserSandboxes.xml` file are interpreted on IBM 4690, see

Example

To allow the user with the MQMD user name `guest` to transfer any file from the `/home/user/public`
directory or any of its subdirectories on the system where the agent AGENT_JUPITER is running, add the
following <sandbox> element to the file `UserSandboxes.xml` in AGENT_JUPITER's configuration directory

```
<?xml version="1.0" encoding="UTF-8"?>
<tns:userSandboxes
  xmlns:tns="http://wmqfte.ibm.com/UserSandboxes"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://wmqfte.ibm.com/UserSandboxes UserSandboxes.xsd">
 <tns:agent>
  <tns:sandbox user="guest">
   <tns:read>
    <tns:include name="/home/user/public/**"/>
    </tns:read>
  </tns:sandbox>
 </tns:agent>
</tns:userSandboxes>
```

Example

To allow any user with the MQMD user name `account` followed by a single digit, for example `account4`,
to complete the following actions:

- Transfer any file from the `/home/account` directory or any of its subdirectories, excluding the
  `/home/account/private` directory on the system where the agent AGENT_SATURN is running
- Transfer any file to the `/home/account/output` directory or any of its subdirectories on the system
  where the agent AGENT_SATURN is running
- Read messages from queues on the local queue manager starting with the prefix `ACCOUNT.` unless it
  starts with `ACCOUNT.PRIVATE.` (that is has `PRIVATE` at the second level).
- Transfer data onto queues starting with the prefix `ACCOUNT.OUTPUT.` on any queue manager.

add the following <sandbox> element to the file `UserSandboxes.xml`, in AGENT_SATURN's configuration
directory,

```
<?xml version="1.0" encoding="UTF-8"?>
<tns:userSandboxes
   xmlns:tns="http://wmqfte.ibm.com/UserSandboxes"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://wmqfte.ibm.com/UserSandboxes UserSandboxes.xsd">
 <tns:agent>
  <tns:sandbox user="account[0-9]" userPattern="regex">
   <tns:read>
    <tns:include name="/home/account/**"/>
    <tns:include name="ACCOUNT.**" type="queue"/>
    <tns:exclude name="ACCOUNT.PRIVATE.**" type="queue"/>
    <tns:exclude name="/home/account/private/**"/>
                                  </tns:read>
   <tns:write>
    <tns:include name="/home/account/output/**"/>
    <tns:include name="ACCOUNT.OUTPUT.**" type="queue"/>
   </tns:write>
  </tns:sandbox>
 </tns:agent>
</tns:userSandboxes>
```

Logger configuration properties for WebSphere MQ Managed File Transfer

The logger has a set of configuration properties. Specify these properties in the `logger.properties` file, which is in the *MQ_DATA_PATH*/mqft/config/*coordination_qmgr_name*/loggers/*logger_name* directory.

For WebSphere MQ V7.5, there is the ability for environment variables to be used in some Managed File Transfer properties that represent file or directory locations. This allows the locations of files or directories that are used when running parts of the product, to vary depending on environment changes, such as which user is running the process. For more information, see "Environment variables in WebSphere MQ Managed File Transfer properties" on page 537.

Note: When you specify file paths on Windows, the backslash (\) separator character must appear as double backslashes (\\) (that is, escaped \). Alternatively, you can use a single forward slash character (/) as a separator. For more information about character escaping in Java properties files in Oracle, see Javadoc for the Properties class.

Property name	Description	Default value
wmqfte.logger.type	The logger type in use: file, or database. Set this value to `FILE`, or `DATABASE`.	No default value
wmqfte.max.transaction.messages	The maximum number of messages that is processed in a transaction before the transaction is committed. In circular logging mode, a queue manager has a fixed amount of space available for inflight data. Ensure that you set this property with a sufficiently low value so that the available space does not run out.	50
wmqfte.max.transaction.time	The maximum length of time in milliseconds that passes between transaction commits.	5000
wmqfte.max.consecutive.reject	The maximum number of messages that can be rejected consecutively (that is, without encountering a valid message). If this number is exceeded the logger concludes that the problem is not with the messages themselves but with the configuration. For example, if you make an agent-name column in the database narrower than all of your agent names, all messages referring to agents are rejected.	50
wmqfte.reject.queue.name	The name of a queue to which the logger puts messages that the logger cannot handle. If you have a database logger see Database logger error handling and rejection for details of which messages might be put onto this queue.	SYSTEM.FTE.LOG.RJCT.*logger_name*
wmqfte.command.queue.name	The name of a queue that the logger reads command messages controlling its behavior from.	SYSTEM.FTE.LOG.CMD.*logger_name*
wmqfte.queue.manager	The queue manager that the logger connects to (the queue manager must be on the same machine as the logger).	No default value

Property name	Description	Default value
wmqfte.message.source.type	One of the following values: **automatic subscription** The default value. The logger creates and uses its own durable, managed subscription on the queue manager that is defined in SYSTEM.FTE/Log/#. This is an appropriate value for most scenarios. **administrative subscription** If the automatic subscription is not appropriate, you can define a different subscription (for example, by using the WebSphere MQ Explorer, MQSC, or PCF) and instruct the logger to use that subscription. For example, use this value to partition the log space so that one logger handles agents from A-H, another logger handles I-P, and a third logger from Q-Z. **queue** If the WebSphere MQ topology means that creating a subscription for the logger is not convenient, you can use a queue instead. Configure WebSphere MQ so that the queue receives the messages that are typically received by a subscription to SYSTEM.FTE/Log/# on the coordination queue manager.	`automatic subscription`
wmqfte.message.source.name	If the message source type is `administrative subscription` or `queue`, the name of the subscription or queue to use. This property is ignored if the source type is `automatic subscription`.	No default value
wmqfte.database.credentials.file	The file that contains the user name and password for connecting to the database. For WebSphere MQ V7.5, the value of this property can contain environment variables. For more information, see "MQMFT credentials file format" on page 835.	The default value for this property is `%HOMEDRIVE%%HOMEPATH%\mqmftcredentials.xml`" on Windows and `$HOME/MQMFTCredentials.xml` on other platforms.
wmqfte.database.driver	The location of the JDBC driver classes for the database. This is typically the path and file name of a JAR file. For example, the Type 2 driver for Db2 on AIX systems requires the file `/opt/IBM/db2/V9.5/java/db2jcc.jar`. On Windows systems, specify the path separator as a forward slash character (/) for example, `C:/Program Files/IBM/SQLLIB/java/db2jcc.jar`. If your database driver consists of multiple JAR files (for example, Db2 V9.1 requires a driver JAR file and a license JAR file), include all of these JAR files in this property. Separate multiple file names by using the classpath separator for your platform, that is, the semicolon character (;) on Windows systems and the colon character (:) on other platforms.	No default value

Property name	Description	Default value
wmqfte.database.exclude.duplicate.metadata	Controls whether entries are stored in the metadata table that contains information that can be found in other tables within the database logger schema. Set this value to true, or false. These metadata entries are no longer stored by default as it is duplication of existing data and a waste of database storage capacity. The property entries and the tables, where the same data appears, are as follows: • com.ibm.wmqfte.SourceAgent TRANSFER_EVENT or CALL_REQUEST • com.ibm.wmqfte.DestinationAgent TRANSFER_EVENT • com.ibm.wmqfte.MqmdUser TRANSFER_EVENT or CALL_REQUEST • com.ibm.wmqfte.OriginatingUser TRANSFER_EVENT or CALL_REQUEST • com.ibm.wmqfte.OriginatingHost TRANSFER_EVENT or CALL_REQUEST • com.ibm.wmqfte.TransferId TRANSFER or CALL_REQUEST • com.ibm.wmqfte.JobName TRANSFER or CALL_REQUEST Setting the value of this property to false causes these metadata entries to be stored in the metadata table.	true
wmqfte.database.host	DB2 only: For WebSphere MQ V7.5, the host name of the database server to connect to using a Type 4 JDBC driver. If a value for this property is specified, then a value for wmqfte.database.port must also be specified. If both properties are not defined, the database logger connects by using the default Type 2 JDBC driver. If a value for this property is specified, then a credentials file for this logger (file path defined by the wmqfte.database.credentials.file property) must exist, and be accessible to define the user name and password for connecting to the database, even if the database is on the local system.	No default value
wmqfte.database.name	The name of the database that contains the WebSphere MQ Managed File Transfer log tables.	No default value
wmqfte.database.type	The database management system in use: Db2 or Oracle. Set this value to db2 or oracle.	db2

Property name	Description	Default value
wmqfte.database.port	DB2 only: For WebSphere MQ V7.5, the port number of the database server to connect to using a Type 4 JDBC driver. If a value for this property is specified, then a value for `wmqfte.database.host` must also be specified. If both properties are not defined, the database logger connects by using the default Type 2 JDBC driver. If a value for this property is specified, then a credentials file for this logger (file path defined by the wmqfte.database.credentials.file property) must exist, and be accessible to define the user name and password for connecting to the database, even if the database is on the local system.	No default value
wmqfte.database.schema	The database schema that contains the WebSphere MQ Managed File Transfer logging tables. In most cases the default value is appropriate, but you might need to specify an alternative value depending on your own site-specific database considerations.	`FTELOG`
wmqfte.database.native.library.path	The path that contains the native libraries that are needed by your chosen database driver (if any). For example, the Type 2 driver for DB2 on AIX systems requires libraries from `/opt/IBM/db2/V9.5/lib32/`. As an alternative to this property, you can set the java.library.path system property by using other methods. On Solaris and HP-UX systems, before running the **fteStartLogger** command, you must also set and export the LD_LIBRARY_PATH environment variable to include the path.	No default value
wmqfte.file.logger.fileDirectory	The directory where the file logger log files are located.	`mqft/logs/coordination_dir/loggers/` `logger_name/logs`

wmqfte.file.logger.fileSize	The maximum size that a log file is allowed to grow to. The size value is a positive integer, greater than zero, followed by one of the following units: KB, MB, GB, m (minutes), h (hours), d (days), w (weeks). For example,
	
	`wmqfte.file.logger.fileSize=5MB`
	
	Specifies a maximum file size of 5MB.
	`wmqfte.file.logger.fileSize=2d`
	
	Specifies a maximum file size of 2 days of data.
		`10MB`

wmqfte.file.logger.fileCount	The maximum number of log files to create. When the amount of data exceeds the maximum amount that can be stored in this number of files, the oldest file is deleted so that the number of files never exceeds the value that is specified.	3

Property name	Description	Default value

wmqfte.file.logger.mode	The logger mode in use: circular, or linear. Set this value to `CIRCULAR`, or `LINEAR`.
	
	CIRCULAR - The file logger writes information to a file until that file reaches its maximum size as defined by using the **`wmqfte.file.logger.fileSize`** property. When the maximum size is reached, the file logger starts a new file. The maximum number of files that are written in this mode is controlled by the value that is defined by using the **`wmqfte.file.logger.fileCount`** property. When this maximum number of files is reached, the file logger deletes the first file and re-creates it for use as the currently active file. If the value defined in the **`wmqfte.file.logger.fileSize`** property is a fixed size byte unit (for example, KB, MB, or GB) then the upper limit on disk space that is used in this mode equals fileSize multiplied by fileCount. If the value defined in the **`wmqfte.file.logger.fileSize`** property is a time unit (for example, `m`, `h`, `d`, or `w`) then the maximum size depends on the throughput of log messages in your system over these time periods. The log file naming convention that is used when running in this mode is: *logger_namenumber-timestamp*`.log` where:
	
	• *logger_name* is the name that is given to the logger in the **`fteCreateLogger`** command.
	• *number* is the number of the file within the set.
	• *timestamp* is the timestamp of when the file was created.
	
	For example,
	`LOGGER1-20111216123430147.log`
	
	LINEAR - The file logger writes information to a file until that file reaches its maximum size as defined by using the **`wmqfte.file.logger.fileSize`** property. When the maximum size is reached the file logger starts a new file. Previously written files are not deleted, which allows them to be kept as a historical record of log messages. Files are not deleted when running in `linear` mode, so the **`wmqfte.file.logger.fileCount`** property is ignored because there is no upper limit to the number of files that can be created. Because there is no upper limit when running in this mode, it is necessary to track the amount of disk space that is used by the log files to avoid running low on disk space. The log file naming convention that is used when running in this mode is: *logger_name-timestamp*`.log` where:
	
	• *logger_name* is the name that is given to the logger in the **`fteCreateLogger`** command.
	• *timestamp* is the timestamp of when the file was created.
	
	For example,
	`LOGGER-20111216123430147.log`
		No default value

Property name	Description	Default value
wmqfte.max.retry.interval	The maximum time, in seconds, between retries when the logger encounters a persistent error. Some error conditions (for example, loss of database connection) prevent the logger from continuing. When this type of condition occurs, the logger rolls back the current transaction, waits for a period, and then retries. The time that the logger waits is initially very short, so that transitory errors can be overcome quickly. However, each time the logger retries, the time that it waits is increased. This prevents too much unnecessary work from taking place when the error condition is longer-lasting, for example when a database is taken down for maintenance. Use this property to set a limit to the length of the wait so that a retry occurs in a reasonable time of the error condition being resolved.	600
loggerQMgrRetryInterval	The interval, in seconds, between checks on the availability of the queue manager by the logger's process controller.	30
maxRestartCount	The maximum number of restarts that can happen within the time interval specified by the value of the maxRestartInterval property. When this value is exceeded the logger's process controller stops restarting the logger, and instead performs an action that is based on the value of the maxRestartDelay property.	4
maxRestartInterval	The interval, in seconds, that the logger's process controller measures logger restarts over. If the number of restarts in this interval exceeds the value of the maxRestartCount property, the logger's process controller stops restarting the logger. Instead the logger's process controller performs an action that is based on the value of the maxRestartDelay property.	120
maxRestartDelay	Determines the behavior of the logger's process controller when the rate of logger restarts exceeds the value of the maxRestartCount and maxRestartInterval properties. If you specify a value less than or equal to zero, the logger's process controller is stopped. If you specify a value greater than zero, this is the number of seconds to wait before the restart history information held by the logger's process controller is reset and the logger is restarted.	-1
wmqfte.oracle.port	The port that the logger uses to connect to the Oracle instance. This port is also known as a TNS listener.	1521
wmqfte.oracle.host	The host that the logger uses to connect to the Oracle instance.	localhost

Property name	Description	Default value
trace	Optional property. Trace specification when the logger is to be run with trace enabled at logger start. The trace specification is a comma-separated list of classes, the equals character, and a trace level. For example, `com.ibm.wmqfte.databaselogger,` `com.ibm.wmqfte.databaselogger.operation=all`. You can specify multiple trace specifications in a colon-separated list. For example, `com.ibm.wmqfte.databaselogger=moderate:com.ibm.wmqfte.databaselogger.operation=all`	None
traceFiles	Optional property. The total number of trace files to keep. This value applies to the process controller of a logger, as well as the logger itself.	5
traceSize	Optional property. The maximum size in MB of each trace file, before trace wraps onto the next file. This value applies to the process controller of the logger, and the logger itself.	20

Related reference:

"Environment variables in WebSphere MQ Managed File Transfer properties" on page 537
In WebSphere MQ V7.5, it is possible for environment variables to be used in Managed File Transfer properties that represent file or directory locations. This allows the locations of files or directories used when running parts of the product, to vary depending on environment changes. For example, which user is running the process.

Java system properties

A number of WebSphere MQ Managed File Transfer command and agent properties must be defined as Java system properties, because they define configuration for early function that is unable to use the command or agent properties mechanism.

Define system properties and other JVM options for the JVM that is to run WebSphere MQ Managed File Transfer commands by defining the environment variable FTE_JVM_PROPERTIES. For example, to set the com.ibm.wmqfte.maxConsoleLineLength property on a UNIX-type platform, define the variable as follows:

`export FTE_JVM_PROPERTIES="-Dcom.ibm.wmqfte.maxConsoleLineLength=132"`

If you are running an agent as a Windows service, you can modify the agent's Java system properties by specifying the -sj parameter on the **fteModifyAgent** command.

Table 36. Java system properties

Property name	Description	Value
com.ibm.wmqfte.maxConsoleLineLength	Maximum length of line that can be written to the console. Lines that exceed this length are word wrapped. This value is expressed in bytes (not characters).	The length is unlimited.
com.ibm.wmqfte.daemon.windows.windowsServiceLogFilesm	(Windows only.) Specifies the maximum number of Windows service log files to keep. Windows service log files are created in the agent and database logger logs directories if these applications are running as a Windows service. Windows service log files are named with the prefix *service*, and contain messages about the starting and stopping of the service.	5

Related concepts:

"Configuration options" on page 124
WebSphere MQ Managed File Transfer provides a set of properties files that contain key information
about your setup and are required for operation. These properties files are in the configuration directory
that you defined when you installed the product.

SSL properties

Use SSL withWebSphere MQ and WebSphere MQ Managed File Transfer to prevent unauthorized
connections between agents and queue managers, and to encrypt message traffic between agents and
queue managers.

For information about using SSL with WebSphere MQ Managed File Transfer, see "Configuring SSL
encryption for WebSphere MQ Managed File Transfer" on page 109.

For WebSphere MQ V7.5 or later, there is the ability for environment variables to be used in some
Managed File Transfer properties that represent file or directory locations. This allows the locations of
files or directories that are used when running parts of the product to vary depending on environment
changes, such as which user is running the process. For more information, see "Environment variables in
WebSphere MQ Managed File Transfer properties" on page 537.

Table 37. SSL properties for the agent.properties file

Property name	Description	Default value
agentSslCipherSpec	Specifies the protocol, hash algorithm, and encryption algorithm that is used, and how many bits are used in the encryption key, when data is exchanged between the agent and the agent queue manager. The value of agentSslCipherSpec is a CipherSpec name. This CipherSpec name is the same as the CipherSpec name used on the agent queue manager channel. A list of valid CipherSpec names is included in SSL CipherSpecs and CipherSuites in WebSphere MQ classes for Java. agentSslCipherSpec is similar to agentSslCipherSuite. If both agentSslCipherSuite and agentSslCipherSpec are specified, the value of agentSslCipherSpec is used.	None
agentSslCipherSuite	Specifies SSL aspects of how the agent and the agent queue manager exchange data. The value of agentSslCipherSuite is a CipherSuite name. The CipherSuite name maps to the CipherSpec name used on the agent queue manager channel. For more information, see CipherSuite and CipherSpec name mappings. agentSslCipherSuite is similar to agentSslCipherSpec. If both agentSslCipherSuite and agentSslCipherSpec are specified, the value of agentSslCipherSpec is used.	None
agentSslPeerName	Specifies a distinguished name skeleton that must match the name that is provided by the agent queue manager. The distinguished name is used to check the identifying certificate that is presented by the queue manager on connection.	None

Table 37. SSL properties for the agent.properties file (continued)

Property name	Description	Default value
agentSslTrustStore	Specifies the location of the certificates that the agent trusts. The value of agentSslTrustStore is a file path. If it is a Windows file path the backslash character (\) must be escaped (\\). For WebSphere MQ V7.5 or later, the value of this property can contain environment variables.	None
agentSslKeyStore	Specifies the location of the private key of the agent. The value of agentSslKeyStore is a file path. If it is a Windows file path the backslash character (\) must be escaped (\\). This property is only required if the agent queue manager requires client authentication. For WebSphere MQ V7.5 or later, the value of this property can contain environment variables.	None
agentSslFipsRequired	Specifies that you want to enable FIPS support at the level of the agent. The value of this property can be `true` or `false`. For more information, see "FIPS support" on page 691. If you set this property to true, you are also specifying that FIPS support is required for communication between a protocol bridge agent and any configured FTPS servers.	false
agentSslKeyStoreType	The type of SSL keystore you want to use. JKS and PKCS#12 keystores are supported. The value of this property can be either `jks` or `pkcs12`.	jks
agentSslKeyStoreCredentialsFile	The path to the file that contains the agentSslKeyStore credential. For WebSphere MQ V7.5 or later, the value of this property can contain environment variables.	The default value for this property is `%USERPROFILE%\ MQMFTCredentials.xml` on Windows, `f:/adxetc/mft75/ mqft/config/ mqmftcredentials.xml` on IBM 4690, and `$HOME/ MQMFTCredentials.xml` on other platforms.
agentSslTrustStoreType	The type of SSL keystore you want to use. JKS and PKCS#12 keystores are supported. The value of this property can be either `jks` or `pkcs12`.	jks
agentSslTrustStoreCredentialsFile	The path to the file that contains the agentSslTrustStore credential. For WebSphere MQ V7.5 or later, the value of this property can contain environment variables.	The default value for this property is `%USERPROFILE%\ MQMFTCredentials.xml` on Windows, `f:/adxetc/mft75/ mqft/config/ mqmftcredentials.xml` on IBM 4690, and `$HOME/ MQMFTCredentials.xml` on other platforms.

Table 38. SSL properties for the coordination.properties file

Property name	Description	Default value
coordinationSslCipherSpec	Specifies the protocol, hash algorithm, and encryption algorithm that is used, and how many bits are used in the encryption key, when data is exchanged between the commands and the coordination queue manager. The value of coordinationSslCipherSpec is a CipherSpec name. This CipherSpec name is the same as the CipherSpec name used on the coordination queue manager channel. A list of valid CipherSpec names is included in SSL CipherSpecs and CipherSuites in WebSphere MQ classes for Java. coordinationSslCipherSpec is similar to coordinationSslCipherSuite. If both coordinationSslCipherSuite and coordinationSslCipherSpec are specified, the value of coordinationSslCipherSpec is used.	None
coordinationSslCipherSuite	Specifies SSL aspects of how the commands and the coordination queue manager exchange data. The value of coordinationSslCipherSuite is a CipherSuite name. The CipherSuite name maps to the CipherSpec name used on the agent queue manager channel. For more information, see CipherSuite and CipherSpec name mappings. coordinationSslCipherSuite is similar to coordinationSslCipherSpec. If both coordinationSslCipherSuite and coordinationSslCipherSpec are specified, the value of coordinationSslCipherSpec is used.	None
coordinationSslPeerName	Specifies a distinguished name skeleton that must match the name that is provided by the coordination queue manager. The distinguished name is used to check the identifying certificate that is presented by the coordination queue manager on connection.	None
coordinationSslTrustStore	Specifies the location of the certificates that the commands trust. The value of coordinationSslTrustStore is a file path. If it is a Windows file path, the backslash character (\) must be escaped (\\). For WebSphere MQ V7.5 or later, the value of this property can contain environment variables.	None
coordinationSslTrustStoreType	The type of SSL keystore you want to use. JKS and PKCS#12 keystores are supported. The value of this property can be either jks or pkcs12.	jks
coordinationSslTrustStoreCredentialsFile	The path to the file that contains the coordinationSslTrustStore credentials. For WebSphere MQ V7.5 or later, the value of this property can contain environment variables.	The default value for this property is %USERPROFILE%\ MQMFTCredentials.xml on Windows, f:/adxetc/mft75/ mqft/config/ mqmftcredentials.xml on IBM 4690, and $HOME/ MQMFTCredentials.xml on other platforms.

Table 38. SSL properties for the coordination.properties file (continued)

Property name	Description	Default value
coordinationSslKeyStore	Specifies the location of the private key of the commands. The value of coordinationSslKeyStore is a file path. If it is a Windows file path, the backslash character (\) must be escaped (\\). This property is only required if the coordination queue manager requires client authentication. For WebSphere MQ V7.5 or later, the value of this property can contain environment variables.	None
coordinationSslKeyStoreType	The type of SSL keystore you want to use. JKS and PKCS#12 keystores are supported. The value of this property can be either `jks` or `pkcs12`.	jks
coordinationSslKeyStoreCredentialsFile	The path to the file that contains the coordinationSslKeyStore credentials. For WebSphere MQ V7.5 or later, the value of this property can contain environment variables.	The default value for this property is `%USERPROFILE%\ MQMFTCredentials.xml` on Windows, `f:/adxetc/mft75/ mqft/config/ mqmftcredentials.xml` on IBM 4690, and `$HOME/ MQMFTCredentials.xml` on other platforms.
coordinationSslFipsRequired	Specifies that you want to enable FIPS support at the level of the coordination queue manager. The value of this property can be `true` or `false`. For more information, see "FIPS support" on page 691.	false

Table 39. SSL properties for the command.properties file

Property name	Description	Default value
connectionSslCipherSpec	Specifies the protocol, hash algorithm, and encryption algorithm that is used, and how many bits are used in the encryption key, when data is exchanged between the commands and the command queue manager. The value of connectionSslCipherSpec is a CipherSpec name. This CipherSpec name is the same as the CipherSpec name used on the command queue manager channel. A list of valid CipherSpec names is included in SSL CipherSpecs and CipherSuites in WebSphere MQ classes for Java. connectionSslCipherSpec is similar to connectionSslCipherSuite. If both connectionSslCipherSuite and connectionSslCipherSpec are specified, the value of connectionSslCipherSpec is used.	None
connectionSslCipherSuite	Specifies SSL aspects of how the commands and the command queue manager exchange data. The value of connectionSslCipherSuite is a CipherSuite name. The CipherSuite name maps to the CipherSpec name used on the agent queue manager channel. For more information, see CipherSuite and CipherSpec name mappings. connectionSslCipherSuite is similar to connectionSslCipherSpec. If both connectionSslCipherSuite and connectionSslCipherSpec are specified, the value of connectionSslCipherSpec is used.	None
connectionSslPeerName	Specifies a distinguished name skeleton that must match the name that is provided by the command queue manager. The distinguished name is used to check the identifying certificate that is presented by the command queue manager on connection.	None

Table 39. SSL properties for the command.properties file (continued)

Property name	Description	Default value
connectionSslTrustStore	Specifies the location of the certificates that the commands trust. The value of connectionSslTrustStore is a file path. If it is a Windows file path, the backslash character (\) must be escaped (\\). For WebSphere MQ V7.5 or later, the value of this property can contain environment variables.	None
connectionSslTrustStoreType	The type of SSL truststore you want to use. JKS and PKCS#12 keystores are supported. The value of this property can be either `jks` or `pkcs12`.	jks
connectionSslTrustStoreCredentialsFile	The path to the file that contains the connectionSslTrustStore credentials. For WebSphere MQ V7.5 or later, the value of this property can contain environment variables.	The default value for this property is `%USERPROFILE%\ MQMFTCredentials.xml` on Windows, `f:/adxetc/mft75/ mqft/config/ mqmftcredentials.xml` on IBM 4690, and `$HOME/ MQMFTCredentials.xml` on other platforms.
connectionSslKeyStore	Specifies the location of the private key of the commands. The value of connectionSslKeyStore is a file path. If it is a Windows file path, the backslash character (\) must be escaped (\\). This property is only required if the command queue manager requires client authentication. For WebSphere MQ V7.5 or later, the value of this property can contain environment variables.	None
connectionSslKeyStoreType	The type of SSL keystore you want to use. JKS and PKCS#12 keystores are supported. The value of this property can be either `jks` or `pkcs12`. For WebSphere MQ V7.5 or later, the value of this property can contain environment variables.	jks
connectionSslKeyStoreCredentialsFile	The path to the file that contains the connectionSslKeyStore credentials. For WebSphere MQ V7.5 or later, the value of this property can contain environment variables.	The default value for this property is `%USERPROFILE%\ MQMFTCredentials.xml` on Windows, `f:/adxetc/mft75/ mqft/config/ mqmftcredentials.xml` on IBM 4690, and `$HOME/ MQMFTCredentials.xml` on other platforms.
connectionSslFipsRequired	Specifies that you want to enable FIPS support at the level of the command queue manager. The value of this property can be `true` or `false`. For more information, see "FIPS support" on page 691.	false

SHA-2 CipherSpecs and CipherSuites

WebSphere MQ Managed File Transfer supports SHA-2 CipherSpecs and CipherSuites.

To enable use of SHA-2 CipherSpecs and CipherSuites on connections between agents and IBM WebSphere MQ queue managers in IBM WebSphere MQ V7.5.0.2, you must use IBM JREs 6.0 SR13 FP2, 7.0 SR4 FP2, or later.

For more information about CipherSpecs and CipherSuites that are available for connections between agents and IBM WebSphere MQ queue managers, see SSL CipherSpecs and CipherSuites.

SHA-2 connections to 4690 OS are not supported.

Related reference:

"SSL properties" on page 598
Use SSL withWebSphere MQ and WebSphere MQ Managed File Transfer to prevent unauthorized connections between agents and queue managers, and to encrypt message traffic between agents and queue managers.

File logger configuration files

In addition to the `logger.properties` file, a stand-alone file logger also has an XML configuration file in its configuration directory. This configuration file is called `FileLoggerFormat.xml` and it defines the format used by the file logger to write messages to the log file. The content of this file must conform to the XML schema defined in the `FileLoggerFormat.xsd` file.

Stand-alone file logger default log format definition
Default log file format definition for the stand-alone file logger.

```
<?xml version="1.0" encoding="UTF-8"?>
<logFormatDefinition  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  version="1.00" xsi:noNamespaceSchemaLocation="FileLoggerFormat.xsd">
  <messageTypes>
    <callCompleted>
      <format>
        <inserts>
          <insert type="user"   width="19" ignoreNull="false">/transaction/action/@time</insert>
          <insert type="user"   width="48" ignoreNull="false">/transaction/@ID</insert>
          <insert type="system" width="6"  ignoreNull="false">type</insert>
          <insert type="user"   width="3"  ignoreNull="false">/transaction/status/@resultCode</insert>
          <insert type="user"   width="0"  ignoreNull="false">/transaction/agent/@agent</insert>
          <insert type="user"   width="0"  ignoreNull="false">/transaction/agent/@QMgr</insert>
          <insert type="user"   width="0"  ignoreNull="false">/transaction/job/name</insert>
          <insert type="user"   width="0"  ignoreNull="true">/transaction/transferSet/call/command/@type</insert>
          <insert type="user"   width="0"  ignoreNull="true">/transaction/transferSet/call/command/@name</insert>
          <insert type="system" width="0"  ignoreNull="true">callArguments</insert>
          <insert type="user"   width="0"  ignoreNull="true">/transaction/transferSet/call/callResult/@outcome</insert>
          <insert type="user"   width="0"  ignoreNull="true">/transaction/transferSet/call/callResult/result/error</insert>
        </inserts>
        <separator>;</separator>
      </format>
    </callCompleted>
    <callStarted>
      <format>
        <inserts>
          <insert type="user"   width="19" ignoreNull="false">/transaction/action/@time</insert>
          <insert type="user"   width="48" ignoreNull="false">/transaction/@ID</insert>
          <insert type="system" width="6"  ignoreNull="false">type</insert>
          <insert type="user"   width="0"  ignoreNull="false">/transaction/agent/@agent</insert>
          <insert type="user"   width="0"  ignoreNull="false">/transaction/agent/@QMgr</insert>
          <insert type="user"   width="0"  ignoreNull="false">/transaction/job/name</insert>
          <insert type="user"   width="0"  ignoreNull="true">/transaction/transferSet/call/command/@type</insert>
          <insert type="user"   width="0"  ignoreNull="true">/transaction/transferSet/call/command/@name</insert>
          <insert type="system" width="0"  ignoreNull="true">callArguments</insert>
        </inserts>
        <separator>;</separator>
      </format>
    </callStarted>
    <monitorAction>
      <format>
        <inserts>
          <insert type="user"   width="19" ignoreNull="false">/monitorLog/action/@time</insert>
          <insert type="user"   width="48" ignoreNull="false">/monitorLog/@referenceId</insert>
          <insert type="system" width="6"  ignoreNull="false">type</insert>
          <insert type="user"   width="3"  ignoreNull="false">/monitorLog/status/@resultCode</insert>
          <insert type="user"   width="0"  ignoreNull="false">/monitorLog/@monitorName</insert>
          <insert type="user"   width="0"  ignoreNull="false">/monitorLog/monitorAgent/@agent</insert>
          <insert type="user"   width="0"  ignoreNull="false">/monitorLog/monitorAgent/@QMgr</insert>
          <insert type="user"   width="0"  ignoreNull="false">/monitorLog/action</insert>
        </inserts>
        <separator>;</separator>
```

```
          </format>
      </monitorAction>
      <monitorCreate>
        <format>
          <inserts>
            <insert type="user"    width="19" ignoreNull="false">/monitorLog/action/@time</insert>
            <insert type="user"    width="48" ignoreNull="false">/monitorLog/@referenceId</insert>
            <insert type="system" width="6"  ignoreNull="false">type</insert>
            <insert type="user"    width="0"  ignoreNull="false">/monitorLog/@monitorName</insert>
            <insert type="user"    width="0"  ignoreNull="false">/monitorLog/monitorAgent/@agent</insert>
            <insert type="user"    width="0"  ignoreNull="false">/monitorLog/monitorAgent/@QMgr</insert>
            <insert type="user"    width="0"  ignoreNull="false">/monitorLog/action</insert>
          </inserts>
          <separator>;</separator>
        </format>
      </monitorCreate>
      <monitorFired>
        <format>
          <inserts>
            <insert type="user"    width="19" ignoreNull="false">/monitorLog/action/@time</insert>
            <insert type="user"    width="48" ignoreNull="false">/monitorLog/@referenceId</insert>
            <insert type="system" width="6"  ignoreNull="false">type</insert>
            <insert type="user"    width="3"  ignoreNull="false">/monitorLog/status/@resultCode</insert>
            <insert type="user"    width="0"  ignoreNull="false">/monitorLog/@monitorName</insert>
            <insert type="user"    width="0"  ignoreNull="false">/monitorLog/monitorAgent/@agent</insert>
            <insert type="user"    width="0"  ignoreNull="false">/monitorLog/monitorAgent/@QMgr</insert>
            <insert type="user"    width="0"  ignoreNull="false">/monitorLog/action</insert>
            <insert type="user"    width="48" ignoreNull="false">/monitorLog/references/taskRequest</insert>
          </inserts>
          <separator>;</separator>
        </format>
      </monitorFired>
      <notAuthorized>
        <format>
          <inserts>
            <insert type="user"    width="19" ignoreNull="false">/notAuthorized/action/@time</insert>
            <insert type="user"    width="48" ignoreNull="false">/notAuthorized/@ID</insert>
            <insert type="system" width="6"  ignoreNull="false">type</insert>
            <insert type="user"    width="3"  ignoreNull="false">/notAuthorized/status/@resultCode</insert>
            <insert type="user"    width="12" ignoreNull="false">/notAuthorized/action</insert>
            <insert type="user"    width="12" ignoreNull="false">/notAuthorized/authority</insert>
            <insert type="user"    width="0"  ignoreNull="false">/notAuthorized/originator/userID</insert>
            <insert type="user"    width="0"  ignoreNull="false">/notAuthorized/status/supplement</insert>
          </inserts>
          <separator>;</separator>
        </format>
      </notAuthorized>
      <scheduleDelete>
        <format>
          <inserts>
            <insert type="user"    width="19" ignoreNull="false">/schedulelog/action/@time</insert>
            <insert type="user"    width="48" ignoreNull="false">/schedulelog/@ID</insert>
            <insert type="system" width="6"  ignoreNull="false">type</insert>
            <insert type="user"    width="3"  ignoreNull="false">/schedulelog/status/@resultCode</insert>
            <insert type="user"    width="0"  ignoreNull="false">/schedulelog/sourceAgent/@agent</insert>
            <insert type="user"    width="12" ignoreNull="false">/schedulelog/action</insert>
            <insert type="user"    width="0"  ignoreNull="false">/schedulelog/originator/userID</insert>
            <insert type="user"    width="0"  ignoreNull="true">/schedulelog/status/supplement</insert>
          </inserts>
          <separator>;</separator>
        </format>
      </scheduleDelete>
      <scheduleExpire>
        <format>
          <inserts>
            <insert type="user"    width="19" ignoreNull="false">/schedulelog/action/@time</insert>
            <insert type="user"    width="48" ignoreNull="false">/schedulelog/@ID</insert>
            <insert type="system" width="6"  ignoreNull="false">type</insert>
            <insert type="user"    width="3"  ignoreNull="false">/schedulelog/status/@resultCode</insert>
            <insert type="user"    width="0"  ignoreNull="false">/schedulelog/sourceAgent/@agent</insert>
            <insert type="user"    width="12" ignoreNull="false">/schedulelog/action</insert>
            <insert type="user"    width="0"  ignoreNull="false">/schedulelog/originator/userID</insert>
            <insert type="user"    width="0"  ignoreNull="true">/schedulelog/status/supplement</insert>
          </inserts>
          <separator>;</separator>
        </format>
      </scheduleExpire>
      <scheduleSkipped>
        <format>
```

```
     <inserts>
       <insert type="user"   width="19" ignoreNull="false">/schedulelog/action/@time</insert>
       <insert type="user"   width="48" ignoreNull="false">/schedulelog/@ID</insert>
       <insert type="system" width="6"  ignoreNull="false">type</insert>
       <insert type="user"   width="3"  ignoreNull="false">/schedulelog/status/@resultCode</insert>
       <insert type="user"   width="0"  ignoreNull="false">/schedulelog/sourceAgent/@agent</insert>
       <insert type="user"   width="12" ignoreNull="false">/schedulelog/action</insert>
       <insert type="user"   width="0"  ignoreNull="false">/schedulelog/originator/userID</insert>
       <insert type="user"   width="0"  ignoreNull="true">/schedulelog/status/supplement</insert>
     </inserts>
     <separator>;</separator>
   </format>
</scheduleSkipped>
<scheduleSubmitInfo>
   <format>
     <inserts>
       <insert type="user"   width="19" ignoreNull="false">/schedulelog/action/@time</insert>
       <insert type="user"   width="48" ignoreNull="false">/schedulelog/@ID</insert>
       <insert type="system" width="6"  ignoreNull="false">type</insert>
       <insert type="user"   width="3"  ignoreNull="false">/schedulelog/status/@resultCode</insert>
       <insert type="user"   width="0"  ignoreNull="false">/schedulelog/sourceAgent/@agent</insert>
       <insert type="user"   width="12" ignoreNull="false">/schedulelog/action</insert>
       <insert type="user"   width="0"  ignoreNull="false">/schedulelog/originator/userID</insert>
       <insert type="user"   width="0"  ignoreNull="true">/schedulelog/schedule/submit/</insert>
       <insert type="user"   width="0"  ignoreNull="true">/schedulelog/schedule/submit/@timezone</insert>
       <insert type="user"   width="3"  ignoreNull="true">/schedulelog/schedule/repeat/frequency</insert>
       <insert type="user"   width="12" ignoreNull="true">/schedulelog/schedule/repeat/frequency/@interval</insert>
       <insert type="user"   width="3"  ignoreNull="true">/schedulelog/schedule/repeat/expireCount</insert>
       <insert type="user"   width="0"  ignoreNull="true">/schedulelog/status/supplement</insert>
     </inserts>
     <separator>;</separator>
   </format>
</scheduleSubmitInfo>
<scheduleSubmitTransfer>
   <format>
     <inserts>
       <insert type="user"   width="19" ignoreNull="false">/schedulelog/action/@time</insert>
       <insert type="user"   width="48" ignoreNull="false">/schedulelog/@ID</insert>
       <insert type="system" width="10" ignoreNull="false">type</insert>
       <insert type="user"   width="0"  ignoreNull="false">/transaction/sourceAgent/@agent |
       /transaction/sourceWebUser/@webGatewayAgentName |
       /transaction/sourceWebGateway/@webGatewayAgentName</insert>
       <insert type="user"   width="0"  ignoreNull="false">/transaction/sourceAgent/@QMgr |
        /transaction/sourceWebUser/@webGatewayAgentQMgr |
       /transaction/sourceWebGateway/@webGatewayAgentQMgr</insert>
       <insert type="user"   width="0"  ignoreNull="false">/transaction/destinationAgent/@agent |
       /transaction/destinationWebUser/@webGatewayAgentName |
       /transaction/destinationWebGateway/@webGatewayAgentName</insert>
       <insert type="user"   width="0"  ignoreNull="false">/transaction/destinationAgent/@QMgr |
        /transaction/destinationWebUser/@webGatewayAgentQMgr |
       /transaction/destinationWebGateway/@webGatewayAgentQMgr</insert>
     </inserts>
     <separator>;</separator>
   </format>
</scheduleSubmitTransfer>
<scheduleSubmitTransferSet>
   <format>
     <inserts>
       <insert type="user"   width="19" ignoreNull="false">/schedulelog/action/@time</insert>
       <insert type="user"   width="48" ignoreNull="false">/schedulelog/@ID</insert>
       <insert type="system" width="10" ignoreNull="false">type</insert>
       <insert type="user"   width="0"  ignoreNull="false">source/file | source/queue</insert>
       <insert type="user"   width="5"  ignoreNull="true">source/@type</insert>
       <insert type="user"   width="6"  ignoreNull="true">source/@disposition</insert>
       <insert type="user"   width="0"  ignoreNull="false">destination/file | destination/queue</insert>
       <insert type="user"   width="5"  ignoreNull="true">destination/@type</insert>
       <insert type="user"   width="9"  ignoreNull="true">destination/@exist</insert>
     </inserts>
     <separator>;</separator>
   </format>
</scheduleSubmitTransferSet>
<transferStarted>
   <format>
     <inserts>
       <insert type="user"   width="19" ignoreNull="false">/transaction/action/@time</insert>
       <insert type="user"   width="48" ignoreNull="false">/transaction/@ID</insert>
       <insert type="system" width="6"  ignoreNull="false">type</insert>
       <insert type="user"   width="3"  ignoreNull="true">/transaction/status/@resultCode</insert>
       <insert type="user"   width="0"  ignoreNull="false">/transaction/sourceAgent/@agent |
```

```
            /transaction/sourceWebUser/@webGatewayAgentName |
         /transaction/sourceWebGateway/@webGatewayAgentName</insert>
            <insert type="user"   width="0"  ignoreNull="true">/transaction/sourceAgent/@QMgr |
      /transaction/sourceWebUser/@webGatewayAgentQMgr |
       /transaction/sourceWebGateway/@webGatewayAgentQMgr</insert>
            <insert type="user"   width="0"  ignoreNull="true">/transaction/sourceAgent/@agentType |
      /transaction/sourceWebUser/@webGatewayAgentType |
      /transaction/sourceWebGateway/@webGatewayAgentType</insert>
            <insert type="user"   width="0"  ignoreNull="false">/transaction/destinationAgent/@agent |
      /transaction/destinationWebUser/@webGatewayAgentName |
       /transaction/destinationWebGateway/@webGatewayAgentName</insert>
            <insert type="user"   width="0"  ignoreNull="true">/transaction/destinationAgent/@QMgr |
      /transaction/destinationWebUser/@webGatewayAgentQMgr |
      /transaction/destinationWebGateway/@webGatewayAgentQMgr</insert>
            <insert type="user"   width="0"  ignoreNull="true">/transaction/originator/userID</insert>
            <insert type="user"   width="0"  ignoreNull="true">/transaction/job/name</insert>
            <insert type="user"   width="0"  ignoreNull="true">/transaction/scheduleLog/@ID</insert>
          </inserts>
          <separator>;</separator>
        </format>
    </transferStarted>
    <transferCancelled>
      <format>
        <inserts>
          <insert type="user"    width="19" ignoreNull="false">/transaction/action/@time</insert>
          <insert type="user"    width="48" ignoreNull="false">/transaction/@ID</insert>
          <insert type="system" width="6"  ignoreNull="false">type</insert>
          <insert type="user"    width="3"  ignoreNull="true">/transaction/status/@resultCode</insert>
          <insert type="user"    width="0"  ignoreNull="false">/transaction/sourceAgent/@agent |
      /transaction/sourceWebUser/@webGatewayAgentName |
      /transaction/sourceWebGateway/@webGatewayAgentName</insert>
            <insert type="user"   width="0"  ignoreNull="true">/transaction/sourceAgent/@QMgr |
      /transaction/sourceWebUser/@webGatewayAgentQMgr |
      /transaction/sourceWebGateway/@webGatewayAgentQMgr</insert>
            <insert type="user"   width="0"  ignoreNull="true">/transaction/sourceAgent/@agentType |
      /transaction/sourceWebUser/@webGatewayAgentType |
      /transaction/sourceWebGateway/@webGatewayAgentType</insert>
            <insert type="user"   width="0"  ignoreNull="false">/transaction/destinationAgent/@agent |
    /transaction/destinationWebUser/@webGatewayAgentName |
    /transaction/destinationWebGateway/@webGatewayAgentName</insert>
            <insert type="user"   width="0"  ignoreNull="true">/transaction/destinationAgent/@QMgr |
    /transaction/destinationWebUser/@webGatewayAgentQMgr |
    /transaction/destinationWebGateway/@webGatewayAgentQMgr</insert>
            <insert type="user"   width="0"  ignoreNull="true">/transaction/destinationAgent/@agentType |
  /transaction/destinationWebUser/@webGatewayAgentType |
  /transaction/destinationWebGateway/@webGatewayAgentType</insert>
            <insert type="user"   width="0"  ignoreNull="true">/transaction/originator/userID</insert>
            <insert type="user"   width="0"  ignoreNull="true">/transaction/job/name</insert>
            <insert type="user"   width="0"  ignoreNull="true">/transaction/status/supplement</insert>
          </inserts>
          <separator>;</separator>
        </format>
    </transferCancelled>
    <transferComplete>
      <format>
        <inserts>
          <insert type="user"    width="19" ignoreNull="false">/transaction/action/@time</insert>
          <insert type="user"    width="48" ignoreNull="false">/transaction/@ID</insert>
          <insert type="system" width="6"  ignoreNull="false">type</insert>
          <insert type="user"    width="3"  ignoreNull="true">/transaction/status/@resultCode</insert>
          <insert type="user"    width="0"  ignoreNull="false">/transaction/sourceAgent/@agent |
           /transaction/sourceWebUser/@webGatewayAgentName |
          /transaction/sourceWebGateway/@webGatewayAgentName</insert>
          <insert type="user"    width="0"  ignoreNull="true">/transaction/sourceAgent/@QMgr |
          /transaction/sourceWebUser/@webGatewayAgentQMgr |
          /transaction/sourceWebGateway/@webGatewayAgentQMgr</insert>
          <insert type="user"    width="0"  ignoreNull="true">/transaction/sourceAgent/@agentType |
          /transaction/sourceWebUser/@webGatewayAgentType |
          /transaction/sourceWebGateway/@webGatewayAgentType</insert>
          <insert type="user"    width="0"  ignoreNull="false">/transaction/destinationAgent/@agent |
          /transaction/destinationWebUser/@webGatewayAgentName |
          /transaction/destinationWebGateway/@webGatewayAgentName</insert>
          <insert type="user"    width="0"  ignoreNull="true">/transaction/destinationAgent/@QMgr |
          /transaction/destinationWebUser/@webGatewayAgentQMgr |
          /transaction/destinationWebGateway/@webGatewayAgentQMgr</insert>
          <insert type="user"    width="0"  ignoreNull="true">/transaction/destinationAgent/@agentType |
          /transaction/destinationWebUser/@webGatewayAgentType |
          /transaction/destinationWebGateway/@webGatewayAgentType</insert>
          <insert type="user"    width="0"  ignoreNull="true">/transaction/originator/userID</insert>
```

```
          <insert type="user"   width="0"  ignoreNull="true">/transaction/job/name</insert>
          <insert type="user"   width="0"  ignoreNull="true">/transaction/status/supplement</insert>
        </inserts>
        <separator>;</separator>
      </format>
    </transferComplete>
    <transferDelete>
      <format>
        <inserts>
          <insert type="user"   width="19" ignoreNull="false">/transaction/action/@time</insert>
          <insert type="user"   width="48" ignoreNull="false">/transaction/@ID</insert>
          <insert type="system" width="6"  ignoreNull="false">type</insert>
          <insert type="user"   width="3"  ignoreNull="true">/transaction/status/@resultCode</insert>
          <insert type="user"   width="0"  ignoreNull="false">/transaction/sourceAgent/@agent |
          /transaction/sourceWebUser/@webGatewayAgentName |
          /transaction/sourceWebGateway/@webGatewayAgentName</insert>
          <insert type="user"   width="0"  ignoreNull="true">/transaction/sourceAgent/@QMgr |
          /transaction/sourceWebUser/@webGatewayAgentQMgr |
          /transaction/sourceWebGateway/@webGatewayAgentQMgr</insert>
          <insert type="user"   width="0"  ignoreNull="true">/transaction/sourceAgent/@agentType |
          /transaction/sourceWebUser/@webGatewayAgentType |
          /transaction/sourceWebGateway/@webGatewayAgentType</insert>
          <insert type="user"   width="0"  ignoreNull="false">/transaction/destinationAgent/@agent |
          /transaction/destinationWebUser/@webGatewayAgentName |
          /transaction/destinationWebGateway/@webGatewayAgentName</insert>
          <insert type="user"   width="0"  ignoreNull="true">/transaction/destinationAgent/@QMgr |
          /transaction/destinationWebUser/@webGatewayAgentQMgr |
          /transaction/destinationWebGateway/@webGatewayAgentQMgr</insert>
          <insert type="user"   width="0"  ignoreNull="true">/transaction/destinationAgent/@agentType |
          /transaction/destinationWebUser/@webGatewayAgentType |
          /transaction/destinationWebGateway/@webGatewayAgentType</insert>
          <insert type="user"   width="0"  ignoreNull="true">/transaction/originator/userID</insert>
          <insert type="user"   width="0"  ignoreNull="true">/transaction/job/name</insert>
          <insert type="user"   width="0"  ignoreNull="true">/transaction/status/supplement</insert>
        </inserts>
        <separator>;</separator>
      </format>
    </transferDelete>
    <transferProgress>
      <format>
        <inserts>
          <insert type="user"   width="19" ignoreNull="false">/transaction/action/@time</insert>
          <insert type="user"   width="48" ignoreNull="false">/transaction/@ID</insert>
          <insert type="system" width="6"  ignoreNull="false">type</insert>
          <insert type="user"   width="3"  ignoreNull="true">status/@resultCode</insert>
          <insert type="user"   width="0"  ignoreNull="false">source/file | source/queue</insert>
          <insert type="user"   width="0"  ignoreNull="false">source/file/@size | source/queue/@size</insert>
          <insert type="user"   width="5"  ignoreNull="true">source/@type</insert>
          <insert type="user"   width="6"  ignoreNull="true">source/@disposition</insert>
          <insert type="user"   width="0"  ignoreNull="true">source/file/@alias | source/queue/@alias</insert>
          <insert type="user"   width="0"  ignoreNull="true">source/file/@filespace | source/queue/@filespace</insert>
          <insert type="user"   width="0"  ignoreNull="true">source/@correlationBoolean1</insert>
          <insert type="user"   width="0"  ignoreNull="true">source/@correlationNum1</insert>
          <insert type="user"   width="0"  ignoreNull="true">source/@correlationString1</insert>
          <insert type="user"   width="0"  ignoreNull="false">destination/file | destination/queue</insert>
          <insert type="user"   width="0"  ignoreNull="false">destination/file/@size | destination/queue/@size</insert>
          <insert type="user"   width="5"  ignoreNull="true">destination/@type</insert>
          <insert type="user"   width="9"  ignoreNull="true">destination/@exist</insert>
          <insert type="user"   width="0"  ignoreNull="true">destination/file/@alias | destination/queue/@alias</insert>
          <insert type="user"   width="0"  ignoreNull="true">destination/file/@filespace | destination/queue/@filespace</insert>

          <insert type="user"   width="0"  ignoreNull="true">destination/@correlationBoolean1</insert>
          <insert type="user"   width="0"  ignoreNull="true">destination/@correlationNum1</insert>
          <insert type="user"   width="0"  ignoreNull="true">destination/@correlationString1</insert>
          <insert type="user"   width="0"  ignoreNull="true">status/supplement</insert>
        </inserts>
        <separator>;</separator>
      </format>
    </transferProgress>
  </messageTypes>
</logFormatDefinition>
```

Related reference:

"Stand-alone file logger format" on page 135

The format of message information written by the file logger can be defined in the `FileLoggerFormat.xml` file.

"Stand-alone file logger format XSD"

The schema for a stand-alone file format.

Stand-alone file logger format XSD

The schema for a stand-alone file format.

Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
@start_non_restricted_prolog@
Version: %Z% %I% %W% %E% %U% [%H% %T%]

Licensed Materials - Property of IBM

5724-H72

Copyright IBM Corp. 2011 All Rights Reserved.

US Government Users Restricted Rights - Use, duplication or
disclosure restricted by GSA ADP Schedule Contract with
IBM Corp.
@end_non_restricted_prolog@
 -->


<!--
    This schema defines the format of the FileLoggerFormat XML file that contains the definition
    of the format to use when logging FTE log messages to a file. When an XML file that conforms
    to this schema is processed by a file logger it can contain definitions for one or more
    message type(s) that define how log messages of those types are output to the file log.
-->

<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

<xsd:include schemaLocation="fteutils.xsd"/>

    <!--
        Defines the logFileDefinition and version number
        <logFileDefinition version="1.00" ...
            <messageTypes>
                ...
            </messageTypes>
        </logFileDefinition>
     -->
    <xsd:element name="logFileDefinition">
        <xsd:complexType>
            <xsd:sequence>
             <xsd:element    name="messageTypes" type="messageTypesType" maxOccurs="1" minOccurs="1"/>
          </xsd:sequence>
            <xsd:attribute name="version"      type="versionType"      use="required"/>
        </xsd:complexType>
    </xsd:element>

    <!--
        Defines the set of accepted message types. The definition of individual message types
        is optional. If a particular types element is present but empty then no line will be
        output for messages of that type. If a particular types element is not present then
        the default format will be used to format messages of that type.
     -->
    <xsd:complexType name="messageTypesType">
        <xsd:sequence>
            <xsd:element name="callCompleted"            type="messageType"  maxOccurs="1" minOccurs="0"/>
            <xsd:element name="callStarted"              type="messageType"  maxOccurs="1" minOccurs="0"/>
            <xsd:element name="monitorAction"            type="messageType"  maxOccurs="1" minOccurs="0"/>
            <xsd:element name="monitorCreate"            type="messageType"  maxOccurs="1" minOccurs="0"/>
            <xsd:element name="monitorFired"             type="messageType"  maxOccurs="1" minOccurs="0"/>
            <xsd:element name="notAuthorized"            type="messageType"  maxOccurs="1" minOccurs="0"/>
            <xsd:element name="scheduleDelete"           type="messageType"  maxOccurs="1" minOccurs="0"/>
            <xsd:element name="scheduleExpire"           type="messageType"  maxOccurs="1" minOccurs="0"/>
            <xsd:element name="scheduleSkipped"          type="messageType"  maxOccurs="1" minOccurs="0"/>
```

```xml
              <xsd:element name="scheduleSubmitInfo"        type="messageType"  maxOccurs="1" minOccurs="0"/>
              <xsd:element name="scheduleSubmitTransfer"     type="messageType"  maxOccurs="1" minOccurs="0"/>
              <xsd:element name="scheduleSubmitTransferSet"  type="messageType"  maxOccurs="1" minOccurs="0"/>
              <xsd:element name="transferStarted"           type="messageType"  maxOccurs="1" minOccurs="0"/>
              <xsd:element name="transferCancelled"         type="messageType"  maxOccurs="1" minOccurs="0"/>
              <xsd:element name="transferComplete"          type="messageType"  maxOccurs="1" minOccurs="0"/>
              <xsd:element name="transferDelete"            type="messageType"  maxOccurs="1" minOccurs="0"/>
              <xsd:element name="transferProgress"          type="messageType"  maxOccurs="1" minOccurs="0"/>
        </xsd:sequence>
</xsd:complexType>

<!--
    Defines the content of a message type definition e.g.

    <callStarted>
     <format>
      ...
     </format>
    <callStarted>
 -->
<xsd:complexType name="messageType">
    <xsd:sequence>
        <xsd:element name="format"    type="messageFormatType"  maxOccurs="1" minOccurs="0"/>
    </xsd:sequence>
</xsd:complexType>

<!--
    Defines the content of a message format definition e.g.

     <format>
      <inserts>
       ...
      </inserts
      <separator>;</separator>
     </format>
 -->
<xsd:complexType name="messageFormatType">
    <xsd:sequence>
        <xsd:element name="inserts"    type="insertsType"  maxOccurs="1" minOccurs="1"/>
        <xsd:element name="separator" type="scheduleType" maxOccurs="1" minOccurs="1"/>
    </xsd:sequence>
</xsd:complexType>

<!--
    Defines the content of the inserts element e.g.

    <inserts>
     <insert ...>
     <insert ...>
     ...
    </inserts>
 -->
<xsd:complexType name="insertsType">
    <xsd:sequence>
       <xsd:element name="insert" type="insertType" maxOccurs="unbounded" minOccurs="1"/>
    </xsd:sequence>
</xsd:complexType>

<!--
    Defines the content of an insert definition e.g.

    <insert type="user" width="0" ignoreNull="true">/transaction/@ID</insert>
 -->
<xsd:complexType name="insertType">
    <xsd:attribute name="type"       type="insertTypeType"         use="required"/>
    <xsd:attribute name="width"      type="xsd:nonNegativeInteger" use="required"/>
    <xsd:attribute name="ignoreNull" type="xsd:boolean"            use="required"/>
</xsd:complexType>

<!--
    Defines the accepted choices for the insert type attribute.
 -->
<xsd:simpleType name="insertTypeType">
    <xsd:restriction base="xsd:token">
        <xsd:enumeration value="user"/>
        <xsd:enumeration value="system"/>
```

```
        </xsd:restriction>
    </xsd:simpleType>
```

```
</xsd:schema>
```

Related reference:

"Stand-alone file logger format" on page 135
The format of message information written by the file logger can be defined in the `FileLoggerFormat.xml` file.

"Stand-alone file logger default log format definition" on page 603
Default log file format definition for the stand-alone file logger.

The SYSTEM.FTE topic

The SYSTEM.FTE topic is a topic on the coordination queue manager that WebSphere MQ Managed File Transfer uses to log transfers and store information about agents, monitors, schedules, and templates.

Topic structure

```
SYSTEM.FTE
    /Agents
        /agent_name
    /monitors
        /agent_name
    /Scheduler
        /agent_name
    /Templates
        /template_ID
    /Transfers
        /agent_name
            /transfer_ID
    /Log
        /agent_name
            /Monitors
            /schedule_ID
            /transfer_ID
```

SYSTEM.FTE/Agents/*agent_name*
>   This topic contains a retained publication that describes an agent in your WebSphere MQ
>   Managed File Transfer network and its properties. The message on this topic is updated
>   periodically with the agent status. For more information, see "Agent status message format" on
>   page 611.

SYSTEM.FTE/monitors/*agent_name*
>   This topic contains retained publications that describe the resource monitors associated with the
>   agent *agent_name*. The XML of the retained publication conforms to the schema `MonitorList.xsd`.
>   For more information, see "Monitor list message format" on page 613.

SYSTEM.FTE/Scheduler/*agent_name*
>   This topic contains a retained publication that describes all of the active schedules that are
>   associated with the agent *agent_name*. The XML of the retained publication conforms to the
>   schema `ScheduleList.xsd`. For more information, see "Schedule list message format" on page 618.

SYSTEM.FTE/Templates
>   This topic contains retained publications that describe all of the templates that are defined in
>   your WebSphere MQ Managed File Transfer topology.
>
>   • The publication that is associated with each template is published to a subtopic with the name
>   `SYSTEM.FTE/Templates/`*template_ID*.
>
>   For an example of the contents of this retained publication, see "Example template XML
>   message" on page 622.

SYSTEM.FTE/Transfers/*agent_name*
>   This topic contains publications that describe that status of transfers that originate at the agent

agent_name. The publications that are associated with each transfer are published to a subtopic with the name SYSTEM.FTE/Transfers/*agent_name*/*transfer_ID*. These publications are used by the WebSphere MQ Explorer plug-in to provide progress information about individual transfers. The XML of the publication conforms to the schema `TransferStatus.xsd`. For more information, see "File transfer status message format" on page 623.

SYSTEM.FTE/Log/*agent_name*

This topic contains publications that log information about transfers, monitors, and schedules that originate at the agent *agent_name*. These publications can be logged by the database logger to provide audit records of the events that happen in your WebSphere MQ Managed File Transfer network.

- The publications that are associated with each transfer are published to a subtopic with the name SYSTEM.FTE/Log/*agent_name*/*transfer_ID* and the XML of the publication conforms to the schema `TransferLog.xsd`. For more information, see "File transfer log message formats" on page 626.
- The publications that are associated with each scheduled transfer are published to a subtopic with the name SYSTEM.FTE/Log/*agent_name*/*schedule_ID* and the XML of the publication conforms to the schema `ScheduleLog.xsd`. For more information, see "Scheduled transfer log message formats" on page 650.
- The publications that are associated with each monitor are published to a subtopic with the name SYSTEM.FTE/Log/*agent_name*/Monitors/*monitor_name*/*monitor_ID* and the XML of the publication conforms to the schema `MonitorLog.xsd`. For more information, see "Monitor log message format" on page 656.

Agent status message format

When an agent is created or started, the agent publishes its details to the SYSTEM.FTE topic on its coordination queue manager (on the SYSTEM.FTE/Agents/*agent name* topic).

The following information is included:
- Agent name
- Platform the agent is running on
- Agent description (if provided)
- Agent's queue manager
- Time zone that the agent is running in
- Agent version
- Agent transfer limits
- State of each of the agent's current transfers. These states are listed in Agent transfer states
- Type of agent

If the agent is a protocol bridge agent the following information is also included:
- Type of protocol bridge agent
- Host name or IP address of the protocol bridge server

If the agent is a web agent the following information is also included:
- Name of the Web Gateway the web agent connects to

The agent status is republished whenever the agent transfer states change, but by default no more than every 30 seconds. You can change this default setting using the agentStatusPublishRateLimit agent property, which is described in: Advanced agent properties.

The following example output shows the keys used for each data element in the agent status:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">
<properties>
    <entry key="agentOsName">Windows 7</entry>
    <entry key="agentDescription"/>
    <entry key="queueManager">QM1</entry>
    <entry key="agentTimeZone">Europe/London</entry>
    <entry key="agentVersion">1.00</entry>
    <entry key="agentName">FTEAGENT</entry>
    <entry key="maxDestinationTransfers">25</entry>
    <entry key="maxSourceTransfers">25</entry>
    <entry key="maxQueuedTransfers">100</entry>
    <entry key="DestinationTransferStates">414d51204d554e474f20202020202020d857374a60a72622=RunningTransfer
                                           414d51204d554e474f20202020202020d857374a69a72622=RunningTransfer
                                           414d51204d554e474f20202020202020d857374a75a72622=RunningTransfer
    </entry>
    <entry key="SourceTransferStates">414d51204d554e474f20202020202020d857374a93a72622=NegotiatingTransfer
                                      414d51204d554e474f20202020202020d857374a78a72622=RunningTransfer
                                      414d51204d554e474f20202020202020d857374aaba72622=NewSenderTransfer
                                      414d51204d554e474f20202020202020d857374a63a72622=RunningTransfer
    </entry>
</properties>
```

The following example output shows the keys used for each data element in the agent status of a protocol bridge agent:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">
<properties>
    <entry key="agentOsName">Windows 7</entry>
    <entry key="agentDescription"/>
    <entry key="queueManager">QM1</entry>
    <entry key="agentTimeZone">Europe/London</entry>
    <entry key="agentVersion">1.00</entry>
    <entry key="agentName">BRIDGE</entry>
    <entry key="protocolBridgeType">ftp</entry>
    <entry key="protocolBridgeServerHost">ftpserver.example.org</entry>
    <entry key="maxDestinationTransfers">25</entry>
    <entry key="maxSourceTransfers">25</entry>
    <entry key="maxQueuedTransfers">100</entry>
    <entry key="DestinationTransferStates">414d51204d554e474f20202020202020d857374a60a72622=RunningTransfer
    </entry>
    <entry key="SourceTransferStates">414d51204d554e474f20202020202020d857374a93a72622=NegotiatingTransfer
    </entry>
</properties>
```

Agent transfer states:

An agent that is started publishes its details to the SYSTEM.FTE topic on its coordination queue manager. These details include the states of each of the current transfers that involved that agent. The states are as follows:

Transfer state	Explanation
NewSenderTransfer	A new transfer from the source agent that the negotiation has not started for.
NewReceiverTransfer	A new transfer has been created at the destination agent as part of negotiation, but the transfer is not yet running.
NegotiatingTransfer	A source agent is in negotiation with the destination agent before running a transfer.
RunningTransfer	A transfer from either a source agent or destination agent that is in the normal running state

Transfer state	Explanation
RecoveringTransfer	When either a source or destination agent starts the recovery process, any transfers in running state are moved into transfer state. Transfers are moved out of this state into ReSynchronisingTransfer state when a resynchronization message is sent to the peer agent. For example, if the destination agent starts the recovery process for a running transfer, the transfer is moved into the ReSynchronisingTransfer state when a resynchronization message is sent to its source agent.
ReSynchronisingTransfer	A transfer source or destination agent has found a problem and has sent a resynchronization message to its respective destination or source agent.
CompletedTransfer	A destination agent has completed the transfer and has sent a completion message to the source agent. The destination agent is waiting for an acknowledgment message from the source agent.
CompleteReceivedTransfer	A source agent has received a completion message from the destination agent and has sent a message back to the destination agent to acknowledge the completion message.
CancelledNewTransfer	A source agent has received a cancel message for a new transfer.
CancelledInProgressTransfer	A source agent has received a cancel message for an in-progress transfer.
ResumingTransfer	A source agent has received a resynchronize response message and now schedules the transfer to restart.
RestartingTransfer	A source or destination agent has received a resynchronize request message and is waiting for the respective destination or source agent to restart.
WaitingForDestinationCapacity	A source agent has received a DESTINATION_CAPACITY_EXCEEDED error from the destination agent. The transfer is now in a waiting state to be retried after a period.
FailedTransferEnding	The transfer has failed but the completion log message has not been published and the transfer has not been removed from the state store. For example, this state can occur if an agent process is stopped after a failure response has been received from the destination agent but before the subsequent processing has been completed.

Monitor list message format

The XML messages that are published as retained publications to the topic string SYSTEM.FTE/
monitors/*agent_name*/*monitor_name* conform to the MonitorList.xsd schema. Each XML message lists an
active monitor belonging to that agent. This information is used by the **fteListMonitors** command and
the WebSphere MQ Explorer plug-in to display a list of monitors to the user. The MonitorList.xsd
schema document is located in the *MQ_INSTALLATION_PATH*/mqft/samples/schema directory. The
MonitorList.xsd schema imports Monitor.xsd, which is in the same directory.

Schema

The following schema describes which elements are valid in a monitor list XML message.

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        targetNamespace="http://www.ibm.com/xmlns/wmqfte/7.0.1/MonitorDefinition"
        xmlns="http://www.ibm.com/xmlns/wmqfte/7.0.1/MonitorDefinition">

    <xsd:include schemaLocation="Monitor.xsd"/>

    <xsd:element name="monitorList">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="status"        type="monitorStatusType"        minOccurs="1" maxOccurs="1"/>
                <xsd:element name="configuration" type="monitorConfigurationType"  minOccurs="1" maxOccurs="1"/>
```

```
                    <xsd:element name="pollInterval"  type="pollIntervalType"           minOccurs="1" maxOccurs="1"/>
                    <xsd:element name="batch"          type="batchType"                  minOccurs="1" maxOccurs="1"/>
                    <xsd:any minOccurs="0"/>
            </xsd:sequence>
            <xsd:attribute name="version" type="versionType" use="required"/>
            <xsd:attribute name="agent"   type="xsd:string"  use="required"/>
            <xsd:attribute name="monitor" type="xsd:string"  use="required"/>
        </xsd:complexType>
</xsd:element>

<xsd:complexType name="monitorStatusType">
        <xsd:sequence>
            <xsd:any minOccurs="0"/>
        </xsd:sequence>
        <xsd:attribute name="state"        type="xsd:token"/>
        <xsd:anyAttribute/>
</xsd:complexType>

<xsd:complexType name="monitorConfigurationType">
        <xsd:sequence>
            <xsd:element name="description" type="xsd:string"           minOccurs="1" maxOccurs="1" />
            <xsd:element name="resources"   type="monitorResourcesType" minOccurs="0" maxOccurs="1" />
            <xsd:element name="triggerMatch" type="triggerMatchType"      minOccurs="0" maxOccurs="1" />
            <xsd:element name="tasks"        type="monitorListTasksType" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:anyAttribute/>
</xsd:complexType>

<xsd:complexType name="monitorListTasksType">
        <xsd:sequence>
            <xsd:element name="task" type="monitorListTaskType"          minOccurs="1" maxOccurs="1" />
        </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="monitorListTaskType">
        <xsd:sequence>
            <xsd:element name="name"        type="monitorTaskNameType"  minOccurs="0" maxOccurs="1" />
            <xsd:element name="description" type="xsd:string"           minOccurs="0" maxOccurs="1" />
            <xsd:element name="taskXML"     type="xsd:string"           minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
</xsd:complexType>

</xsd:schema>
```

Understanding the monitor list message

The elements and attributes used in the monitor list messages are described in the following list:

`<monitorList>`
Group element containing the elements describe a monitor that is defined for the agent.

Attribute	Description
agent	Required. The name of the agent that the resource monitor is defined on.
monitor	Required. The name of the monitor. Unique for this agent.
version	Required. The version of the monitor list message format.

`<status>`
The status of the monitor.

Attribute	Description
state	The state of the monitor.

`<configuration>`
Group element containing the elements describe the configuration of the monitor.

> **`<description>`**
> A description of the monitor. (Not currently used.)

<resources>
The resource or resources being monitored.

<directory>
A directory to monitor.

Attribute	Description
recursionLevel	The number of directory levels down from the top level to monitor.
id	The ID of the resource.

<queue>
A queue to monitor.

Attribute	Description
id	The ID of the resource.

<triggerMatch>
Element that contains the <conditions> element.

<conditions>
Element that contains the condition or conditions that the resource monitor is monitoring for. This element can contain only one of the following elements: <allOf>, <anyOf>, or <condition>.

<allOf>
Element that contains the condition or conditions that the resource monitor is monitoring for. This element can contain one or many <condition> elements. For the resource monitor to be triggered all of the conditions inside of this element must be met.

<anyOf>
Element that contains the condition or conditions that the resource monitor is monitoring for. This element can contain one or many <condition> elements. For the resource monitor to be triggered only one of the conditions inside of this element must be met.

<condition>
Element that contains a single condition that the resource monitor is monitoring for. This element can contain only one of the following elements: <fileMatch>, <fileNoMatch>, <fileSize>, <queueNotEmpty>, <completeGroups>, or <fileSizeSame>. It can also contain a <name> element and a <resource> element.

If the resource that is being monitored is a directory, one of the following three elements must be specified in the condition:
- fileMatch
- fileNoMatch
- fileSize

If the resource that is being monitored is a queue, one of the following two elements must be specified in the condition:
- queueNotEmpty
- completeGroups

<fileMatch>
Group element for a file name match condition.

<pattern>
Specifies a file name match pattern. Files on the resource must match the pattern in order to satisfy the condition. The default pattern is * (any file will match).

<fileNoMatch>
Group element for an inverse file name match condition.

<pattern>
Specifies an inverse file name match pattern. If no files on the monitored resource match, the condition is satisfied. The default pattern is * (the absence of any file will match).

<fileSize>
Group element for a file size comparison.

<compare>
Specifies a file size comparison. The value must be a non-negative integer.

Attribute	Description
operator	Comparison operator to use. Only >=' is supported.
units	Specifies file size units, which can be one of: • B - bytes • KB - kilobytes • MB - megabytes • GB - gigabytes The units value is case insensitive, so mb' works as well as MB'.

<pattern>
File name pattern to match. Default is * (any file will match).

<queueNotEmpty>
This can only be specified if the resource is a queue. Specifies that there must be a message on the queue for the monitor to be triggered.

<completeGroups>
This can only be specified if the resource is a queue. Specifies that there must be a complete group of messages present on the queue for the monitor to be triggered. A single transfer task is executed for each complete group on the queue.

<name>
Name of the condition.

<resource>
Identifies the resource definition to compare the condition against.

Attribute	Description
id	Unique identifier for the resource.

<tasks>
Group element to contain elements which specify the tasks to invoke when the monitor trigger conditions are satisfied.

<task>
Group element which defines an individual task that the monitor will invoke when the trigger conditions are satisfied. Currently only one task can be specified.

<name>
Name of the task. Accepts any alphanumeric characters.

<description>
Description of the task. Any text value is allowed.

`<taskXML>`
> The XML message that describes the task that the monitor is to perform. The contents of this element are in an escaped XML format.

`<pollInterval>`
> The time interval between each check of the resource against the trigger condition.

Attribute	Description
units	Specifies the time units for the poll interval. Valid values are: • seconds • minutes • hours • days • weeks • months • years

`<batch>`
> The maximum number of trigger matches to include in a single batch.

Attribute	Description
maxSize	The maximum number of trigger matches to include in a single batch

The following XML shows an example of a retained publication which is published to the topic string SYSTEM.FTE/monitors/*agent_name*/MONITORTWO when the monitor called MONITORTWO is created on AGENT_JUPITER. The escaped XML within the <taskXML> element describes the task that is submitted when the monitor condition is met.

```
<?xml version="1.0" encoding="UTF-8"?>
<lst:monitorList xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                 xmlns:lst="http://www.ibm.com/xmlns/wmqfte/7.0.1/MonitorDefinition"
                 xsi:schemaLocation="http://www.ibm.com/xmlns/wmqfte/7.0.1/MonitorDefinition MonitorList.xsd"
                 version="4.00"
                 agent="AGENT_JUPITER"
                 monitor="MONITORTWO">
   <status state="started"/>
   <configuration>
       <description/>
       <resources>
           <directory recursionLevel="0" id="">/srv/nfs/incoming</directory>
       </resources>
       <triggerMatch>
           <conditions>
               <condition>
                   <name/>
                   <resource id=""/>
                   <fileMatch>
                       <pattern>*.completed</pattern>
                   </fileMatch>
               </condition>
           </conditions>
       </triggerMatch>
       <tasks>
           <task>
               <name/>
               <description/>
               <taskXML>&lt;?xml version="1.0" encoding="UTF-8"?&gt;&lt;request
                       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.00"
                       xsi:noNamespaceSchemaLocation="FileTransfer.xsd"&gt;&lt;managedTransfer&gt;
                       &lt;originator&gt;&lt;hostName&gt;example.com.&lt;/hostName&gt;
                       &lt;userID&gt;mqm&lt;/userID&gt;&lt;/originator&gt;
                       &lt;sourceAgent QMgr="QM_JUPITER" agent="AGENT_JUPITER"/&gt;
                       &lt;destinationAgent QMgr="QM_JUPITER" agent="AGENT_SATURN"/&gt;
                       &lt;transferSet&gt;&lt;item checksumMethod="MD5" mode="binary"&gt;
                       &lt;source disposition="leave" recursive="false"&gt;&lt;file
                       &gt;/srv/nfs/incoming/*.txt&lt;/file&gt;&lt;/source&gt;
                       &lt;destination exist="error" type="directory"&gt;
                       &lt;file&gt;/srv/backup&lt;/file&gt;&lt;/destination&gt;
                       &lt;/item&gt;&lt;/transferSet&gt;&lt;/managedTransfer&gt;
                       &lt;/request&gt;
               </taskXML>
           </task>
       </tasks>
```

```
    </configuration>
    <pollInterval units="minutes">1</pollInterval>
    <batch maxSize="1"/>
</lst:monitorList>
```

Schedule list message format

The XML message that is published to a retained publication to the topic string SYSTEM.FTE/Scheduler/
agent_name conforms to the ScheduleList.xsd schema. This XML message lists all active schedules
belonging to that agent. This information is used by the **fteListScheduledTransfers** command and the
WebSphere MQ Explorer plug-in to display a list of schedules to the user. The ScheduleList.xsd schema
document is located in the *MQ_INSTALLATION_PATH*/mqft/samples/schema directory. The ScheduleList.xsd
schema imports FileTransfer.xsd, which is in the same directory.

Schema

The following schema describes which elements are valid in a monitor list XML message.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

    <xsd:include schemaLocation="FileTransfer.xsd"/>

    <xsd:element name="schedules">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="managedTransfer" type="scheduledManagedTransferType" minOccurs="0" maxOccurs="unbounded"/>
            </xsd:sequence>
            <xsd:attribute name="version" type="versionType"          use="required"/>
            <xsd:attribute name="size"    type="xsd:nonNegativeInteger" use="required"/>
            <xsd:attribute name="agent"   type="xsd:string"            use="required"/>
        </xsd:complexType>
    </xsd:element>

    <xsd:complexType name="scheduledManagedTransferType">
        <xsd:sequence>
            <xsd:element name="originator"       type="origRequestType"  maxOccurs="1" minOccurs="1"/>
            <xsd:element name="schedule"         type="scheduleListType" maxOccurs="1" minOccurs="0"/>
            <xsd:element name="sourceAgent"      type="agentType"        maxOccurs="1" minOccurs="1"/>
            <xsd:element name="destinationAgent" type="agentClientType"  maxOccurs="1" minOccurs="1"/>
            <xsd:element name="trigger"          type="triggerType"      maxOccurs="1" minOccurs="0"/>
            <xsd:element name="reply"            type="replyType"        maxOccurs="1" minOccurs="0"/>
            <xsd:element name="transferSet"      type="transferSetType"  maxOccurs="1" minOccurs="1"/>
            <xsd:element name="job"              type="jobType"          maxOccurs="1" minOccurs="0" />
        </xsd:sequence>
        <xsd:attribute name="id" type="idType" use="required"/>
    </xsd:complexType>

    <xsd:complexType name="scheduleListType">
        <xsd:sequence>
            <xsd:element name="submit" type="submitType"     maxOccurs="1" minOccurs="1"/>
            <xsd:element name="repeat" type="repeatType"     maxOccurs="1" minOccurs="0"/>
            <xsd:element name="next"   type="noZoneTimeType" maxOccurs="1" minOccurs="1"/>
        </xsd:sequence>
    </xsd:complexType>

</xsd:schema>
```

Understanding the schedule list message

The elements and attributes used in the schedule list messages are described in the following list:

<schedules>
    Group element containing information about all of the schedules defined on a single agent.

Attribute	Description
agent	Required. The name of the source agent that the schedule is defined on.
size	Required. The number of schedules defined on this agent.
version	Required. The version of the schedule list message format.

`<managedTransfer>`

Group element containing information about a single schedule.

Attribute	Description
id	Required. The hexadecimal string ID of the schedule request message.

`<originator>`

The originator of the schedule request.

`<hostName>`

The host name of the machine that the schedule request was submitted from.

`<userID>`

The user ID of the user that submitted the schedule request.

`<mqmdUserID>`

The MQMD user ID of the user that submitted the schedule request.

`<webBrowser>`

If the schedule request was submitted through the Web Gateway, the web browser that the request was submitted from.

`<webUserID>`

If the schedule request was submitted through the Web Gateway, the web user ID of the user that submitted the schedule request.

`<schedule>`

Element that contains the elements that describe when the scheduled transfer occurs.

`<submit>`

Specifies the date and time that the scheduled transfer is due to start.

Attribute	Description
timebase	Specifies which time zone to use. The value of this attribute can be one of the following values: • source - use the time zone of the source agent • admin - use the time zone of the administrator issuing the command • UTC - use Coordinated Universal Time
timezone	The time zone description according to the timebase value

`<repeat>`

Group element that contains details about how often a scheduled transfer repeats, how many times a scheduled transfer repeats, and when a scheduled transfer stops repeating.

Attribute	Description
interval	The interval units, which must be one of the following values: • minutes • hours • days • weeks • months • years

`<frequency>`
> The time period that must elapse before the transfer repeats.

Attribute	Description
interval	The interval units, which must be one of the following values: • minutes • hours • days • weeks • months • years

`<expireTime>`
> Optional element that specifies the date and time that a repeating scheduled transfer stops. This element and the <expireCount> element are mutually exclusive.

`<expireCount>`
> Optional element that specifies the number of times the scheduled file transfer occurs before stopping. This element and the <expireTime> element are mutually exclusive.

`<next>`
> Specifies the date and time when the next scheduled transfer is due to start.

`<sourceAgent>`
> Specifies the name of the agent on the system where the source file is located.

Attribute	Description
agent	Specifies the name of the agent.
QMgr	The name of the agent queue manager.

`<destinationAgent>`
> Specifies the name of the agent on the system you want to transfer the file to.

Attribute	Description
agent	Specifies the name of the agent.
QMgr	The name of the agent queue manager.

`<trigger>`
> Optional element that specifies a condition that must be true for the file transfer to take place.

Attribute	Description
log	A flag indicating whether trigger failures are logged. The following are valid values: • yes - log entries are created for failed triggered transfers • no - log entries are not created for failed triggered transfers

`<reply>`

Specifies the name of the temporary reply queue generated for synchronous file transfers (specified with the **-w** parameter on the command line). The name of the queue is defined by the key **dynamicQueuePrefix** in the command.properties configuration file or the default of WMQFTE.* if not specified.

Attribute	Description
QMGR	The name of the command queue manager on which the temporary dynamic queue is generated to receive replies.

`<transferSet>`

Specifies a group of file transfers you want the scheduled transfer to perform together. During transmission <transferSet> is a group element containing <item> elements.

Attribute	Description
priority	Priority level of the transfer. Priority is a value in the range 0-9, where 0 is the lowest priority. The default priority level is 0 and by default the transfer uses the priority level of the source agent.

`<job>`

Optional group element containing job information for the entire transfer specification. <job> is a user-defined job name identifier that is added to the log message when the transfer has started. This <job> element is the same as the <job> element that appears in the transfer log message, which is described in the following topic: "File transfer log message formats" on page 626.

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<schedules xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         size="2"
         version="4.00"
         agent="AGENT_JUPITER"
         xsi:noNamespaceSchemaLocation="ScheduleList.xsd">
    <managedTransfer id="1">
        <originator>
            <hostName>example.com.</hostName>
            <userID>mqm</userID>
        </originator>
        <schedule>
            <submit timebase="admin" timezone="Europe/London">2010-01-01T21:00+0000</submit>
            <next>2010-01-01T21:00+0000</next>
        </schedule>
        <sourceAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER"/>
        <destinationAgent agent="AGENT_SATURN" QMgr="QM_JUPITER"/>
        <reply QMGR="QM_JUPITER">WMQFTE.4D400F8B20004E06</reply>
        <transferSet>
            <item mode="binary" checksumMethod="MD5">
                <source recursive="false" disposition="leave">
                    <file>/etc/passwd</file>
                </source>
```

```
            <destination type="directory" exist="overwrite">
                <file>/tmp</file>
            </destination>
        </item>
    </transferSet>
  </managedTransfer>
  <managedTransfer id="2">
      <originator>
          <hostName>example.com.</hostName>
          <userID>mqm</userID>
      </originator>
      <schedule>
          <submit timebase="admin" timezone="Europe/London">2010-12-31T09:00+0000</submit>
          <next>2010-12-31T09:00+0000</next>
      </schedule>
      <sourceAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER"/>
      <destinationAgent agent="AGENT_NEPTUNE" QMgr="QM_JUPITER"/>
      <reply QMGR="QM_JUPITER">WMQFTE.4D400F8B20004E09</reply>
      <transferSet>
          <item mode="binary" checksumMethod="MD5">
              <source recursive="false" disposition="leave">
                  <file>/etc/passwd</file>
              </source>
              <destination type="directory" exist="overwrite">
                  <file>/tmp</file>
              </destination>
          </item>
      </transferSet>
  </managedTransfer>
</schedules
```

Example template XML message

When a template is created, a message is published to the SYSTEM.FTE topic with a topic string of
Templates/*template_ID*. This example XML describes a single template defined in your WebSphere MQ
Managed File Transfer network.

```
<?xml version="1.0" encoding="UTF-8"?>
<transferTemplate version="4.00" id="baf9df73-45c2-4bb0-a085-292232ab66bc">
    <name>BASIC_TEMPLATE</name>
    <sourceAgentName>AGENT_JUPITER</sourceAgentName>
    <sourceAgentQMgr>QM_JUPITER</sourceAgentQMgr>
    <destinationAgentName>AGENT_SATURN</destinationAgentName>
    <destinationAgentQMgr>QM_JUPITER</destinationAgentQMgr>
    <fileSpecs>
        <item mode="binary" checksumMethod="MD5">
            <source recursive="false" disposition="leave">
                <file>/etc/passwd</file>
            </source>
            <destination type="directory" exist="overwrite">
                <file>/tmp</file>
            </destination>
        </item>
    </fileSpecs>
    <priority>0</priority>
</transferTemplate>
```

Related tasks:

"Creating a file transfer template using the WebSphere MQ Explorer" on page 233
You can create a file transfer template from the WebSphere MQ Explorer or from the command line. You can then use that template to create new file transfers using the template details or submit the template to start the file transfer.

Related reference:

"**fteCreateTemplate** (create new file transfer template)" on page 449
The **fteCreateTemplate** command creates a file transfer template that you can keep for future use. The only required parameter is the **-tn** (*template_name*) parameter. All other parameters are optional, although if you specify a source file specification, you must also provide a destination file. Similarly, if you specify a destination file, you must also specify a source file specification.

File transfer status message format

Messages are published to the coordination queue manager to indicate transfer status of each file in the transfer set. Every time a request for file transfer is processed by the agent, a transaction message is published to the coordination queue manager (on its SYSTEM.FTE/Transfers/*agent_name/transfer ID* topic), which conforms to the TransferStatus.xsd XML schema. The TransferStatus.xsd file is located in the *MQ_INSTALLATION_PATH*/mqft/samples/schema directory of your WMQMFT installation.

Schema

The following schema describes which elements are valid in a transfer status XML message.

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

    <xsd:include schemaLocation="fteutils.xsd"/>

    <xsd:element name="transaction">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="sourceAgent"      type="agentType"
                            maxOccurs="1"            minOccurs="1"/>
                <xsd:element name="destinationAgent" type="agentType"
                            maxOccurs="1"            minOccurs="1"/>
                <xsd:element name="transferSet"      type="transferSetType"
                            maxOccurs="1"            minOccurs="1"/>
            </xsd:sequence>
            <xsd:attribute name="version" type="versionType"   use="required"/>
            <xsd:attribute name="ID"      type="IDType"        use="required"/>
        </xsd:complexType>
    </xsd:element>

    <xsd:complexType name="transferSetType">
        <xsd:sequence>
            <xsd:element name="stats"   type="statsType"
                        maxOccurs="1" minOccurs="1" />
            <xsd:element name="current" type="currentType"
                        maxOccurs="1" minOccurs="0" />
        </xsd:sequence>
        <xsd:attribute name="time" type="xsd:dateTime" use="required" />
    </xsd:complexType>

    <xsd:complexType name="currentType">
        <xsd:sequence>
            <xsd:element name="source"      type="fileSourceType"
                        maxOccurs="1"       minOccurs="1" />
            <xsd:element name="destination" type="fileDestinationType"
                        maxOccurs="1"       minOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="transferred" type="xsd:nonNegativeInteger" use="required" />
        <xsd:attribute name="size" type="xsd:nonNegativeInteger"  use="required" />
    </xsd:complexType>
```

```
<xsd:complexType name="statsType">
    <xsd:attribute name="bytes"        type="xsd:nonNegativeInteger"  use="required" />
    <xsd:attribute name="seconds"      type="xsd:decimal"             use="required" />
    <xsd:attribute name="currentItem"  type="xsd:nonNegativeInteger"  use="required" />
    <xsd:attribute name="totalItems"   type="xsd:nonNegativeInteger"  use="required" />
</xsd:complexType>

</xsd:schema>
```

Understanding the transfer status message

The elements and attributes used in the transfer status messages are described in the following list:

`<transaction>`
> Group element that contains all of the elements for the file transfers.

Attribute	Description
version	Specifies the version of this element as supplied by WebSphere MQ Managed File Transfer.
ID	The unique identifier for the file transfer.

`<sourceAgent>`
> Specifies the name of the agent on the system where the source file is located.

Attribute	Description
agent	The name of the agent.
QMgr	The name of the agent queue manager.

`<destinationAgent>`
> Specifies the name of the agent on the system you want to transfer the file to.

Attribute	Description
agent	The name of the agent.
QMgr	The name of the agent queue manager.

`<transferset>`
> Specifies a group of file transfers being performed together. All of the files in the transfer must originate at the same source agent and end at the same destination agent.

Attribute	Description
time	Specifies the date and time (in date time format).

`<stats>`
> Required. Defines metrics about the transfer, including the number of bytes copied so far, in the given number of seconds. Also supplies the current item number out of the total number of items in the <transferSet>.

Attribute	Description
bytes	Number of bytes copied so far.
seconds	Number of seconds taken to transfer those bytes.
currentItem	The index of the current item being transferred.
totalItems	The total number of items being transferred.

<current>

Optional element. Group element that contains elements that specify the file transfer currently in progress. The <current> element indicates how many bytes of data have been transferred so far for the current item and the expected total number of bytes

<source>

Group element that contains the element specifying the source file name.

<file>

Specifies the source path of the file that is being transferred. The path is as specified for the transfer. This path might differ from the path that is output as part of the transfer log, which is the absolute form the of path.

<destination>

Group element that contains the element specifying the destination file name or specification.

<file>

Specifies the destination path of the file that is being transferred. The path is as specified for the transfer. This path might differ from the path that is output as part of the transfer log, which is the absolute form the of path.

Attribute	Description
alias	Specifies an alias for the destination file. This alias is the name of the source file, excluding any directory path specified for the transfer.
filespace	Specifies the name of the file space where the destination file is written.

<queue>

When used with the <destination> element, specifies the name of the queue you want to transfer to. This name is in the format QUEUE or QUEUE@QUEUE_MANAGER.

Transfer progress message examples:

When a transfer is in progress, messages are published to the SYSTEM.FTE topic with a topic string of Transfers/*agent_name/transfer_ID*. The XML examples show the progress message for a single file transfer and for a multiple file transfer.

Single file transfer

The following example shows the details of a single file transfer that is in progress.

```
<?xml version="1.0" encoding="UTF-8"?>
<transaction xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             version="4.00"
             ID="414d51205553322e42494e44494e47538b0f404d223d0020"
             xsi:noNamespaceSchemaLocation="TransferStatus.xsd">
    <sourceAgent agent="US2.BINDINGS.FILE" QMgr="US2.BINDINGS"/>
    <destinationAgent agent="US2.BINDINGS.FILE" QMgr="US2.BINDINGS"/>
    <transferSet time="2011-01-26T13:03:26.542Z">
    <stats bytes="1198" seconds="0.018" currentItem="1" totalItems="1"/>
    <current transferred="1151" size="1151">
        <source>
            <file>/etc/passwd</file>
        </source>
        <destination>
            <file>/tmp/passwd</file>
        </destination>
    </current>
    </transferSet>
</transaction>
```

Multiple file transfer

If there were more files in the transfer set, the transfer status message indicates which one is being processed and how many bytes have been transferred so far.

```
<?xml version="1.0" encoding="UTF-8"?>
<transaction xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             version="4.00"
             ID="414d51205553322e42494e44494e47538b0f404d035c0020"
             xsi:noNamespaceSchemaLocation="TransferStatus.xsd">
    <sourceAgent agent="US2.BINDINGS.FILE" QMgr="US2.BINDINGS"/>
    <destinationAgent agent="US2.BINDINGS.FILE" QMgr="US2.BINDINGS"/>
    <transferSet time="2011-01-26T13:12:58.636Z">
        <stats bytes="440" seconds="0.082" currentItem="10" totalItems="10"/>
        <current transferred="0" size="0">
            <source>
                <file>/srv/nfs/incoming/file10.txt</file>
            </source>
            <destination>
                <file>/srv/nfs/outgoing/file10.txt</file>
            </destination>
        </current>
    </transferSet>
</transaction>
```

File transfer log message formats

File transfer log messages are published to the SYSTEM.FTE topic with a topic string of Log/*agent_name*/*transfer_ID*. These messages conform to the schema TransferLog.xsd, which is located in the *MQ_INSTALLATION_PATH*/mqft/samples/schema directory of your WebSphere MQ Managed File Transfer installation.

If you want to monitor file transfers or collect data about them, set up a subscription to a wildcard topic tailored to the transfers you are interested in. For example:

```
Log/#
```

or,

```
Log/FTEAGENT/#
```

This subscription can be durable or non-durable. Durable subscriptions continue to exist when a subscribing application's connection to the queue manager is closed. Non-durable subscriptions exist only as long as a subscribing application's connection to the queue manager remains open.

Schema

The following schema describes which elements are valid in a transfer log XML message.

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

    <xsd:include schemaLocation="fteutils.xsd"/>

    <xsd:element name="transaction">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="action"               type="actionType"
                            maxOccurs="1"                minOccurs="0"/>
                <xsd:element name="sourceAgent"          type="agentExitStatusType"
                            maxOccurs="1"                minOccurs="0"/>
                <xsd:element name="sourceWebGateway"     type="webGatewayType"
                            maxOccurs="1"                minOccurs="0"/>
                <xsd:element name="sourceWebUser"        type="webUserType"
                            maxOccurs="1"                minOccurs="0"/>
                <xsd:element name="destinationAgent"     type="agentExitStatusType"
                            maxOccurs="1"                minOccurs="0"/>
                <xsd:element name="destinationWebGateway" type="webGatewayType"
                            maxOccurs="1"                minOccurs="0"/>
                <xsd:element name="destinationWebUser"   type="webUserType"
```

```
                                      maxOccurs="1"              minOccurs="0"/>
            <xsd:element name="agent"                  type="agentExitStatusType"
                                      maxOccurs="1"              minOccurs="0"/>
            <xsd:element name="originator"             type="origRequestType"
                                      maxOccurs="1"              minOccurs="1"/>
            <xsd:element name="status"                 type="statusType"
                                      maxOccurs="1"              minOccurs="0"/>
            <xsd:element name="trigger"                type="triggerType"
                                      maxOccurs="1"              minOccurs="0" />
            <xsd:element name="transferSet"            type="transferSetType"
                                      maxOccurs="1"              minOccurs="1"/>
            <xsd:element name="job"                    type="jobType"
                                      maxOccurs="1"              minOccurs="0"/>
            <xsd:element name="scheduleLog"            type="scheduleLogType"
                                      maxOccurs="1"              minOccurs="0"/>
            <xsd:element name="statistics"             type="statisticsType"
                                      maxOccurs="1"              minOccurs="0"/>
        </xsd:sequence>
        <xsd:attribute name="version"   type="versionType"    use="required"/>
        <xsd:attribute name="ID"        type="IDType"         use="required"/>
        <xsd:attribute name="relatedID" type="IDType"         use="optional"/>
        <xsd:attribute name="agentRole" type="agentRoleType" use="optional"/>
    </xsd:complexType>
</xsd:element>

<xsd:complexType name="agentExitStatusType">
    <xsd:complexContent>
        <xsd:extension base="agentType">
            <xsd:sequence>
                <xsd:element name="startExits" type="exitGroupType" minOccurs="0" maxOccurs="1"/>
                <xsd:element name="endExits"   type="exitGroupType" minOccurs="0" maxOccurs="1"/>
                <xsd:element name="systemInfo" type="systemInfoType" minOccurs="0" maxOccurs="1"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="transferSetType">
    <xsd:sequence>
        <xsd:element name="metaDataSet"          type="metaDataSetType"
                     maxOccurs="1"              minOccurs="0" />
        <xsd:element name="call"                 type="callGroupType"
                     maxOccurs="1"              minOccurs="0"/>
        <xsd:element name="preSourceCall"        type="callGroupType"
                     maxOccurs="1"              minOccurs="0"/>
        <xsd:element name="postSourceCall"       type="callGroupType"
                     maxOccurs="1"              minOccurs="0"/>
        <xsd:element name="preDestinationCall"   type="callGroupType"
                     maxOccurs="1"              minOccurs="0"/>
        <xsd:element name="postDestinationCall"  type="callGroupType"
                     maxOccurs="1"              minOccurs="0"/>
        <xsd:element name="item"                 type="itemType"
                     maxOccurs="unbounded"      minOccurs="0"/>
    </xsd:sequence>
    <xsd:attribute name="index"     type="xsd:nonNegativeInteger" use="optional" />
    <xsd:attribute name="size"      type="xsd:nonNegativeInteger" use="optional" />
    <xsd:attribute name="startTime" type="xsd:dateTime"           use="required" />
    <xsd:attribute name="total"     type="xsd:nonNegativeInteger" use="required" />
    <xsd:attribute name="bytesSent" type="xsd:nonNegativeInteger" use="required" />
</xsd:complexType>

<xsd:complexType name="itemType">
    <xsd:sequence>
        <xsd:element name="source"      type="fileSourceChecksumType"
                     maxOccurs="1"      minOccurs="1" />
        <xsd:element name="destination" type="fileDestinationChecksumType"
                     maxOccurs="1"      minOccurs="1" />
        <xsd:element name="status"      type="statusType"
                     maxOccurs="1"      minOccurs="1" />
    </xsd:sequence>
    <xsd:attribute name="mode" type="modeType" use="required" />
</xsd:complexType>

<xsd:complexType name="fileSourceChecksumType">
    <xsd:complexContent>
        <xsd:extension base="fileSourceType">
            <xsd:sequence>
                <xsd:element name="checksum" type="checksumType" minOccurs="0" maxOccurs="1"/>
            </xsd:sequence>
```

```
            </xsd:extension>
        </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="fileDestinationChecksumType">
    <xsd:complexContent>
        <xsd:extension base="fileDestinationType">
            <xsd:sequence>
                <xsd:element name="checksum" type="checksumType"
                             minOccurs="0"   maxOccurs="1"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="actionType">
    <xsd:simpleContent>
        <xsd:extension base="actionEnumType">
            <xsd:attribute name="time" type="xsd:dateTime" use="required" />
        </xsd:extension>
    </xsd:simpleContent>
</xsd:complexType>

<xsd:simpleType name="actionEnumType">
    <xsd:restriction base="xsd:token">
        <xsd:enumeration value="cancelled"/>
        <xsd:enumeration value="started"/>
        <xsd:enumeration value="progress"/>
        <xsd:enumeration value="completed"/>
        <xsd:enumeration value="malformed"/>
        <xsd:enumeration value="notAuthorized"/>
        <xsd:enumeration value="deleted"/>
    </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="systemInfoType">
    <xsd:attribute name="architecture" type="xsd:string" use="required"/>
    <xsd:attribute name="name"         type="xsd:string" use="required"/>
    <xsd:attribute name="version"      type="xsd:string" use="required"/>
</xsd:complexType>

<xsd:element name="malformed">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="action"  type="actionType"
                         maxOccurs="1"  minOccurs="1"/>
            <xsd:element name="agent"   type="agentExitStatusType"
                         maxOccurs="1"  minOccurs="0"/>
            <xsd:element name="status"  type="statusType"
                         maxOccurs="1"  minOccurs="1"/>
        </xsd:sequence>
        <xsd:attribute name="version"   type="versionType"    use="required"/>
        <xsd:attribute name="ID"        type="IDType"         use="required"/>
        <xsd:attribute name="agentRole" type="agentRoleType"  use="required"/>
    </xsd:complexType>
</xsd:element>

<xsd:element name="notAuthorized">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="action"     type="actionType"
                         maxOccurs="1"     minOccurs="1"/>
            <xsd:element name="originator" type="origRequestType"
                         maxOccurs="1"     minOccurs="1"/>
            <xsd:element name="authority"  type="xsd:string"
                         minOccurs="1"     maxOccurs="1"/>
            <xsd:element name="status"     type="statusType"
                         maxOccurs="1"     minOccurs="1"/>
        </xsd:sequence>
        <xsd:attribute name="version"   type="versionType"    use="required"/>
        <xsd:attribute name="ID"        type="IDType"         use="required"/>
        <xsd:attribute name="agentRole" type="agentRoleType"  use="required"/>
    </xsd:complexType>
</xsd:element>

<xsd:complexType name="statisticsType">
    <xsd:sequence>
        <xsd:element name="actualStartTime" type="xsd:dateTime"
                     maxOccurs="1"           minOccurs="0"/>
```

```
            <xsd:element name="retryCount"       type="xsd:nonNegativeInteger"
                         maxOccurs="1"            minOccurs="1"/>
            <xsd:element name="numFileFailures" type="xsd:nonNegativeInteger"
                         maxOccurs="1"            minOccurs="1"/>
            <xsd:element name="numFileWarnings" type="xsd:nonNegativeInteger"
                         maxOccurs="1"            minOccurs="1"/>
        </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="webGatewayType">
        <xsd:attribute name="webGatewayName"      type="xsd:string" use="optional" />
        <xsd:attribute name="webGatewayAgentName" type="xsd:string" use="optional" />
        <xsd:attribute name="webGatewayAgentQMgr" type="xsd:string" use="optional" />
    </xsd:complexType>

    <xsd:complexType name="webUserType">
        <xsd:attribute name="webGatewayName"      type="xsd:string" use="required" />
        <xsd:attribute name="webGatewayAgentName" type="xsd:string" use="optional" />
        <xsd:attribute name="webGatewayAgentQMgr" type="xsd:string" use="optional" />
    </xsd:complexType>

</xsd:schema>
```

Understanding the transfer log message

`<transaction>`

Group element that specifies a group of transfers you want to perform together.

Attribute	Description
version	Specifies the version of this element as detailed by WebSphere MQ Managed File Transfer.
ID	Specifies the unique transaction ID. The ID can be a maximum of 48 alphanumeric characters.
relatedID	Optional. If the transaction is the delete or download of a file from a file space, **relatedID** specifies the transaction ID of the transfer that uploaded the file to the file space.
agentRole	Optional. Specifies whether the agent concerned is on the source or destination system
xmlns:xsi	Namespace declaration. Indicates that the elements and data types used in this schema derive from the "http://www.w3.org/2001/XMLSchema-instance" namespace.
xsi:noNamespaceSchemaLocation	Specifies the name and location of the XML schema document to validate this message against if there is no namespace declaration. The value you specify for this attribute must refer to a WebSphere MQ Managed File Transfer TransferLog.xsd document.

`<action>`

Describes the status of the file transfer at the time logged by the time attribute. The status can be one of the following values:

- started
- progress
- completed
- cancelled
- malformed (indicates the file transfer request message content can not be interpreted.)
- notAuthorized
- deleted

Attribute	Description
time	The time that the transfer status was captured, expressed in UTC format.

\<sourceAgent\>

Specifies the name of the agent on the system where the source file is located. Only one of \<sourceAgent\>, \<sourceWebUser\>, and \<sourceWebGateway\> can be specified.

\<startExits\>

Group element that contains one or more user exit elements. This element can occur once only.

\<endExits\>

Group element that contains one or more user exit elements. This element can occur once only.

\<systemInfo\>

Describes the system architecture, name, and version. This element can occur once only.

Attribute	Description
agent	The name of the agent on the source system.
QMgr	The name of the queue manager on the source system.
agentType	The type of the agent. Valid values are: • STANDARD - a normal agent • BRIDGE - a protocol bridge agent • CD_BRIDGE - a Connect:Direct bridge agent • EMBEDDED - an embedded agent • WEB_GATEWAY - a web agent • SFG - a Sterling File Gateway embedded agent
bridgeURL	Optional. If the agent is a protocol bridge agent, the host name of the system hosting the protocol server.
pnode	Optional. If the agent is a Connect:Direct bridge agent, the name of the Connect:Direct primary node involved in the transfer.
snode	Optional. If the agent is a Connect:Direct bridge agent, the name of the Connect:Direct secondary node involved in the transfer.
bridgeNode	Optional. If the agent is a Connect:Direct bridge agent, the name of the Connect:Direct node that is part of the Connect:Direct bridge. This is the same node as either the primary node or the secondary node.

\<sourceWebUser\>

Specifies the name of the web user that uploads the source file to the Web Gateway. Only one of \<sourceAgent\>, \<sourceWebUser\>, and \<sourceWebGateway\> can be specified.

Attribute	Description
webGatewayName	The name of the Web Gateway that receives the file from the web user.
webGatewayAgentName	The name of the web agent that the Web Gateway uses to send the file to the destination.
webGatewayAgentQMgr	The name of the queue manager of the web agent.

`<sourceWebGateway>`

Specifies the name of the Web Gateway that the source file is downloaded from. Only one of <sourceAgent>, <sourceWebUser>, and <sourceWebGateway> can be specified.

Attribute	Description
webGatewayName	The name of the Web Gateway that receives the file from the web user.
webGatewayAgentName	The name of the web agent that the Web Gateway uses to send the file to the destination.
webGatewayAgentQMgr	The name of the queue manager of the web agent.

`<destinationAgent>`

Specifies the name of the agent on the system the file was transferred to. Only one of <destinationAgent>, <destinationWebGateway>, and <destinationWebUser> can be specified.

Attribute	Description
agent	The name of the agent on the destination system.
QMgr	The name of the queue manager on the destination system.
agentType	The type of the agent. Valid values are: • STANDARD - a normal agent • BRIDGE - a protocol bridge agent • CD_BRIDGE - a Connect:Direct bridge agent • EMBEDDED - an embedded agent • WEB_GATEWAY - a web agent • SFG - a Sterling File Gateway embedded agent
bridgeURL	Optional. If the agent is a protocol bridge agent, the host name of the system hosting the protocol server.
pnode	Optional. If the agent is a Connect:Direct bridge agent, the name of the Connect:Direct primary node involved in the transfer.
snode	Optional. If the agent is a Connect:Direct bridge agent, the name of the Connect:Direct secondary node involved in the transfer.
bridgeNode	Optional. If the agent is a Connect:Direct bridge agent, the name of the Connect:Direct node that is part of the Connect:Direct bridge. This is the same node as either the primary node or the secondary node.

<startExits>

Group element that contains one or more user exit elements. This element can occur once only.

<endExits>

Group element that contains one or more user exit elements. This element can occur once only.

<systemInfo>

Describes the system architecture, name, and version. This element can occur once only.

`<destinationWebUser>`

Specifies the name of the web user who downloads the file from the Web Gateway. Only one of <destinationAgent>, <destinationWebGateway>, and <destinationWebUser> can be specified.

Attribute	Description
webGatewayName	The name of the Web Gateway that receives the file from the web user.

`<destinationWebGateway>`

Specifies the name of the web user who downloads the file from the Web Gateway. Only one of <destinationAgent>, <destinationWebGateway>, and <destinationWebUser> can be specified.

Attribute	Description
webGatewayName	The name of the Web Gateway that receives the file from the web user.
webGatewayAgentName	The name of the web agent that the Web Gateway uses.
webGatewayAgentQMgr	The name of the queue manager of the web agent.

`<originator>`

Group element that contains the elements specifying the originator of the request.

<hostName>

The host name of the system where the source file is located.

<userID>

The user ID that originated the file transfer.

<mqmdUserID>

The WebSphere MQ user ID that was supplied in the message descriptor (MQMD)

<webUserID>

Optional. The user ID that was supplied to the web browser submitting the transfer request.

<webBrowser>

Optional. The web browser that the transfer request was submitted from.

`<status>`

The result code and supplement messages.

`<trigger>`

Group element that contains the trigger elements defined in the original transfer request. These elements can be either or both of the following:

<fileExist>

Trigger condition based on whether a file exists

<fileSize>

Trigger condition based on whether a file meets or exceeds the specified size

`<transferSet>`

Specifies a group of file transfers you want to perform together. During transmission <transferSet> is a group element containing <item> elements.

Attribute	Description
startTime	Records the time that the set of transfers started, expressed in UTC format.
total	Specifies the total number of items in this set of transfers.
index	Optional attribute. Specifies the position of the first item in progress of the transfer set. The index attribute increments from zero. For example, if the index is set to 1, the progress message is the second of two items.

Attribute	Description
size	Optional attribute. Specifies the number of items in the progress report.
priority	Optional attribute. Priority level of the transfer. Priority is a value in the range 0-9, where 0 is the lowest priority. The default priority level is 0 and by default the transfer uses the source agent priority level.

\<metaDataSet\>

Group element containing one or more of the following attributes:

\<metaData\>

Attribute	Description
key	The key half of a metadata key-value pair. The \<metaData\> element content contains the value half of the pair. For example \<metaData key="testkey1"\>testvalue1\</metaData\>

`<job>`

Group element that contains an element specifying job details. \<job\> is a user-defined job name identifier that is added to the log message when the transfer has started. This \<job\> element is the same as the \<job\> element that is included in the transfer request message, which is described in the following topic: "File transfer request message format" on page 803.

\<name\>

The value of name can be any string.

`<scheduleLog>`

Group element that contains elements specifying the source and destination file names and locations.

Attribute	Description
ID	Matches the schedule ID if the transfer is a scheduled transfer.

`<item>`

Group element that contains elements specifying the source and destination file names and locations.

`<source>`

Group element that contains the \<file\> element or the \<queue\> element, and the \<checksum\> element for the file on the source system.

Attribute	Description
recursive	Specifies that files are transferred recursively in subdirectories when the \<source\> element is a directory or contains wildcard characters.
disposition	Specifies the action that is taken on the \<source\> element when \<source\> has successfully been transferred to its destination. The valid options are as follows: • leave - the source files are left unchanged. • delete - the source files are deleted from the source system after the source file is successfully transferred.
correlationBoolean	A boolean correlation value. If the source is a Connect:Direct bridge, this specifies whether the Connect:Direct process is user-defined.

Attribute	Description
correlationString1	A string correlation value. If the source is a Connect:Direct bridge, this specifies the name of the Connect:Direct process that occurs at the destination of the transfer.
correlationNum1	A numeric correlation value. If the source is a Connect:Direct bridge, this specifies the ID number of the Connect:Direct process that occurs at the destination of the transfer.

`<queue>`

When used with the <source> element, specifies the name of the queue that the transferred messages were read from, which is located on the source agent queue manager.

Attribute	Description
messageCount	The number of messages that were read from the queue.
groupId	The WebSphere MQ group ID of the messages read from the queue.

`<destination>`

Group element that contains the <file> element or the <queue> element, and <checksum> element for the destination.

Only one of <file> and <queue> is present as a child element of destination.

Attribute	Description
type	The type of destination. The valid options are as follows: • file - specifies a file as the destination • directory - specifies a directory as the destination • queue- specifies a WebSphere MQ queue as the destination The options file and directory can be present only when the **<destination>** element has a child element of **<file>**. The option queue can be present only when the **<destination>** element has a child element of **<queue>**.
exist	Specifies the action that is taken if a destination file exists on the destination system. The valid options are as follows: • error - reports an error and the file is not transferred. • overwrite - overwrites the existing destination file. This attribute cannot be present if the **<destination>** element has a child element of **<queue>**.
correlationBoolean	A boolean correlation value. If the destination is a Connect:Direct bridge, this specifies whether the Connect:Direct process is user-defined.
correlationString1	A string correlation value. If the destination is a Connect:Direct bridge, this specifies the name of the Connect:Direct process that occurs at the destination of the transfer.

Attribute	Description
correlationNum1	A numeric correlation value. If the destination is a Connect:Direct bridge, this specifies the ID number of the Connect:Direct process that occurs at the destination of the transfer.

`<file>`

> Specifies the absolute path of the file that was transferred (both at the source and destination). The fully-qualified path is in the format consistent with your operating system, for example `C:/from/here.txt`. File URIs are not used.

`<queue>`

> When used with the <destination> element, specifies the name of the queue that was transferred to, which is located on any queue manager that is connected to the destination agent queue manager.

Attribute	Description
messageCount	The number of messages that were written to the queue.
messageLength	The length of the messages written to the queue.
groupId	If the transfer request specified that the file is split into multiple messages, the value of this attribute is the WebSphere MQ group ID of the messages written to the queue.
messageId	If the transfer request did not specify that the file is split into multiple messages, the value of this attribute is the WebSphere MQ message ID of the message written to the queue.

`<checksum>`

> Optional element.

> Specifies the type of hash algorithm that generated the message digest to create the digital signature. Currently WebSphere MQ Managed File Transfer supports Message Digest algorithm 5 (MD5) only. The checksum provides a way for you to confirm the integrity of transferred files is intact.

`<malformed>`

> Group element for malformed messages.

Attribute	Description
version	
ID	
agentRole	Either source agent or destination agent

`<statistics>`

> Group element for statistical information for the transfer (when available).

> **<actualStartTime>**

>> The actual time that the agent started running the transfer. Typically, the time is the same as (or very close to) the start time recorded for the transfer. However, when an agent is busy submitted transfers might be queued until the agent has capacity to run the transfers.

> **<retryCount>**

>> The number of times that the transfer went into the recovery state and was retried by the agent. A transfer can go into a recovery state because the source and destination agents lose communication, either because of a WebSphere MQ network error or because they are not receiving data or acknowledgment messages for a period. This period is determined by the agent properties: transferAckTimeout and transferAckTimeoutRetries.

<numFileFailures>
> The number of files in the transferSet that failed to transfer successfully.

<numFileWarnings>
> The number of files in the transferSet that generated warnings while being transferred, but otherwise transferred successfully.

Examples

Examples of XML messages that conform to this schema are provided for each of the following types of transfer:

- A transfer of a single file
- A transfer that contains multiple files
- A failed file transfer
- A transfer defined with a trigger
- A transfer started by a schedule
- A transfer that calls user exits
- A transfer requested through the Web Gateway
- A transfer through a Connect:Direct bridge node

Single transfer log message examples:

When a transfer occurs, messages are published to the SYSTEM.FTE topic with a topic string of Log/*agent_name/transfer_ID*. The XML examples show the log messages for a single file transfer being started, in progress, and completed.

Single file transfer - started

```
<?xml version="1.0" encoding="UTF-8"?>
<transaction xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             version="4.00"
             ID="414d51205553322e42494e44494e47538b0f404d223d0020"
             agentRole="sourceAgent"
             xsi:noNamespaceSchemaLocation="TransferLog.xsd"
             xmlns="">
    <action time="2011-01-26T13:03:26.484Z">started</action>
    <sourceAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER">
        <systemInfo architecture="x86" name="Linux" version="2.6.31-21-generic"/>
    </sourceAgent>
    <destinationAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER"/>
    <originator>
        <hostName>dhcp-9-20-240-199.hursley.ibm.com.</hostName>
        <userID>mqm</userID>
        <mqmdUserID>mqm</mqmdUserID>
    </originator>
    <transferSet startTime="2011-01-26T13:03:26.484Z" total="1" bytesSent="0">
        <metaDataSet>
            <metaData key="com.ibm.wmqfte.SourceAgent">AGENT_JUPITER</metaData>
            <metaData key="com.ibm.wmqfte.DestinationAgent">AGENT_JUPITER</metaData>
            <metaData key="com.ibm.wmqfte.MqmdUser">mqm</metaData>
            <metaData key="com.ibm.wmqfte.OriginatingUser">mqm</metaData>
            <metaData key="com.ibm.wmqfte.OriginatingHost">dhcp-9-20-240-199.hursley.ibm.com.</metaData>
            <metaData key="com.ibm.wmqfte.TransferId">414d51205553322e42494e44494e47538b0f404d223d0020</metaData>
            <metaData key="com.ibm.wmqfte.ScheduleId">3</metaData>
            <metaData key="com.ibm.wmqfte.Priority">0</metaData>
        </metaDataSet>
    </transferSet>
    <scheduleLog ID="3"/>
</transaction>
```

Single file transfer success - progress

```
<?xml version="1.0" encoding="UTF-8"?>
<transaction xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             version="4.00"
             ID="414d51205553322e42494e44494e47538b0f404d223d0020"
```

```
                agentRole="sourceAgent"
                xsi:noNamespaceSchemaLocation="TransferLog.xsd"
                xmlns="">
        <action time="2011-01-26T13:03:26.615Z">progress</action>
        <sourceAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER">
            <systemInfo architecture="x86" name="Linux" version="2.6.31-21-generic"/>
        </sourceAgent>
        <destinationAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER">
            <systemInfo architecture="x86" name="Linux" version="2.6.31-21-generic"/>
        </destinationAgent>
        <originator>
            <hostName>example.com.</hostName>
            <userID>mqm</userID>
            <mqmdUserID>mqm</mqmdUserID>
        </originator>
        <transferSet index="0" size="1" startTime="2011-01-26T13:03:26.484Z" total="1" bytesSent="1198">
        <item mode="binary">
            <source disposition="leave" type="file">
                <file size="1151" last-modified="2009-11-02T10:37:01.000Z">/etc/passwd</file>
                <checksum method="MD5">2287181c07199f879de28296371cb24c</checksum>
            </source>
            <destination type="file">
                <file size="1151" last-modified="2011-01-26T13:03:26.000Z">/tmp/passwd</file>
                <checksum method="MD5">2287181c07199f879de28296371cb24c</checksum>
            </destination>
            <status resultCode="0"/>
        </item>
    </transferSet>
</transaction>
```

Single file transfer success - completed

```
<?xml version="1.0" encoding="UTF-8"?>
<transaction xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            version="4.00"
            ID="414d51205553322e42494e44494e47538b0f404d223d0020"
            agentRole="sourceAgent"
            xsi:noNamespaceSchemaLocation="TransferLog.xsd"
            xmlns="">
    <action time="2011-01-26T13:03:26.622Z">completed</action>
    <sourceAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER">
        <systemInfo architecture="x86" name="Linux" version="2.6.31-21-generic"/>
    </sourceAgent>
    <destinationAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER">
        <systemInfo architecture="x86" name="Linux" version="2.6.31-21-generic"/>
    </destinationAgent>
    <originator>
        <hostName>example.com.</hostName>
        <userID>mqm</userID>
        <mqmdUserID>mqm</mqmdUserID>
    </originator>
    <status resultCode="0">
        <supplement>BFGRP0032I: The file transfer request has successfully completed.</supplement>
    </status>
    <transferSet startTime="2011-01-26T13:03:26.484Z" total="1" bytesSent="1198">
        <metaDataSet>
            <metaData key="com.ibm.wmqfte.SourceAgent">AGENT_JUPITER</metaData>
            <metaData key="com.ibm.wmqfte.DestinationAgent">AGENT_JUPITER</metaData>
            <metaData key="com.ibm.wmqfte.MqmdUser">mqm</metaData>
            <metaData key="com.ibm.wmqfte.OriginatingUser">mqm</metaData>
            <metaData key="com.ibm.wmqfte.OriginatingHost">example.com.</metaData>
            <metaData key="com.ibm.wmqfte.TransferId">414d51205553322e42494e44494e47538b0f404d223d0020</metaData>
            <metaData key="com.ibm.wmqfte.ScheduleId">3</metaData>
            <metaData key="com.ibm.wmqfte.Priority">0</metaData>
        </metaDataSet>
    </transferSet>
    <statistics>
        <actualStartTime>2011-01-26T13:03:26.541Z</actualStartTime>
        <retryCount>0</retryCount>
        <numFileFailures>0</numFileFailures>
        <numFileWarnings>0</numFileWarnings>
    </statistics>
</transaction>
```

Multiple file transfer log message examples:

Examples of the messages that are published to the SYSTEM.FTE topic with a topic string of
Log/*agent_name*/*transfer_ID* when a transfer that contains multiple files occurs.

Multiple file transfer - started

```
<?xml version="1.0" encoding="UTF-8"?>
<transaction xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             version="4.00"
             ID="414d51205553322e42494e44494e47538b0f404d035c0020"
             agentRole="sourceAgent"
             xsi:noNamespaceSchemaLocation="TransferLog.xsd"
             xmlns="">
    <action time="2011-01-26T13:12:58.534Z">started</action>
    <sourceAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER">
        <systemInfo architecture="x86" name="Linux" version="2.6.31-21-generic"/>
    </sourceAgent>
    <destinationAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER"/>
    <originator>
        <hostName>example.com</hostName>
        <userID>mqm</userID>
        <mqmdUserID>mqm</mqmdUserID>
    </originator>
    <transferSet startTime="2011-01-26T13:12:58.534Z" total="6" bytesSent="0">
        <metaDataSet>
            <metaData key="com.ibm.wmqfte.SourceAgent">AGENT_JUPITER</metaData>
            <metaData key="com.ibm.wmqfte.DestinationAgent">AGENT_JUPITER</metaData>
            <metaData key="com.ibm.wmqfte.MqmdUser">mqm</metaData>
            <metaData key="com.ibm.wmqfte.OriginatingUser">mqm</metaData>
            <metaData key="com.ibm.wmqfte.OriginatingHost">example.com</metaData>
            <metaData key="com.ibm.wmqfte.TransferId">414d51205553322e42494e44494e47538b0f404d035c0020</metaData>
            <metaData key="com.ibm.wmqfte.Priority">0</metaData>
        </metaDataSet>
    </transferSet>
</transaction>
```

Multiple file transfer - progress

```
<?xml version="1.0" encoding="UTF-8"?>
<transaction xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             version="4.00"
             ID="414d51205553322e42494e44494e47538b0f404d035c0020"
             agentRole="sourceAgent"
             xsi:noNamespaceSchemaLocation="TransferLog.xsd"
             xmlns="">
    <action time="2011-01-26T13:12:58.753Z">progress</action>
    <sourceAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER">
        <systemInfo architecture="x86" name="Linux" version="2.6.31-21-generic"/>
    </sourceAgent>
    <destinationAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER">
        <systemInfo architecture="x86" name="Linux" version="2.6.31-21-generic"/>
    </destinationAgent>
    <originator>
        <hostName>example.com.</hostName>
        <userID>mqm</userID>
        <mqmdUserID>mqm</mqmdUserID>
    </originator>
    <transferSet index="0" size="6" startTime="2011-01-26T13:12:58.534Z" total="6" bytesSent="440">
        <item mode="binary">
            <source disposition="leave" type="file">
                <file size="0" last-modified="2011-01-26T13:10:19.000Z">/srv/nfs/incoming/file01.txt</file>
                <checksum method="MD5">d41d8cd98f00b204e9800998ecf8427e</checksum>
            </source>
            <destination type="file">
                <file size="0" last-modified="2011-01-26T13:12:58.000Z">/srv/nfs/outgoing/file01.txt</file>
                <checksum method="MD5">d41d8cd98f00b204e9800998ecf8427e</checksum>
            </destination>
            <status resultCode="0"/>
        </item>
        <item mode="binary">
            <source disposition="leave" type="file">
                <file size="0" last-modified="2011-01-26T13:10:19.000Z">/srv/nfs/incoming/file02.txt</file>
                <checksum method="MD5">d41d8cd98f00b204e9800998ecf8427e</checksum>
            </source>
            <destination type="file">
                <file size="0" last-modified="2011-01-26T13:12:58.000Z">/srv/nfs/outgoing/file02.txt</file>
```

```
            <checksum method="MD5">d41d8cd98f00b204e9800998ecf8427e</checksum>
            </destination>
            <status resultCode="0"/>
        </item>
        <item mode="binary">
            <source disposition="leave" type="file">
                <file size="0" last-modified="2011-01-26T13:10:19.000Z">/srv/nfs/incoming/file03.txt</file>
                <checksum method="MD5">d41d8cd98f00b204e9800998ecf8427e</checksum>
            </source>
            <destination type="file">
                <file size="0" last-modified="2011-01-26T13:12:58.000Z">/srv/nfs/outgoing/file03.txt</file>
                <checksum method="MD5">d41d8cd98f00b204e9800998ecf8427e</checksum>
            </destination>
            <status resultCode="0"/>
        </item>
        <item mode="binary">
            <source disposition="leave" type="file">
                <file size="0" last-modified="2011-01-26T13:10:19.000Z">/srv/nfs/incoming/file04.txt</file>
                <checksum method="MD5">d41d8cd98f00b204e9800998ecf8427e</checksum>
            </source>
            <destination type="file">
                <file size="0" last-modified="2011-01-26T13:12:58.000Z">/srv/nfs/outgoing/file04.txt</file>
                <checksum method="MD5">d41d8cd98f00b204e9800998ecf8427e</checksum>
            </destination>
            <status resultCode="0"/>
        </item>
        <item mode="binary">
            <source disposition="leave" type="file">
                <file size="0" last-modified="2011-01-26T13:10:19.000Z">/srv/nfs/incoming/file05.txt</file>
                <checksum method="MD5">d41d8cd98f00b204e9800998ecf8427e</checksum>
            </source>
            <destination type="file">
                <file size="0" last-modified="2011-01-26T13:12:58.000Z">/srv/nfs/outgoing/file05.txt</file>
                <checksum method="MD5">d41d8cd98f00b204e9800998ecf8427e</checksum>
            </destination>
            <status resultCode="0"/>
        </item>
        <item mode="binary">
            <source disposition="leave" type="file">
                <file size="0" last-modified="2011-01-26T13:10:19.000Z">/srv/nfs/incoming/file06.txt</file>
                <checksum method="MD5">d41d8cd98f00b204e9800998ecf8427e</checksum>
            </source>
            <destination type="file">
                <file size="0" last-modified="2011-01-26T13:12:58.000Z">/srv/nfs/outgoing/file06.txt</file>
                <checksum method="MD5">d41d8cd98f00b204e9800998ecf8427e</checksum>
            </destination>
            <status resultCode="0"/>
        </item>
    </transferSet>
</transaction>
```

Multiple file transfer - completed

```
<?xml version="1.0" encoding="UTF-8"?>
<transaction xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             version="4.00"
             ID="414d51205553322e42494e44494e47538b0f404d035c0020"
             agentRole="sourceAgent"
             xsi:noNamespaceSchemaLocation="TransferLog.xsd"
             xmlns="">
    <action time="2011-01-26T13:12:58.766Z">completed</action>
    <sourceAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER">
        <systemInfo architecture="x86" name="Linux" version="2.6.31-21-generic"/>
    </sourceAgent>
    <destinationAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER">
        <systemInfo architecture="x86" name="Linux" version="2.6.31-21-generic"/>
    </destinationAgent>
    <originator>
        <hostName>example.com.</hostName>
        <userID>mqm</userID>
        <mqmdUserID>mqm</mqmdUserID>
    </originator>
    <status resultCode="0">
        <supplement>BFGRP0032I: The file transfer request has successfully completed.</supplement>
    </status>
    <transferSet startTime="2011-01-26T13:12:58.534Z" total="6" bytesSent="440">
        <metaDataSet>
            <metaData key="com.ibm.wmqfte.SourceAgent">AGENT_JUPITER</metaData>
```

```
            <metaData key="com.ibm.wmqfte.DestinationAgent">AGENT_JUPITER</metaData>
            <metaData key="com.ibm.wmqfte.MqmdUser">mqm</metaData>
            <metaData key="com.ibm.wmqfte.OriginatingUser">mqm</metaData>
            <metaData key="com.ibm.wmqfte.OriginatingHost">example.com.</metaData>
            <metaData key="com.ibm.wmqfte.TransferId">414d51205553322e42494e44494e47538b0f404d035c0020</metaData>
            <metaData key="com.ibm.wmqfte.Priority">0</metaData>
        </metaDataSet>
    </transferSet>
    <statistics>
        <actualStartTime>2011-01-26T13:12:58.634Z</actualStartTime>
        <retryCount>0</retryCount>
        <numFileFailures>0</numFileFailures>
        <numFileWarnings>0</numFileWarnings>
    </statistics>
</transaction>
```

Failed transfer log message examples:

When a transfer occurs, messages are published to the SYSTEM.FTE topic with a topic string of
Log/*agent_name*/*transfer_ID*. The XML examples show the log messages for a file transfer that fails being
started, in progress, and completed.

File transfer failure - started

```
<?xml version="1.0" encoding="UTF-8"?>
<transaction xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        version="4.00"
        ID="414d51205553322e42494e44494e47538b0f404d03620020"
        agentRole="sourceAgent"
        xsi:noNamespaceSchemaLocation="TransferLog.xsd"
        xmlns="">
    <action time="2011-01-26T13:19:15.767Z">started</action>
    <sourceAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER">
        <systemInfo architecture="x86" name="Linux" version="2.6.31-21-generic"/>
    </sourceAgent>
    <destinationAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER"/>
    <originator>
        <hostName>example.com.</hostName>
        <userID>mqm</userID>
        <mqmdUserID>mqm</mqmdUserID>
    </originator>
    <transferSet startTime="2011-01-26T13:19:15.767Z" total="1" bytesSent="0">
        <metaDataSet>
            <metaData key="com.ibm.wmqfte.SourceAgent">AGENT_JUPITER</metaData>
            <metaData key="com.ibm.wmqfte.DestinationAgent">AGENT_JUPITER</metaData>
            <metaData key="com.ibm.wmqfte.MqmdUser">mqm</metaData>
            <metaData key="com.ibm.wmqfte.OriginatingUser">mqm</metaData>
            <metaData key="com.ibm.wmqfte.OriginatingHost">example.com.</metaData>
            <metaData key="com.ibm.wmqfte.TransferId">414d51205553322e42494e44494e47538b0f404d03620020</metaData>
            <metaData key="com.ibm.wmqfte.Priority">0</metaData>
        </metaDataSet>
    </transferSet>
</transaction>
```

File transfer failure - progress

```
<?xml version="1.0" encoding="UTF-8"?>
<transaction xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            version="4.00"
            ID="414d51205553322e42494e44494e47538b0f404d03620020"
            agentRole="sourceAgent"
            xsi:noNamespaceSchemaLocation="TransferLog.xsd"
            xmlns="">
    <action time="2011-01-26T13:19:15.944Z">progress</action>
    <sourceAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER">
        <systemInfo architecture="x86" name="Linux" version="2.6.31-21-generic"/>
    </sourceAgent>
    <destinationAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER">
        <systemInfo architecture="x86" name="Linux" version="2.6.31-21-generic"/>
    </destinationAgent>
    <originator>
        <hostName>example.com.</hostName>
        <userID>mqm</userID>
        <mqmdUserID>mqm</mqmdUserID>
    </originator>
    <transferSet index="0" size="1" startTime="2011-01-26T13:19:15.767Z" total="1" bytesSent="0">
```

```
            <item mode="binary">
                <source disposition="leave" type="file">
                    <file size="0" last-modified="2011-01-26T13:10:19.000Z">/srv/nfs/incoming/file01.txt</file>
                    <checksum method="MD5">d41d8cd98f00b204e9800998ecf8427e</checksum>
                </source>
                <destination type="file">
                    <file>/srv/nfs/outgoing/file01.txt</file>
                </destination>
                <status resultCode="1">
                    <supplement>BFGIO0006E: File "/srv/nfs/outgoing/file01.txt" already exists.</supplement>
                </status>
            </item>
        </transferSet>
</transaction>
```

File transfer failure - completed

```
<?xml version="1.0" encoding="UTF-8"?>
<transaction xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             version="4.00"
             ID="414d51205553322e42494e44494e47538b0f404d03620020"
             agentRole="sourceAgent"
             xsi:noNamespaceSchemaLocation="TransferLog.xsd"
             xmlns="">
    <action time="2011-01-26T13:19:15.948Z">completed</action>
    <sourceAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER">
        <systemInfo architecture="x86" name="Linux" version="2.6.31-21-generic"/>
    </sourceAgent>
    <destinationAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER">
        <systemInfo architecture="x86" name="Linux" version="2.6.31-21-generic"/>
    </destinationAgent>
    <originator>
        <hostName>example.com.</hostName>
        <userID>mqm</userID>
        <mqmdUserID>mqm</mqmdUserID>
    </originator>
    <status resultCode="40">
        <supplement>BFGRP0034I: The file transfer request has
                    completed with no files being transferred.
        </supplement>
    </status>
    <transferSet startTime="2011-01-26T13:19:15.767Z" total="1" bytesSent="0">
        <metaDataSet>
            <metaData key="com.ibm.wmqfte.SourceAgent">AGENT_JUPITER</metaData>
            <metaData key="com.ibm.wmqfte.DestinationAgent">AGENT_JUPITER</metaData>
            <metaData key="com.ibm.wmqfte.MqmdUser">mqm</metaData>
            <metaData key="com.ibm.wmqfte.OriginatingUser">mqm</metaData>
            <metaData key="com.ibm.wmqfte.OriginatingHost">example.com.</metaData>
            <metaData key="com.ibm.wmqfte.TransferId">414d51205553322e42494e44494e47538b0f404d03620020</metaData>
            <metaData key="com.ibm.wmqfte.Priority">0</metaData>
        </metaDataSet>
    </transferSet>
    <statistics>
        <actualStartTime>2011-01-26T13:19:15.878Z</actualStartTime>
        <retryCount>0</retryCount>
        <numFileFailures>1</numFileFailures>
        <numFileWarnings>0</numFileWarnings>
    </statistics>
</transaction>
```

Triggered transfer message format:

When a transfer is in progress, messages are published to the SYSTEM.FTE topic with a topic string of
Log*agent_name/transfer_ID*. The XML example shows the log message that is created when a file transfer
containing a trigger condition is started.

Trigger single file transfer success - started

```
<?xml version="1.0" encoding="UTF-8"?>
<transaction version="1.00"
 ID="414d5120514d31202020202020202020207e970d492000a102" agentRole="sourceAgent"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="TransferLog.xsd"
 xmlns="">
 <action time="2008-11-02T22:05:18.703Z">started</action>
 <sourceAgent agent="FTEAGENT" QMgr="QM1">
```

```
 <systemInfo architecture="x86" name="Windows 7"
  version="6.1 build 7601 Service Pack 1" />
</sourceAgent>
<destinationAgent agent="FTEAGENT" QMgr="QM1" />
<originator>
 <hostName>reportserver.com</hostName>
 <userID>USER1</userID>
 <mqmdUserID>USER1   </mqmdUserID>
</originator>
<trigger log="yes">
 <fileExist comparison="=" value="exist">c:\trigger.txt</fileExist>
</trigger>
<transferSet startTime="2008-11-02T22:05:18.703Z" total="1"></transferSet>
</transaction>
```

Scheduled transfer log message examples:

When a transfer is in progress, messages are published to the SYSTEM.FTE topic with a topic string of Log/*agent_name*/*transfer_ID*. The XML examples show the log messages that are created when a file transfer occurs as a result of a schedule.

Schedule transfer transaction messages

When the file transfer is started as a result of the schedule entry expiring, the file transfer follows the usual sequence of publishing transaction messages on the SYSTEM.FTE/Log/*agent_ name* topic for:

- Action started (`TransferLog.xsd`)
- Action progress (`TransferLog.xsd`)
- Action completed (`TransferLog.xsd`)

Only the log transaction message with the action of started contains the ID of the scheduled transfer, in the ID attribute of the <scheduleLog> element. This allows the schedule ID to be tied to the transfer ID throughout the lifecycle of the entire transfer.

Started:
```
<?xml version="1.0" encoding="UTF-8"?>
<transaction version="1.00"
 ID="414d5120514d31202020202020202020248e294920004016" agentRole="sourceAgent"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="TransferLog.xsd"
 xmlns="">
 <action time="2008-11-23T21:55:03.111Z">started</action>
 .
 .
 .
 <scheduleLog ID="6" />
</transaction>
```

Progress:
```
<?xml version="1.0" encoding="UTF-8"?>
<transaction version="1.00"
 ID="414d5120514d31202020202020202020248e294920004016" agentRole="sourceAgent"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="TransferLog.xsd"
 xmlns="">
 <action time="2008-11-23T21:55:03.377Z">progress</action>
 .
 .
 .
</transaction>
```

Completed:
```
<?xml version="1.0" encoding="UTF-8"?>
<transaction version="1.00"
 ID="414d5120514d31202020202020202020248e294920004016" agentRole="sourceAgent"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="TransferLog.xsd"
 xmlns="">
```

```
 <action time="2008-11-23T21:55:03.424Z">completed</action>
 .
 .
 .

</transaction>
```

User exit message formats:

When a transfer is in progress, messages are published to the SYSTEM.FTE topic with a topic string of Log/*agent_name*/*transfer_ID*. The XML examples show the log messages that are created when a file transfer occurs that contains calls to user exits.

Exit single file transfer proceed - started
```
<?xml version="1.0" encoding="UTF-8"?>
<transaction version="1.00"
 ID="414d5120514d312020202020202020207e970d492000d502" agentRole="sourceAgent"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="TransferLog.xsd"
 xmlns="">
 <action time="2008-11-02T22:36:13.046Z">started</action>
 <sourceAgent agent="FTEAGENT" QMgr="QM1">
  <systemInfo architecture="x86" name="Windows 7"
   version="6.1 build 7601 Service Pack 1" />
 </sourceAgent>
 <destinationAgent agent="FTEAGENT" QMgr="QM1" />
 <originator>
  <hostName>reportserver.com</hostName>
  <userID>USER1</userID>
  <mqmdUserID>USER1         </mqmdUserID>
 </originator>
 <transferSet startTime="2008-11-02T22:36:13.046Z" total="1">
  <metaDataSet>
   <metaData key="testkey1">testvalue1</metaData>
   <metaData key="testkey2">testvalue2</metaData>
  </metaDataSet>
 </transferSet>
</transaction>
```

Exit single file transfer proceed - completed
```
<?xml version="1.0" encoding="UTF-8"?>
<transaction version="1.00"
 ID="414d5120514d312020202020202020207e970d492000d502"
  agentRole="sourceAgent"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="TransferLog.xsd"
 xmlns="">
 <action time="2008-11-02T22:36:13.546Z">completed</action>
 <sourceAgent agent="FTEAGENT" QMgr="QM1">
  <startExits>
   <exit name="class testExits.SourceExit1">
    <status resultCode="proceed">
     <supplement>Source Start, modified metadata</supplement>
    </status>
   </exit>
  </startExits>
  <endExits>
   <exit name="class testExits.SourceExit1">
    <status>
     <supplement>Source End</supplement>
    </status>
   </exit>
```

```
   </endExits>
  <systemInfo architecture="x86" name="Windows 7"
   version="6.1 build 7601 Service Pack 1" />
 </sourceAgent>
 <destinationAgent agent="FTEAGENT" QMgr="QM1">
  <startExits>
   <exit name="class testExits.DestinationExitProceed">
    <status resultCode="proceed">
     <supplement>Destination start, with proceed</supplement>
    </status>
   </exit>
  </startExits>
  <endExits>
   <exit name="class testExits.DestinationExitProceed">
    <status>
     <supplement>destination end</supplement>
    </status>
   </exit>
  </endExits>
  <systemInfo architecture="x86" name="Windows 7"
   version="6.1 build 7601 Service Pack 1" />
 </destinationAgent>
 <originator>
  <hostName>reportserver.com</hostName>
  <userID>USER1</userID>
  <mqmdUserID>USER1          </mqmdUserID>
 </originator>
 <transferSet startTime="2008-11-02T22:36:13.046Z" total="1">
  <metaDataSet>
   <metaData key="newkey2">newvalue2</metaData>
   <metaData key="newkey1">newvalue1</metaData>
   <metaData key="newkey4">newvalue4</metaData>
   <metaData key="newkey3">newvalue3</metaData>
   <metaData key="newkey5">newvalue5</metaData>
   <metaData key="testkey1">testvalue1</metaData>
   <metaData key="testkey2">testvalue2</metaData>
  </metaDataSet>
 </transferSet>
</transaction>

<!--
 In this example the source transfer start exit has modified the
  metadata as follows:

 Added keys and values for:
 newkey1, newvalue1
 newkey2, newvalue2
 newkey3, newvalue3
 newkey4, newvalue4
 newkey5, newvalue5

 Replaced values for:
 key1 to modifiedValue1

 Deleted keys and values for:
 key2
-->
```

Exit single file transfer cancel - canceled

```
<?xml version="1.0" encoding="UTF-8"?>
<transaction version="1.00"
 ID="414d5120514d31202020202020202020207e970d492000c702" agentRole="sourceAgent"
```

```
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="TransferLog.xsd"
 xmlns="">
 <action time="2008-11-02T22:25:59.328Z">cancelled</action>
 <sourceAgent agent="FTEAGENT" QMgr="QM1">
  <startExits>
   <exit name="class testExits.SourceExit1">
    <status resultCode="proceed">
     <supplement>Source Start, modified metadata</supplement>
    </status>
   </exit>
  </startExits>
  <endExits>
   <exit name="class testExits.SourceExit1">
    <status>
     <supplement>Source End</supplement>
    </status>
   </exit>
  </endExits>
  <systemInfo architecture="x86" name="Windows 7"
   version="6.1 build 7601 Service Pack 1" />
 </sourceAgent>
 <destinationAgent agent="FTEAGENT" QMgr="QM1">
  <startExits>
   <exit name="class testExits.DestinationExit1">
    <status resultCode="cancelTransfer">
     <supplement>Destination start, with cancel</supplement>
    </status>
   </exit>
  </startExits>
  <endExits>
   <exit name="class testExits.DestinationExit1">
    <status>
     <supplement>destination end</supplement>
    </status>
   </exit>
  </endExits>
  <systemInfo architecture="x86" name="Windows 7"
   version="6.1 build 7601 Service Pack 1" />
 </destinationAgent>
 <originator>
  <hostName>reportserver.com</hostName>
  <userID>USER1</userID>
  <mqmdUserID>USER1        </mqmdUserID>
 </originator>
 <transferSet startTime="2008-11-02T22:25:59.078Z" total="1" />
</transaction>
```

Additions to message formats for web-based transfers:

The Started and Completed log messages from a transfer that was requested through the WebSphere MQ Managed File Transfer Web Gateway SupportPac include extra metadata. This metadata contains information about the HTTP request and about the application server hosting the Web Gateway.

Definitions of web metadata

com.ibm.wmqfte.web.request.authtype
> The method of authorization used by the user who submits the request to the Web Gateway.

com.ibm.wmqfte.web.request.locale
> The locale of the user who submits the request to the Web Gateway.

com.ibm.wmqfte.web.appsrv.type
> The type of application server that hosts the Web Gateway.

com.ibm.wmqfte.web.appsrv.host
> The host name or IP address of the system where the application server that hosts the Web Gateway is running.

com.ibm.wmqfte.web.appsrv.port
> The port number that the application server that hosts the Web Gateway is listening on.

The metadata that is included in the log messages for a transfer that was requested through the Web Gateway is highlighted in the following examples.

Single file transfer - success

```
<?xml version="1.0" encoding="UTF-8"?>
<transaction version="1.00"
 ID="414d5120514d31202020202020202020207e970d4920008202" agentRole="sourceAgent"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="TransferLog.xsd"
 xmlns="">
 <action time="2008-11-02T21:20:37.578Z">started</action>
 <sourceAgent agent="FTEAGENT" QMgr="QM1">
  <systemInfo architecture="x86" name="Windows 7"
   version="6.1 build 7601 Service Pack 1" />
 </sourceAgent>
 <destinationAgent agent="FTEAGENT" QMgr="QM1" />
 <originator>
  <hostName>requestor.example.com</hostName>
  <userID>USER1    </userID>
  <mqmdUserID>USER1</mqmdUserID>
 </originator>
 <transferSet startTime="2008-11-02T21:20:37.593Z" total="1">
  <metaDataSet>
   <metaData key="com.ibm.wmqfte.web.request.authtype">BASIC</metaData>
   <metaData key="com.ibm.wmqfte.web.request.locale">en_GB</metaData>
   <metaData key="com.ibm.wmqfte.web.appsrv.type">
    Apache Geronimo (Embedded Tomcat/6.0.20-20090724)
   </metaData>
   <metaData key="com.ibm.wmqfte.web.appsrv.port">8080</metaData>
   <metaData key="com.ibm.wmqfte.web.appsrv.host">gateway.example.com</metaData>
  </metaDataSet>
 </transferSet>
</transaction>
```

Single file transfer success - completed

```
<?xml version="1.0" encoding="UTF-8"?>
<transaction version="1.00"
 ID="414d5120514d31202020202020202020207e970d4920008202" agentRole="sourceAgent"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="TransferLog.xsd"
 xmlns="">
 <action time="2008-11-02T21:20:38.234Z">completed</action>
 <sourceAgent agent="FTEAGENT" QMgr="QM1">
  <systemInfo architecture="x86" name="Windows 7"
   version="6.1 build 7601 Service Pack 1" />
 </sourceAgent>
 <destinationAgent agent="FTEAGENT" QMgr="QM1">
  <systemInfo architecture="x86" name="Windows 7"
   version="6.1 build 7601 Service Pack 1" />
 </destinationAgent>
 <originator>
  <hostName>requestor.example.com</hostName>
```

```
  <userID>USER1</userID>
  <mqmdUserID>USER1    </mqmdUserID>
 </originator>
 <transferSet startTime="2008-11-02T21:20:37.593Z" total="1">
  <metaDataSet>
   <metaData key="com.ibm.wmqfte.web.request.authtype">BASIC</metaData>
   <metaData key="com.ibm.wmqfte.web.request.locale">en_GB</metaData>
   <metaData key="com.ibm.wmqfte.web.appsrv.type">
    Apache Geronimo (Embedded Tomcat/6.0.20-20090724)
   </metaData>
   <metaData key="com.ibm.wmqfte.web.appsrv.port">8080</metaData>
   <metaData key="com.ibm.wmqfte.web.appsrv.host">gateway.example.com</metaData>
  </metaDataSet>
 </transferSet>
</transaction>
```

Note: Note: The XML message formats described here are not the same as the message formats that are returned as responses from the Web Gateway. The XML formats that are returned by the Web Gateway are documented in the following topic: "Response formats: XML and JSON" on page 876.

Related reference:

"Single transfer log message examples" on page 636
When a transfer occurs, messages are published to the SYSTEM.FTE topic with a topic string of Log/*agent_name*/*transfer_ID*. The XML examples show the log messages for a single file transfer being started, in progress, and completed.

"Triggered transfer message format" on page 641
When a transfer is in progress, messages are published to the SYSTEM.FTE topic with a topic string of Log/*agent_name*/*transfer_ID*. The XML example shows the log message that is created when a file transfer containing a trigger condition is started.

"User exit message formats" on page 643
When a transfer is in progress, messages are published to the SYSTEM.FTE topic with a topic string of Log/*agent_name*/*transfer_ID*. The XML examples show the log messages that are created when a file transfer occurs that contains calls to user exits.

"File transfer log message formats" on page 626
File transfer log messages are published to the SYSTEM.FTE topic with a topic string of Log/*agent_name*/*transfer_ID*. These messages conform to the schema TransferLog.xsd, which is located in the *MQ_INSTALLATION_PATH*/mqft/samples/schema directory of your WebSphere MQ Managed File Transfer installation.

Connect:Direct bridge transfer message examples:

The destinationAgent or sourceAgent element contains additional attributes when the destination agent or source agent is a Connect:Direct bridge agent. The Started log message contains only a subset of the information about the Connect:Direct transfer. The Progress and Completed log messages contain full information about the Connect:Direct transfer.

Source agent is Connect:Direct bridge agent

Started:
```
<?xml version="1.0" encoding="UTF-8"?>
<transaction xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             ID="414d5120514d5f696b6b796f20202020a704654d20092507"
             agentRole="sourceAgent"
             version="4.00"
             xsi:noNamespaceSchemaLocation="TransferLog.xsd"
             xmlns="">
   <action time="2011-03-07T13:05:01.838Z">started</action>
   <sourceAgent QMgr="QM_KUIPER" agent="VARUNA" agentType="CD_BRIDGE" bridgeNode="CDNODE_VARUNA">
       <systemInfo architecture="x86" name="Windows 7" version="6.1 build 7601 Service Pack 1"/>
   </sourceAgent>
   <destinationAgent QMgr="QM_KUIPER" agent="IXION"/>
   <originator>
```

```
        <hostName>kuiper.example.com.</hostName>
        <userID>sol</userID>
        <mqmdUserID>sol</mqmdUserID>
    </originator>
    <transferSet bytesSent="0" startTime="2011-03-07T13:05:01.838Z" total="1">
        <metaDataSet>
            <metaData key="com.ibm.wmqfte.SourceAgent">VARUNA</metaData>
            <metaData key="com.ibm.wmqfte.DestinationAgent">IXION</metaData>
            <metaData key="com.ibm.wmqfte.MqmdUser">sol</metaData>
            <metaData key="com.ibm.wmqfte.OriginatingUser">sol</metaData>
            <metaData key="com.ibm.wmqfte.OriginatingHost">kuiper.example.com.</metaData>
            <metaData key="com.ibm.wmqfte.TransferId">414d5120514d5f696b6b796f20202020a704654d20092507</metaData>
            <metaData key="com.ibm.wmqfte.Priority">0</metaData>
        </metaDataSet>
    </transferSet>
</transaction>
```

Progress:

```
<?xml version="1.0" encoding="UTF-8"?>
<transaction xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            ID="414d5120514d5f696b6b796f20202020a704654d20092507"
            agentRole="sourceAgent"
            version="4.00"
            xsi:noNamespaceSchemaLocation="TransferLog.xsd"
            xmlns="">
    <action time="2011-03-07T13:05:03.448Z">progress</action>
    <sourceAgent QMgr="QM_KUIPER" agent="VARUNA" agentType="CD_BRIDGE"
                bridgeNode="CDNODE_VARUNA" pnode="CDNODE_VARUNA" snode="CDNODE_ERIS">
        <systemInfo architecture="x86" name="Windows 7" version="6.1 build 7601 Service Pack 1"/>
    </sourceAgent>
    <destinationAgent QMgr="QM_KUIPER" agent="IXION" agentType="STANDARD">
        <systemInfo architecture="x86" name="Windows 7" version="6.1 build 7601 Service Pack 1"/>
    </destinationAgent>
    <originator>
        <hostName>kuiper.example.com.</hostName>
        <userID>sol</userID>
        <mqmdUserID>sol</mqmdUserID>
    </originator>
    <transferSet bytesSent="48" index="0" size="1" startTime="2011-03-07T13:05:01.838Z" total="1">
        <item mode="binary">
            <source disposition="leave" processName="f2007567" processNumber="68" type="file">
                <file last-modified="2011-03-07T13:05:02.573Z" size="4">CDNODE_ERIS:D:/AGENTS/CDNODE_ERIS/test.txt</file>
                <checksum method="MD5">098f6bcd4621d373cade4e832627b4f6</checksum>
            </source>
            <destination type="file">
                <file last-modified="2011-03-07T13:05:03.338Z" size="4">D:\AGENTS\IXION\test.txt</file>
                <checksum method="MD5">098f6bcd4621d373cade4e832627b4f6</checksum>
            </destination>
            <status resultCode="0"/>
        </item>
    </transferSet>
</transaction>
```

Completed:

```
<?xml version="1.0" encoding="UTF-8"?>
<transaction xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            ID="414d5120514d5f696b6b796f20202020a704654d20092507"
            agentRole="sourceAgent"
            version="4.00" xsi:noNamespaceSchemaLocation="TransferLog.xsd"
            xmlns="">
    <action time="2011-03-07T13:05:03.495Z">completed</action>
    <sourceAgent QMgr="QM_KUIPER" agent="VARUNA" agentType="CD_BRIDGE"
                bridgeNode="CDNODE_VARUNA" pnode="CDNODE_VARUNA" snode="CDNODE_ERIS">
        <systemInfo architecture="x86" name="Windows 7" version="6.1 build 7601 Service Pack 1"/>
    </sourceAgent>
    <destinationAgent QMgr="QM_KUIPER" agent="IXION" agentType="STANDARD">
        <systemInfo architecture="x86" name="Windows 7" version="6.1 build 7601 Service Pack 1"/>
    </destinationAgent>
    <originator>
        <hostName>kuiper.example.com.</hostName>
        <userID>sol</userID>
        <mqmdUserID>sol</mqmdUserID>
    </originator>
    <status resultCode="0">
        <supplement>BFGRP0032I: The file transfer request has successfully completed.</supplement>
    </status>
```

```xml
    <transferSet bytesSent="48" startTime="2011-03-07T13:05:01.838Z" total="1">
        <metaDataSet>
            <metaData key="com.ibm.wmqfte.SourceAgent">VARUNA</metaData>
            <metaData key="com.ibm.wmqfte.DestinationAgent">IXION</metaData>
            <metaData key="com.ibm.wmqfte.MqmdUser">sol</metaData>
            <metaData key="com.ibm.wmqfte.OriginatingUser">sol</metaData>
            <metaData key="com.ibm.wmqfte.OriginatingHost">kuiper.example.com.</metaData>
            <metaData key="com.ibm.wmqfte.TransferId">414d5120514d5f696b6b796f20202020a704654d20092507</metaData>
            <metaData key="com.ibm.wmqfte.Priority">0</metaData>
        </metaDataSet>
    </transferSet>
    <statistics>
        <actualStartTime>2011-03-07T13:05:02.041Z</actualStartTime>
        <retryCount>0</retryCount>
        <numFileFailures>0</numFileFailures>
        <numFileWarnings>0</numFileWarnings>
    </statistics>
</transaction>
```

Destination agent is Connect:Direct bridge agent

Started:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<transaction xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            ID="414d5120514d5f696b6b796f20202020a704654d2008e102"
            agentRole="sourceAgent"
            version="4.00"
            xsi:noNamespaceSchemaLocation="TransferLog.xsd"
            xmlns="">
    <action time="2011-03-07T10:29:44.854Z">started</action>
    <sourceAgent QMgr="QM_ASTEROID" agent="PALLAS" agentType="STANDARD">
        <systemInfo architecture="x86" name="Windows 7" version="6.1 build 7601 Service Pack 1"/>
    </sourceAgent>
    <destinationAgent QMgr="QM_ASTEROID" agent="VESTA"/>
    <originator>
        <hostName>belt.example.com.</hostName>
        <userID>sol</userID>
        <mqmdUserID>sol</mqmdUserID>
    </originator>
    <transferSet bytesSent="0" startTime="2011-03-07T10:29:44.854Z" total="1">
        <metaDataSet>
            <metaData key="com.ibm.wmqfte.SourceAgent">PALLAS</metaData>
            <metaData key="com.ibm.wmqfte.DestinationAgent">VESTA</metaData>
            <metaData key="com.ibm.wmqfte.MqmdUser">sol</metaData>
            <metaData key="com.ibm.wmqfte.OriginatingUser">sol</metaData>
            <metaData key="com.ibm.wmqfte.OriginatingHost">belt.example.com.</metaData>
            <metaData key="com.ibm.wmqfte.TransferId">414d5120514d5f696b6b796f20202020a704654d2008e102</metaData>
            <metaData key="com.ibm.wmqfte.Priority">0</metaData>
        </metaDataSet>
    </transferSet>
</transaction>
```

Progress:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<transaction xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            ID="414d5120514d5f696b6b796f20202020a704654d2008e102"
            agentRole="sourceAgent"
            version="4.00"
            xsi:noNamespaceSchemaLocation="TransferLog.xsd"
            xmlns="">
    <action time="2011-03-07T10:29:46.682Z">progress</action>
    <sourceAgent QMgr="QM_ASTEROID" agent="PALLAS" agentType="STANDARD">
        <systemInfo architecture="x86" name="Windows 7" version="6.1 build 7601 Service Pack 1"/>
    </sourceAgent>
    <destinationAgent QMgr="QM_ASTEROID" agent="VESTA" agentType="CD_BRIDGE"
                    bridgeNode="CDNODE_VESTA" pnode="CDNODE_VESTA" snode="CDNODE_HYGIEA">
        <systemInfo architecture="x86" name="Windows 7" version="6.1 build 7601 Service Pack 1"/>
    </destinationAgent>
    <originator>
        <hostName>belt.example.com</hostName>
        <userID>sol</userID>
        <mqmdUserID>sol</mqmdUserID>
    </originator>
    <transferSet bytesSent="48" index="0" size="1" startTime="2011-03-07T10:29:44.854Z" total="1">
        <item mode="binary">
```

```
            <source disposition="leave" type="file">
                <file last-modified="2011-03-04T14:53:28.323Z" size="4">D:\AGENTS\PALLAS\test.txt</file>
                <checksum method="MD5">098f6bcd4621d373cade4e832627b4f6</checksum>
            </source>
            <destination processName="f2006965" processNumber="59" type="file">
                <file size="4">CDNODE_VESTA:D:/AGENTS/CDNODE_VESTA/test.txt</file>
                <checksum method="MD5">098f6bcd4621d373cade4e832627b4f6</checksum>
            </destination>
            <status resultCode="0"/>
        </item>
    </transferSet>
</transaction>
```

Completed:

```
<?xml version="1.0" encoding="UTF-8"?>
<transaction xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             ID="414d5120514d5f696b6b796f20202020a704654d2008e102"
             agentRole="sourceAgent"
             version="4.00"
             xsi:noNamespaceSchemaLocation="TransferLog.xsd"
             xmlns="">
    <action time="2011-03-07T10:29:46.698Z">completed</action>
    <sourceAgent QMgr="QM_ASTEROID" agent="PALLAS" agentType="STANDARD">
        <systemInfo architecture="x86" name="Windows 7" version="6.1 build 7601 Service Pack 1"/>
    </sourceAgent>
    <destinationAgent QMgr="QM_ASTEROID" agent="VESTA" agentType="CD_BRIDGE"
                      bridgeNode="CDNODE_VESTA" pnode="CDNODE_VESTA" snode="CDNODE_HYGIEA">
        <systemInfo architecture="x86" name="Windows 7" version="6.1 build 7601 Service Pack 1"/>
    </destinationAgent>
    <originator>
        <hostName>belt.example.com</hostName>
        <userID>sol</userID>
        <mqmdUserID>sol</mqmdUserID>
    </originator>
    <status resultCode="0">
        <supplement>BFGRP0032I: The file transfer request has successfully completed.</supplement>
    </status>
    <transferSet bytesSent="48" startTime="2011-03-07T10:29:44.854Z" total="1">
        <metaDataSet>
            <metaData key="com.ibm.wmqfte.SourceAgent">PALLAS</metaData>
            <metaData key="com.ibm.wmqfte.DestinationAgent">VESTA</metaData>
            <metaData key="com.ibm.wmqfte.MqmdUser">sol</metaData>
            <metaData key="com.ibm.wmqfte.OriginatingUser">sol</metaData>
            <metaData key="com.ibm.wmqfte.OriginatingHost">belt.example.com.</metaData>
            <metaData key="com.ibm.wmqfte.TransferId">414d5120514d5f696b6b796f20202020a704654d2008e102</metaData>
            <metaData key="com.ibm.wmqfte.Priority">0</metaData>
        </metaDataSet>
    </transferSet>
    <statistics>
        <actualStartTime>2011-03-07T10:29:45.010Z</actualStartTime>
        <retryCount>0</retryCount>
        <numFileFailures>0</numFileFailures>
        <numFileWarnings>0</numFileWarnings>
    </statistics>
</transaction>
```

Scheduled transfer log message formats

Every time a request for a scheduled file transfer is processed by the agent, a schedule log message is published to the coordination queue manager (on its SYSTEM.FTE/Log/*agent name*/*schedule ID* topic). This message conforms to the ScheduleLog.xsd XML schema.

Schema

The following schema describes which elements are valid in a schedule log XML message.

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:include schemaLocation="fteutils.xsd"/>

    <xsd:element name="schedulelog">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="originator"      type="hostUserIDType"
                             maxOccurs="1"          minOccurs="0"/>
```

```
                <xsd:element name="action"          type="actionType"
                             maxOccurs="1"           minOccurs="1"/>
                <xsd:element name="schedule"        type="scheduleType"
                             maxOccurs="1"           minOccurs="0"/>
                <xsd:element name="sourceAgent"     type="agentType"
                             maxOccurs="1"           minOccurs="0"/>
                <xsd:element name="destinationAgent" type="agentClientType"
                             maxOccurs="1"           minOccurs="0" />
                <xsd:element name="status"          type="statusType"
                             maxOccurs="1"           minOccurs="0"/>
                <xsd:element name="transferSet"     type="transferSetType"
                             maxOccurs="1"           minOccurs="0"/>
                <xsd:element name="job"             type="jobType"
                             maxOccurs="1"           minOccurs="0" />
            </xsd:sequence>
            <xsd:attribute name="version"   type="versionType" use="required"/>
            <xsd:attribute name="ID"        type="xsd:string"  use="required"/>
        </xsd:complexType>
    </xsd:element>

    <xsd:complexType name="actionType">
        <xsd:simpleContent>
            <xsd:extension base="actionEnumType">
                <xsd:attribute name="time" type="xsd:dateTime" use="required" />
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>

    <xsd:simpleType name="actionEnumType">
        <xsd:restriction base="xsd:token">
            <xsd:enumeration value="submit"/>
            <xsd:enumeration value="delete"/>
            <xsd:enumeration value="expire"/>
            <xsd:enumeration value="skipped"/>
        </xsd:restriction>
    </xsd:simpleType>

    <xsd:complexType name="transferSetType">
        <xsd:sequence>
            <xsd:element name="item"            type="itemType"
                         maxOccurs="unbounded"  minOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="size" type="xsd:int" use="required" />
        <xsd:attribute name="priority" type="priorityType" use="optional" />
    </xsd:complexType>

    <xsd:complexType name="itemType">
        <xsd:sequence>
            <xsd:element name="source"      type="fileSourceType"
                         maxOccurs="1"      minOccurs="1" />
            <xsd:element name="destination" type="fileDestinationType"
                         maxOccurs="1"      minOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="mode" type="modeType" use="required" />
        <xsd:attribute name="checksumMethod" type="checkSumMethod" use="required" />
    </xsd:complexType>

</xsd:schema>
```

Understanding the schedule log message

The elements and attributes used in the schedule log message are described:

`<schedulelog>`
  Group element that describes a single submitted scheduled file transfer.

Attribute	Description
version	Specifies the version of this element as detailed by WebSphere MQ Managed File Transfer.
ID	The unique identifier for the submitted schedule file transfer.

`<originator>`
> Group element that contains the elements specifying the originator of the request.

`<hostName>`
> The host name of the system where the source file is located.

`<userID>`
> The user ID that originated the file transfer.

`<mqmdUserID>`
> The WebSphere MQ user ID that was supplied in the message descriptor (MQMD)

`<action>`
> Specifies the action to take with the scheduled transfer matching the ID attribute of <schedulelog> element. This element must be one of the following values:
> - submit - new scheduled transfer
> - delete - cancel schedule transfer
> - expire - schedule transfer entry about to be processed
> - skipped - a transfer that was scheduled cannot be started because the agent is offline. This message is logged when the agent becomes available to indicate the transfer was skipped.

Attribute	Description
time	Specifies the date and time the log entry was published (in date time format).

`<sourceAgent>`
> Specifies the name of the agent on the system where the source file is located.

Attribute	Description
agent	Specifies the name of the agent.
QMgr	The name of the agent queue manager.

`<destinationAgent>`
> Specifies the name of the agent on the system you want to transfer the file to.

Attribute	Description
agent	Specifies the name of the agent.
QMgr	The name of the agent queue manager.

`<status>`
> The result code and supplement messages.

`<transferSet>`
> Specifies a group of file transfers you want to perform together. During transmission <transferSet> is a group element containing <item> elements.

Attribute	Description
size	Specifies the number of transfer items.

Attribute	Description
priority	Priority level of the transfer. Priority is a value in the range 0-9, where 0 is the lowest priority. The default priority level is 0 and by default the transfer uses the priority level of the source agent.

`<item>`

Group element that contains elements specifying the source and destination file names and locations.

Attribute	Description
mode	Specifies the transfer mode as being either binary or text.
checksumMethod	Specifies the type of hash algorithm that generates the message digest to create the digital signature. Permitted values are MD5 or none

`<source>`

Group element that contains the <file> and <checksum> elements for the file on the source system.

Attribute	Description
recursive	Specifies that files are transferred recursively in subdirectories when the <source> element is a directory or contains wildcard characters.
disposition	Specifies the action that is taken on the <source> element when <source> has successfully been transferred to its destination. The valid options are as follows: • leave - the source files are left unchanged. • delete - the source files are deleted from the source system after the source file is successfully transferred.

`<destination>`

Group element that contains the <file> and <checksum> elements for the file on the destination system.

Attribute	Description
type	The type of file or directory at the destination. The valid options are as follows: • file - specifies a file as the destination • directory - specifies a directory as the destination
exist	Specifies the action that is taken if a destination file exists on the destination system. The valid options are as follows: • error - reports an error and the file is not transferred. • overwrite - overwrites the existing destination file.

`<file>`

Specifies the name of the file to transfer. Use the fully qualified path in the format consistent with your operating system, for example `C:/from/here.txt`. Do not use file URIs.

Attribute	Description
encoding	The encoding for a text file transfer.

Attribute	Description
EOL	Specifies the end of line marker. Permitted values are: • LF - line feed character only • CRLF - carriage return and line feed character sequence

<job>

Group element that contains an element specifying job details. <job> is a user-defined job name identifier that is added to the log message when the transfer has started. This <job> element is the same as the <job> element that is included in the transfer request message, which is described in the following topic: "File transfer request message format" on page 803.

<name>
The value of name can be any string.

Examples

Examples of XML messages that conform to this schema are provided for each of the following scheduled transfer actions:

- A scheduled transfer is created
- A scheduled transfer is canceled
- A schedule transfer expires

Transfers that are started by a schedule are logged in the same way as a standard transfer. For examples of log messages for transfers started by a schedule, see "Scheduled transfer log message examples" on page 642.

Schedule log examples:

Examples of the messages that are published to the SYSTEM.FTE topic with a topic string of Log/*agent_name*/*schedule_ID* when a scheduled transfer action occurs.

Scheduled transfer log message

Every time a request for a scheduled file transfer is processed by the agent, a schedule log message is published to the coordination queue manager (on its SYSTEM.FTE/Log/*agent name*/*schedule ID* topic). This message conforms to the ScheduleLog.xsd XML schema. For example:

```
<?xml version="1.0" encoding="UTF-8"?>
<schedulelog version="1.00" ID="5"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="ScheduleLog.xsd">
 <originator>
  <hostName>reportserver.com</hostName>
  <userID>USER1</userID>
 </originator>
 <action time="2008-11-23T21:32:01Z">submit</action>
 <schedule>
  <submit timebase="admin" timezone="Europe/London">2008-11-23T22:00</submit>
 </schedule>
 <sourceAgent agent="FTEAGENT" QMgr="QM1" />
 <destinationAgent agent="FTEAGENT" QMgr="QM1" />
 <status resultCode="0" />
 <transferSet size="1" priority="0">
  <item mode="binary" checksumMethod="MD5">
   <source recursive="false" disposition="leave">
    <file>c:\sourcefiles\source1.doc</file>
   </source>
   <destination type="file" exist="overwrite">
```

```
    <file>c:\destinationfiles\dest1.doc</file>
   </destination>
  </item>
 </transferSet>
</schedulelog>
```

This message is a log of the following information:

- Who originated the request
- When the request was submitted
- When the scheduled transfer starts
- The source and destination agent details
- The transfer specification

The ID attribute of the <schedulelog> element is a unique ID for this scheduled transfer (in the source agent). This ID is used to correlate schedule entries with the actual file transfers.

The <action> element value of submit confirms the request has been received.

Scheduled transfer cancel log message

When a request to cancel a pending scheduled file transfer is received by the agent, the following message is published to the SYSTEM.FTE/Log/*agent_name* topic:

```
<?xml version="1.0" encoding="UTF-8"?>
<schedulelog version="1.00" ID="5"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="ScheduleLog.xsd">
 <originator>
  <hostName>reportserver.com</hostName>
  <userID>USER1</userID>
 </originator>
 <action time="2008-11-23T21:56:27Z">delete</action>
 <status resultCode="0" />
</schedulelog>
```

The ID attribute value corresponds to the ID of the pending transfer request ID in the schedules message.

Scheduled transfer expire log message

When the current time matches the time of the earliest pending file transfer in the schedule list (as indicated by the value of the <next> element), a schedule log message is published to indicate that the scheduled transfer entry has expired:

```
<?xml version="1.0" encoding="UTF-8"?>
<schedulelog xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             version="4.00" ID="3"
             xsi:noNamespaceSchemaLocation="ScheduleLog.xsd">
    <originator>
        <hostName>example.com.</hostName>
        <userID>mqm</userID>
        <mqmdUserID>mqm</mqmdUserID>
    </originator>
    <action time="2011-01-26T13:03:26Z">expire</action>
    <sourceAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER"/>
    <status resultCode="0"/>
</schedulelog>
```

The <action> element value of "expire" confirms the schedule entry has now been removed from the schedule list and is being processed. A schedule message for the agent is published with the expired entry no longer present.

Related reference:

"Scheduled transfer log message formats" on page 650

Every time a request for a scheduled file transfer is processed by the agent, a schedule log message is published to the coordination queue manager (on its SYSTEM.FTE/Log/*agent name*/*schedule ID* topic). This message conforms to the ScheduleLog.xsd XML schema.

"Scheduled transfer log message examples" on page 642

When a transfer is in progress, messages are published to the SYSTEM.FTE topic with a topic string of Log/*agent_name*/*transfer_ID*. The XML examples show the log messages that are created when a file transfer occurs as a result of a schedule.

Monitor log message format

Monitor log messages are published to the SYSTEM.FTE topic with a topic string of Log/*agent_name*/
Monitors/*monitor_name*/*monitor_ID*.

If you want to collect data or view monitor actions, set up a subscription to a wildcard topic tailored to the monitors that you are interested in. For example:

Log/#

or,

Log/*agent_name*/#

This subscription can be durable or non-durable. Durable subscriptions continue to exist when a subscribing application's connection to the queue manager is closed. Non-durable subscriptions exist only as long as a subscribing application's connection to the queue manager remains open.

The MonitorLog.xsd schema document is located in the *MQ_INSTALLATION_PATH*/mqft/samples/schema directory. The MonitorLog.xsd schema imports fteutils.xsd, which is in the same directory.

Schema

The following schema describes which elements are valid in a monitor log XML message.

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

    <xsd:include schemaLocation="fteutils.xsd"/>

    <xsd:element name="monitorLog">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="originator"        type="hostUserIDType"         maxOccurs="1" minOccurs="0"/>
                <xsd:element name="references"        type="referencesType"         maxOccurs="1" minOccurs="0"/>
                <xsd:element name="action"            type="monitorActionType"      maxOccurs="1" minOccurs="1"/>
                <xsd:element name="monitorAgent"      type="agentType"              maxOccurs="1" minOccurs="1"/>
                <xsd:element name="status"            type="statusType"             maxOccurs="1" minOccurs="1"/>
                <xsd:element name="monitorMetaData"   type="monitorMetaDataType"    maxOccurs="1" minOccurs="0"/>
                <xsd:element name="monitorExits"      type="exitGroupType"          maxOccurs="1" minOccurs="0"/>
                <xsd:element name="jobDetails"        type="jobType"                maxOccurs="1" minOccurs="0"/>
                <xsd:element name="taskXMLRequest"    type="taskXMLRequestType"     maxOccurs="1" minOccurs="0"/>
                <xsd:element name="monitorXMLRequest" type="monitorXMLRequestType"            maxOccurs="1" minOccurs="0"/>
            </xsd:sequence>
            <xsd:attribute name="version"     type="versionType" use="required"/>
            <xsd:attribute name="monitorName" type="xsd:string"  use="required"/>
            <xsd:attribute name="referenceId" type="xsd:string"  use="optional"/>
        </xsd:complexType>
    </xsd:element>

    <xsd:complexType name="monitorActionType">
        <xsd:simpleContent>
            <xsd:extension base="monitorActionEnumType">
                <xsd:attribute name="time" type="xsd:dateTime" use="required" />
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>

    <xsd:simpleType name="monitorActionEnumType">
        <xsd:restriction base="xsd:token">
```

```
                <xsd:enumeration value="create"/>
                <xsd:enumeration value="delete"/>
                <xsd:enumeration value="start"/>
                <xsd:enumeration value="stop"/>
                <xsd:enumeration value="triggerSatisfied"/>
                <xsd:enumeration value="triggerNotSatisfied"/>
                <xsd:enumeration value="triggerFail"/>
            </xsd:restriction>
        </xsd:simpleType>

        <xsd:complexType name="monitorMetaDataType">
            <xsd:sequence>
                <xsd:element name="originalMetaData" type="metaDataSetType"  maxOccurs="1" minOccurs="0"/>
                <xsd:element name="updatedMetaData"  type="metaDataSetType"  maxOccurs="unbounded" minOccurs="0"/>
            </xsd:sequence>
        </xsd:complexType>

        <xsd:complexType name="taskXMLRequestType">
            <xsd:sequence>
                <xsd:element name="originalRequest" type="xsd:string"  maxOccurs="1" minOccurs="1"/>
                <xsd:element name="updatedRequest"  type="xsd:string"  maxOccurs="1" minOccurs="0"/>
            </xsd:sequence>
            <xsd:attribute name="taskId" type="xsd:string" use="required"/>
        </xsd:complexType>

        <xsd:complexType name="referencesType">
            <xsd:sequence>
                <xsd:element name="createRequest" type="xsd:string"  maxOccurs="1" minOccurs="0"/>
                <xsd:element name="taskRequest"   type="xsd:string"  maxOccurs="1" minOccurs="0"/>
            </xsd:sequence>
        </xsd:complexType>

        <xsd:complexType name="monitorXMLRequestType">
            <xsd:simpleContent>
                <xsd:extension base="xsd:string">
                    <xsd:attribute name="type" type="xmlContentEnumType" use="required" />
                </xsd:extension>
            </xsd:simpleContent>
        </xsd:complexType>

        <xsd:simpleType name="xmlContentEnumType">
            <xsd:restriction base="xsd:token">
                <xsd:enumeration value="escapedXML"/>
            </xsd:restriction>
        </xsd:simpleType>

</xsd:schema>
```

Understanding the monitor log message

The elements and attributes used in the monitor log messages are described in the following list:

`<monitorLog>`
　　Group element containing the elements describe an action that has been performed by a monitor.

Attribute	Description
version	Required. The version of the monitor list message format.
monitorName	Required. The name of the monitor. Unique for the agent that the monitor is defined on.
referenceId	The ID of the monitor action.

`<originator>`
　　Group element that contains the elements specifying the originator of the request.

`<hostName>`
　　The host name of the system where the source file is located.

`<userID>`
　　The user ID that originated the file transfer.

`<mqmdUserID>`
Optional. The WebSphere MQ user ID that was supplied in the message descriptor (MQMD).

`<references>`
References to the IDs of other messages associated with this monitor action.

`<createRequest>`
The message ID of the XML request message that was used to create the monitor.

`<taskRequest>`
The message ID of the XML request message that the monitor submits as a result of this action.

`<action>`
The action that occurred, which this log message is associated with. The value inside the element can be one of the following: create, delete, start, stop, triggerSatisfied, triggerNotSatisfied, or triggerFail.

`<monitorAgent>`
The agent that is monitoring the resource.

Attribute	Description
agent	Required. The name of the agent.
QMgr	Optional. The name of the queue manager that the agent connects to.
bridgeURL	Optional. If the agent is a protocol bridge agent, the URL of the protocol server.

`<status>`
The status of the resource monitor action being logged.

Attribute	Description
resultCode	Required. The integer result code from the action.

`<supplement>`
Additional information about the status of the resource monitor action being logged.

`<monitorMetaData>`
Group element that contains the <originalMetaData> and <updatedMetaData> elements.

> **`<originalMetaData>`**
> Element that contains one or more <metadata> elements that describe the metadata of the monitor before the action occurs.

> **`<updatedMetaData>`**
> Element that contains one or more <metadata> elements that describe the metadata of the monitor after the action occurs.

> **`<metadata>`**
> Defines a metadata key-value pair. The key is an attribute of the element; the value is the content of the element.

Attirbute	Description
key	The key of the metadata.

`<monitorExits>`
Group element containing one or more <exit> elements.

`<exits>`
Element describing an exit run by the resource monitor.

Attribute	Description
name	Required. The name of the resource monitor exit.

`<status>`
The status of the resource monitor exit that is being logged.

Attribute	Description
resultCode	Required. The integer result code from the exit.

`<supplement>`
Additional information about the status of the resource monitor exit that is being logged.

`<jobDetails>`
Element containing a single <name> element.

`<name>`
The name of the job..

`<taskXMLRequest>`
Group element that contains the <originalRequest> and <updatedRequest> elements.

Attribute	Description
taskId	The ID of the task request message.

`<originalRequest>`
Element that contains the escaped XML request message for the task that the monitor performs.

`<updatedRequest>`
Element that contains the updated escaped XML request message for the task that the monitor performs.

`<monitorXMLRequest>`
The monitor XML request.

Attribute	Description
type	Required. The format of the monitor XML request data inside of the <monitorXMLRequest> element. The only valid value is escapedXML.

Examples

Examples of XML messages that conform to this schema are provided for each of the following monitor actions:

- A monitor is created
- The condition of a monitor is satisfied when the monitor polls the resource
- The condition of a monitor is not satisfied when the monitor polls the resource
- A monitor is deleted

Related reference:

"Monitor log examples"
Examples of the messages that are published to the SYSTEM.FTE topic with a topic string of
Log/*agent_name*/*monitor_ID* when a monitor action occurs.

Monitor log examples:

Examples of the messages that are published to the SYSTEM.FTE topic with a topic string of
Log/*agent_name*/*monitor_ID* when a monitor action occurs.

Monitor created log message

```
<?xml version="1.0" encoding="UTF-8"?>
<monitorLog xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            version="4.00"
            monitorName="MONITORTWO"
            referenceId="414d51205553322e42494e44494e47538b0f404d04410020"
            xsi:noNamespaceSchemaLocation="MonitorLog.xsd">
    <originator>
        <hostName>example.com.</hostName>
        <userID>mqm</userID>
        <mqmdUserID>mqm</mqmdUserID>
    </originator>
    <references>
        <createRequest>414d51205553322e42494e44494e47538b0f404d04410020</createRequest>
    </references>
    <action time="2011-01-26T12:41:24Z">start</action>
    <monitorAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER"/>
    <status resultCode="0"/>
</monitorLog>
```

Monitor condition satisfied log message

```
<?xml version="1.0" encoding="UTF-8"?>
    <monitorLog xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                version="4.00"
                monitorName="MONITORONE"
                referenceId="414d51205553322e42494e44494e47538b0f404d09430020"
                xsi:noNamespaceSchemaLocation="MonitorLog.xsd">
    <originator>
        <hostName>example.com.</hostName>
        <userID>mqm</userID>
        <mqmdUserID>mqm</mqmdUserID>
    </originator>
    <references>
        <createRequest>414d51205553322e42494e44494e47538b0f404d09430020</createRequest>
    </references>
    <action time="2011-01-26T12:56:46Z">triggerSatisfied</action>
    <monitorAgent agent="US2.BINDINGS.FILE" QMgr="US2.BINDINGS"/>
    <status resultCode="0"/>
    <monitorMetaData>
        <originalMetaData>
            <metaData key="AGENTNAME">AGENT_JUPITER</metaData>
            <metaData key="LASTMODIFIEDDATEUTC">2011-01-26</metaData>
            <metaData key="CURRENTTIMESTAMPUTC">20110126125646793</metaData>
            <metaData key="CURRENTTIMESTAMP">20110126125646793</metaData>
            <metaData key="LASTMODIFIEDDATE">2011-01-26</metaData>
            <metaData key="FILENAME">new.completed</metaData>
            <metaData key="LASTMODIFIEDTIMEUTC">12.56</metaData>
            <metaData key="LASTMODIFIEDTIME">12.56</metaData>
            <metaData key="FILESIZE">0</metaData>
            <metaData key="FILEPATH">/srv/nfs/incoming/new.completed</metaData>
        </originalMetaData>
        <updatedMetaData>
            <metaData key="AGENTNAME">AGENT_JUPITER</metaData>
            <metaData key="LASTMODIFIEDDATEUTC">2011-01-26</metaData>
            <metaData key="CURRENTTIMESTAMPUTC">20110126125646793</metaData>
            <metaData key="CURRENTTIMESTAMP">20110126125646793</metaData>
            <metaData key="LASTMODIFIEDDATE">2011-01-26</metaData>
            <metaData key="FILENAME">new.completed</metaData>
            <metaData key="LASTMODIFIEDTIMEUTC">12.56</metaData>
            <metaData key="LASTMODIFIEDTIME">12.56</metaData>
            <metaData key="FILESIZE">0</metaData>
            <metaData key="FILEPATH">/srv/nfs/incoming/new.completed</metaData>
```

```
            </updatedMetaData>
        </monitorMetaData>
        <taskXMLRequest taskId="null">
            <originalRequest>&lt;?xml version="1.0" encoding="UTF-8"?&gt;&lt;request
                        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.00"
                        xsi:noNamespaceSchemaLocation="FileTransfer.xsd"&gt;&lt;managedTransfer&gt;
                        &lt;originator&gt;&lt;hostName&gt;example.com.&lt;/hostName&gt;
                        &lt;userID&gt;mqm&lt;/userID&gt;&lt;/originator&gt;
                        &lt;sourceAgent QMgr="QM_JUPITER" agent="AGENT_JUPITER"/&gt;
                        &lt;destinationAgent QMgr="QM_JUPITER" agent="AGENT_SATURN"/&gt;
                        &lt;transferSet&gt;&lt;item checksumMethod="MD5" mode="binary"&gt;
                        &lt;source disposition="leave" recursive="false"&gt;
                        &lt;file&gt;/srv/nfs/incoming/*.txt&lt;/file&gt;&lt;/source&gt;
                        &lt;destination exist="error" type="directory"&gt;
                        &lt;file&gt;/srv/backup&lt;/file&gt;&lt;/destination&gt;
                        &lt;/item&gt;&lt;/transferSet&gt;&lt;/managedTransfer&gt;&lt;/request&gt;
            </originalRequest>
            <updatedRequest>&lt;?xml version="1.0" encoding="UTF-8"?&gt;&lt;request
                        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.00"
                        xsi:noNamespaceSchemaLocation="FileTransfer.xsd"&gt;&lt;managedTransfer&gt;
                        &lt;originator&gt;&lt;hostName&gt;example.com.&lt;/hostName&gt;
                        &lt;userID&gt;mqm&lt;/userID&gt;&lt;/originator&gt;
                        &lt;sourceAgent QMgr="QM_JUPITER" agent="AGENT_JUPITER"/&gt;
                        &lt;destinationAgent QMgr="QM_JUPITER" agent="AGENT_SATURN"/&gt;
                        &lt;transferSet&gt;&lt;item checksumMethod="MD5" mode="binary"&gt;
                        &lt;source disposition="leave" recursive="false"&gt;
                        &lt;file&gt;/srv/nfs/incoming/*.txt&lt;/file&gt;
                        &lt;/source&gt;&lt;destination exist="error" type="directory"&gt;
                        &lt;file&gt;/srv/backup&lt;/file&gt;&lt;/destination&gt;
                        &lt;/item&gt;&lt;/transferSet&gt;&lt;/managedTransfer&gt;&lt;/request&gt;
            </updatedRequest>
        </taskXMLRequest>
</monitorLog>
```

Monitor condition not satisfied log message

```
<?xml version="1.0" encoding="UTF-8"?>
<monitorLog xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            version="4.00"
            monitorName="MONITORONE"
            referenceId="414d51205553322e42494e44494e47538b0f404d09430020"
            xsi:noNamespaceSchemaLocation="MonitorLog.xsd">
    <originator>
        <hostName>example.com.</hostName>
        <userID>mqm</userID>
        <mqmdUserID>mqm</mqmdUserID>
    </originator>
    <references>
        <createRequest>414d51205553322e42494e44494e47538b0f404d09430020</createRequest>
    </references>
    <action time="2011-01-26T12:58:46Z">triggerNotSatisfied</action>
    <monitorAgent agent="US2.BINDINGS.FILE" QMgr="US2.BINDINGS"/>
    <status resultCode="0"/>
</monitorLog>
```

Monitor deleted log message

```
<?xml version="1.0" encoding="UTF-8"?>
<lst:monitorList xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                 xmlns:lst="http://www.ibm.com/xmlns/wmqfte/7.0.1/MonitorDefinition"
                 version="4.00"
                 agent="AGENT_JUPITER"
                 monitor="MONITORONE"
                 xsi:schemaLocation="http://www.ibm.com/xmlns/wmqfte/7.0.1/MonitorDefinition MonitorList.xsd">
    <status state="deleted"/>
    <configuration>
        <description/>
        <resources>
            <directory recursionLevel="0" id="">/srv/nfs/incoming</directory>
        </resources>
        <triggerMatch>
            <conditions>
                <condition>
                    <name/>
                    <resource id=""/>
                    <fileMatch>
                        <pattern>*.completed</pattern>
                    </fileMatch>
```

```
                </condition>
            </conditions>
        </triggerMatch>
        <tasks>
            <task>
                <name/>
                <description/>
                <taskXML>&lt;?xml version="1.0" encoding="UTF-8"?&gt;&lt;request
                        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.00"
                        xsi:noNamespaceSchemaLocation="FileTransfer.xsd"&gt;&lt;managedTransfer&gt;
                        &lt;originator&gt;&lt;hostName&gt;example.ibm.com.&lt;/hostName&gt;
                        &lt;userID&gt;mqm&lt;/userID&gt;&lt;/originator&gt;
                        &lt;sourceAgent QMgr="QM_JUPITER" agent="AGENT_JUPITER"/&gt;
                        &lt;destinationAgent QMgr="QM_JUPITER" agent="AGENT_SATURN"/&gt;
                        &lt;transferSet&gt;&lt;item checksumMethod="MD5" mode="binary"&gt;
                        &lt;source disposition="leave" recursive="false"&gt;
                        &lt;file&gt;/srv/nfs/incoming/*.txt&lt;/file&gt;&lt;/source&gt;
                        &lt;destination exist="error" type="directory"&gt;
                        &lt;file&gt;/srv/backup&lt;/file&gt;&lt;/destination&gt;
                        &lt;/item&gt;&lt;/transferSet&gt;&lt;/managedTransfer&gt;&lt;/request&gt;
                </taskXML>
            </task>
        </tasks>
    </configuration>
    <pollInterval units="minutes">1</pollInterval>
    <batch maxSize="1"/>
</lst:monitorList>
```

Agent queues for WebSphere MQ Managed File Transfer

The MQSC command scripts generated by the **fteCreateAgent** command create the agent queues with
parameters set to the following values. If you do not use the MQSC scripts provided to create the queues,
but create the queues manually, ensure you set the following parameters to the values given.

Agent operation queues

The agent's operation queues have the following names:

- SYSTEM.FTE.COMMAND.*agent_name*
- SYSTEM.FTE.DATA.*agent_name*
- SYSTEM.FTE.EVENT.*agent_name*
- SYSTEM.FTE.REPLY.*agent_name*
- SYSTEM.FTE.STATE.*agent_name*

Table 40. Agent operation queue parameters

Parameter	Value (if applicable)
DEFPRTY	0
DEFSOPT	SHARED
GET	ENABLED
MAXDEPTH	5000
MAXMSGL	4194304
MSGDLVSQ	PRIORITY
PUT	ENABLED
RETINTVL	999999999
SHARE	
NOTRIGGER	
USAGE	NORMAL
REPLACE	

Agent authority queues

The agent's authority queues have the following names:
- SYSTEM.FTE.AUTHADM1.*agent_name*
- SYSTEM.FTE.AUTHAGT1. *agent_name*
- SYSTEM.FTE.AUTHMON1.*agent_name*
- SYSTEM.FTE.AUTHOPS1.*agent_name*
- SYSTEM.FTE.AUTHSCH1.*agent_name*
- SYSTEM.FTE.AUTHTRN1.*agent_name*

Table 41. Agent authority queue parameters

Parameter	Value (if applicable)
DEFPRTY	0
DEFSOPT	SHARED
GET	ENABLED
MAXDEPTH	0
MAXMSGL	0
MSGDLVSQ	PRIORITY
PUT	ENABLED
RETINTVL	999999999
SHARE	
NOTRIGGER	
USAGE	NORMAL
REPLACE	

Web agent operation queues

If the agent is a web agent it has two additional queues. These queues have the following names:
- SYSTEM.FTE.WEB.*gateway_name*
- SYSTEM.FTE.WEB.RESP.*agent_name*

Table 42. Web agent operation queue parameters

Parameter	Value (if applicable)
DEFPRTY	0
DEFSOPT	SHARED
GET	ENABLED
MAXDEPTH	5000
MAXMSGL	4194304
MSGDLVSQ	PRIORITY
PUT	ENABLED
RETINTVL	999999999
SHARE	
NOTRIGGER	
USAGE	NORMAL
REPLACE	

Related reference:
"fteCreateAgent (create a WebSphere MQ Managed File Transfer agent)" on page 429
The **fteCreateAgent** command creates an agent and its associated configuration.

System queues and the system topic

WebSphere MQ Managed File Transfer has a number of system queues and one system topic that are for internal use only. Do not delete these objects or change them in any way.

Any queues with a name beginning SYSTEM.FTE are internal system queues for WebSphere MQ Managed File Transfer. The topic named SYSTEM.FTE is also for internal use only. Do not modify or delete this topic or these queues, or change the queue contents in any way, because you will prevent WebSphere MQ Managed File Transfer from functioning correctly and you could lose messages.

Temporary queues

WebSphere MQ Managed File Transfer creates temporary queues for a number of purposes. The name of each queue starts with WMQFTE. by default. (The period is part of the default prefix.) If you want to change this prefix, you can use the **dynamicQueuePrefix** property in the `command.properties` file or the `coordination.properties` file or both. The property in the `command.properties` file is used to set the prefix of temporary queues that are created for responses to commands that require a response from the agent. The property in the `coordination.properties` file is used to set the prefix of temporary queues that are created for other purposes; for example, the *WMQFTE*.FTE.TIMECHCK.QUEUE, where *WMQFTE*. is the value defined by the **dynamicQueuePrefix** property.

Object naming conventions for WebSphere MQ Managed File Transfer

Use the following naming conventions for your WebSphere MQ Managed File Transfer objects:
- Agent names can be a maximum of 28 characters long and are not case-sensitive. Agent names entered in lowercase or mixed case are converted to uppercase. Agent names must conform to standard WebSphere MQ object naming conventions. These conventions are detailed as follows: Rules for naming WebSphere MQ objects.
- In addition to the WebSphere MQ object naming conventions, the forward slash (/) character cannot be used in agent names.
- In addition to the WebSphere MQ object naming conventions, the percent (%) character cannot be used in agent names.
- The names of properties in the properties files are case-sensitive.
- Queue manager names are case-sensitive.
- File names are case-sensitive for some platforms.
- Resource monitor names are not case-sensitive. Resource monitor names entered in lowercase or mixed case are converted to uppercase. Resource monitor names must not contain asterisk (*), percent (%) or question mark (?) characters.
- Protocol file server names must be a minimum of 2 characters long but there is no maximum length limit, they are not case-sensitive. Protocol server names must conform to standard WebSphere MQ object naming conventions. These conventions are detailed as follows: Rules for naming WebSphere MQ objects.

Web Gateway names
- Web Gateway names can be a maximum of 28 characters long and are not case-sensitive. Web Gateway names entered in lowercase or mixed case are converted to uppercase. Web Gateway names must conform to standard WebSphere MQ object naming conventions. These conventions are detailed in:

Rules for naming WebSphere MQ objects. In addition to the WebSphere MQ object naming conventions, the forward slash (/) character and percent (%) character cannot be used in Web Gateway names.

- If you deploy multiple instances of the same Web Gateway, use the same name for each instance.
- If you deploy more than one separate Web Gateway, use different names for each gateway. Do not create more than one Web Gateway with the same name.
- Give a web agent that is a component of a Web Gateway a similar name to the name of the Web Gateway. For example, if the Web Gateway is named WG1_GTWY, name the web agent WG1_AGNT_QM1.

Administering

Agent status values

The **fteListAgents** and **fteShowAgentDetails** commands produce agent status information. There are several possible values for this status.

ACTIVE
> The agent is running and is sending or receiving files. The agent is publishing its status at regular intervals. The last update was received within the expected time period.

READY
> The agent is running, but is not sending or receiving files. The agent is publishing its status at regular intervals. The last update was received within the expected time period.

STARTING
> The agent is starting, but is not yet ready to perform transfers.

UNREACHABLE
> Agent status updates were not received at the expected time intervals. The agent might have stopped running due to an error, or have been shut down abruptly, or be running but experiencing communication problems.

STOPPED
> The agent has been stopped. It was shut down in a controlled manner.

ENDED UNEXPECTEDLY
> The agent has ended unexpectedly. The agent will be automatically restarted, unless there have been more than maxRestartCount restarts within the maxRestartInterval time period and the maxRestartDelay value is less than or equal to 0. For more information about these properties, see "The agent.properties file" on page 548.

NO_INFORMATION
> The agent version might be WebSphere MQ File Transfer Edition Version 7.0.2 or earlier. The agent is not publishing updates in a form that this command can process.

UNKNOWN
> The status of the agent cannot be determined. It might have published a status which is not recognized by this tool. If you have mixed product versions on your network, upgrading the installation version of this tool might fix this problem.

PROBLEM
> The agent command handler might not be working. The agent is publishing status messages, but these status messages are out of date.

Use the **fteListAgents** command to list all of the WebSphere MQ Managed File Transfer agents that are registered with a particular coordination queue manager from the command line.
Use the **fteShowAgentDetails** command to display the details of a particular WebSphere MQ Managed File Transfer agent. These are the details that are stored by its WebSphere MQ Managed File Transfer coordination queue manager.
On a heavily loaded system or when there are network problems between the source and destination agents, transfers can occasionally appear to be stuck in a queued or recovering state. There are a number of factors that can cause this.

Agent process controller status values

The **fteShowAgentDetails** command produces agent process controller status information. There are several possible values for this status.

WAITING
> The agent process controller is waiting for the queue manager to become available before starting the agent.

STARTED
> The agent process controller has started the agent process.

STOPPED
> The agent process controller has been stopped, either because of a request to stop the agent or because there have been too many agent process restarts within the restart interval.

RECOVERING
> The agent process unexpectedly stopped and the process controller will attempt to restart it.

ISTOPPING
> The agent process has received a request to shut down immediately. When the agent process has stopped, the process controller will stop.

CSTOPPING
> The agent process has received a request to shut down in a controlled manner. When the agent process has stopped, the process controller will stop.

UNKNOWN
> The agent process controller status cannot be determined. It might be that the agent process controller is not running, or that it is running on a different system from where the fteShowAgentDetails command was run.

Related reference:
Use the **fteShowAgentDetails** command to display the details of a particular WebSphere MQ Managed File Transfer agent. These are the details that are stored by its WebSphere MQ Managed File Transfer coordination queue manager.

Logger status values

The **fteShowLoggerDetails** commands produce logger status information. There are several possible values for this status.

ACTIVE
> The logger is running and is sending or receiving files. The logger is publishing its status at regular intervals. The last update was received within the expected time period.

READY

The logger is running, but is not sending or receiving files. The logger is publishing its status at regular intervals. The last update was received within the expected time period.

STARTING

The logger is starting, but is not yet ready to perform transfers.

UNREACHABLE

Logger status updates were not received at the expected time intervals. The logger might have stopped running due to an error, or have been shut down abruptly, or be running but experiencing communication problems.

STOPPED

The logger has been stopped. It was shut down in a controlled manner.

ENDED UNEXPECTEDLY

The logger has ended unexpectedly. The logger will be automatically restarted, unless there have been more than `maxRestartCount` restarts within the `maxRestartInterval` time period and the `maxRestartDelay` value is less than or equal to 0. For more information about these properties, see "Logger configuration properties for WebSphere MQ Managed File Transfer" on page 144.

For the `fteShowLoggerDetails` command the details for the this status will include a status code, which is the logger process exit code. See "Process Exit Codes" for a list of known exit codes.

NO_INFORMATION

The logger version might be WebSphere MQ File Transfer Edition Version 7.0.2 or earlier. The logger is not publishing updates in a form that this command can process.

UNKNOWN

The status of the logger cannot be determined. It might have published a status which is not recognized by this tool. If you have mixed product versions on your network, upgrading the installation version of this tool might fix this problem.

PROBLEM

The logger command handler might not be working. The logger is publishing status messages, but these status messages are out of date.

Logger process controller status values

The `fteShowLoggerDetails` command produces logger process controller status information. There are several possible values for this status.

WAITING

The logger process controller is waiting for the queue manager to become available before starting the logger.

STARTED

The logger process controller has started the logger process.

STOPPED

The logger process controller has been stopped, either because of a request to stop the logger or because there have been too many logger process restarts within the restart interval.

RECOVERING

The logger process unexpectedly stopped and the process controller will attempt to restart it.

ISTOPPING

The logger process has received a request to shut down immediately. When the logger process has stopped, the process controller will stop.

CSTOPPING

The logger process has received a request to shut down in a controlled manner. When the logger process has stopped, the process controller will stop.

UNKNOWN
> The logger process controller status cannot be determined. It might be that the logger process controller is not running, or that it is running on a different system from where the fteShowLoggerDetails command was run.

Guidelines for transferring files

Depending on the operating system you are transferring from and to and whether you are transferring in binary or text mode, there are guidelines on what behavior to expect.

Read the relevant topics for further information.

Related reference:

"Transferring files and data sets between z/OS and distributed systems" on page 669
You can transfer files and supported data set types between z/OS and distributed file systems by using WebSphere MQ Managed File Transfer. Review the following behavior carefully, which is dependent on the type of system you are transferring from and to.

"Transferring between data sets" on page 670
You can transfer between z/OS data sets using WebSphere MQ Managed File Transfer. Review the following behavior carefully to ensure your data sets are transferred correctly.

"Transferring data sets to and from Connect:Direct nodes" on page 673
You can transfer data sets between WebSphere MQ Managed File Transfer agents and IBM Sterling Connect:Direct nodes using the Connect:Direct bridge. You can specify a data set as the transfer source, transfer destination, or both.

"Mappings between Connect:Direct process statement parameters and BPXWDYN keys" on page 674
When you submit a transfer request for a data set where either the source or destination is a Connect:Direct node, any supported BPXWDYN keys that you provide are converted to a format that is accepted by Connect:Direct processes.

"BPXWDYN properties you must not use with WebSphere MQ Managed File Transfer" on page 679
Some BPXWDYN options must not be specified when using the `fteCreateTemplate` command, the `fteCreateTransfer` command or the `bpxwdynAllocAdditionalProperties` property in the `agent.properties` file.

"Transferring text files" on page 680
Text file transfer involves converting the code page of a file from one code page to another. Text file transfer also involves converting CRLF (carriage return-line feed) characters between systems. This topic summarizes text file transfer behavior of WebSphere MQ Managed File Transfer.

"Transferring text files between Connect:Direct and WebSphere MQ Managed File Transfer" on page 681
Text transfer involves converting the code page of a file from one code page to another. Text file transfer also involves converting CRLF (carriage return line feed) characters between systems. This topic summarizes text file transfer behavior in transfers between an MQMFT agent and a Connect:Direct node.

"Transferring files to or from protocol bridge agents" on page 682
You can transfer files to and from an FTP or SFTP file server outside your WebSphere MQ Managed File Transfer network using a protocol bridge agent.

"Transferring generation data groups (GDGs)" on page 682
WebSphere MQ Managed File Transfer supports generation data groups (GDGs) for source and destination data sets on z/OS. Absolute and relative GDG names are supported. When you write to a new generation, the base GDG must exist.

"Using wildcard characters" on page 683
You can use wildcard characters when you specify source file names and source file paths for file transfers. This allows you to select multiple files simultaneously.

Transferring files and data sets between z/OS and distributed systems

You can transfer files and supported data set types between z/OS and distributed file systems by using WebSphere MQ Managed File Transfer. Review the following behavior carefully, which is dependent on the type of system you are transferring from and to.

z/OS support

WebSphere MQ Managed File Transfer Version 7.5 is not available for z/OS. However, this topic contains information about z/OS to allow you to transfer data sets, for example, to and from a version of WebSphere MQ File Transfer Edition running on a z/OS system.

WebSphere MQ Managed File Transfer supports generation data groups (GDGs) for source and destination data sets on z/OS. Absolute and relative GDG names are supported. When you write to a new generation, the base GDG must exist.

When you transfer a file or data set to tape, any existing data set that is already on the tape is replaced. The attributes for the new data set are set from attributes passed in the transfer definition. If no attributes are specified, attributes are set to the same as those attributes for the source data set or are set to the default values when the source is a file. The attributes of an existing tape data set are ignored.

Transferring from a file to a data set - binary transfers

The format of the destination data set determines the destination record length. Ensure the data set exists on the destination system or specify the destination data set with the correct attributes so that the data set is created properly. If you do not specify attributes, the system specifies the following default: a physical sequential data set with an undefined record format and the maximum block size (BLKSIZE) for the device (as returned by the DEVTYPE macro). For example, for DASD the size is 6144 and for tape the size is 32760. If you want to transfer a file on a distributed system to a z/OS data set in binary mode, note the following behavior:

Physical sequential (PS) destination data sets:
- The source file on the distributed system is read sequentially to fill each record or block.
- On variable format data sets, each record is filled to capacity.

Partitioned data set (PDS) destination data sets:
- Each source file is copied to a PDS member with the same or equivalent name. If the file name is longer than the maximum allowed length of a member name, the file name is converted to a valid member name. For more information about member names, see Object naming conventions. If the source file is a directory, each file in that directory becomes a member of the PDS.
- If a PDS member exists, the member is overwritten if you have specified overwrite existing destination files for the transfer. If you do not specify overwrite, the transfer fails.
- The source file on the distributed system's is read sequentially to fill each record or block for the member.
- On variable format PDS members, each record is filled to capacity.

Transferring from a file to a data set - text transfers

The format of the destination data set determines the destination record length. Ensure the data set exists on the destination system or specify the destination data set with the correct attributes so the data set is created properly. If you want to transfer from a file on a distributed system to a z/OS data set as text, note the following behavior:

Physical sequential (PS) destination data sets:
- Each line of text becomes a record (or a block for undefined record format (RECFM=U) data sets). End-of-line characters are not present in data set records (for non-ASA data sets only).

- When ASA format control characters are used in the destination data set, end-of-line characters are effectively converted to equivalent ASA format control code.
- When a line is longer than a record, the line is split at the record boundary and flows onto the next record.

PDS destination data sets:
- Each source file is copied to a PDS member with the same or equivalent name. If the file name is longer than the maximum allowed length of a member name, the file name is converted to a valid member name. For more information about member names, see Object naming conventions. If the source file is a directory, each file in that directory becomes a member of the PDS.
- If a PDS member exists, the member is overwritten if you have specified overwrite existing destination files for the transfer. If you do not specify overwrite, the transfer fails.
- Each line of text becomes a record (or a block for undefined record format (RECFM=U) data sets). End-of-line characters are not present in member records (for non-ASA data sets only).
- When ASA format control characters are used in the destination data set, end-of-line characters are effectively converted to equivalent ASA format control code.
- When a line is longer than a record, the line is split at the record boundary and flows onto the next record.

Transferring from a data set to a file - binary and text transfers

If you want to transfer from a data set to a file as binary or text, note the following behavior:

- The content of each record is transferred in binary form to a file; no record, block format information, or ASA format control characters are transferred.
- For text transfers only, each data set record becomes a line with text converted to the code page of the destination agent. That is, a carriage return-line feed (CRLF) is appended for a Windows destination system and carriage return (CR) is appended for a UNIX destination system.
- **Non-VSAM and PS source data sets**. The records for the source data set are transferred to the destination file and concatenated together. If the destination file exists, the file is overwritten, depending on the destination file behavior option you have specified for the file transfer.
- **PDS source data sets**. Each specified member, or all members if no member is specified, is extracted to the destination. If the destination specifies a directory, members are extracted to separate files. Otherwise each specified member is written to the destination file, resulting in effectively only one member being transferred. If the destination file exists for a member, the file is overwritten, depending on the destination file behavior option you have specified for the file transfer.

Related reference:

"Guidelines for transferring files" on page 668
Depending on the operating system you are transferring from and to and whether you are transferring in binary or text mode, there are guidelines on what behavior to expect.

"Transferring between data sets"
You can transfer between z/OS data sets using WebSphere MQ Managed File Transfer. Review the following behavior carefully to ensure your data sets are transferred correctly.

"**fteCreateTransfer** (create new file transfer)" on page 461
The **fteCreateTransfer** command creates and starts a new file transfer from the command line. This command can start a file transfer immediately, schedule a file transfer for a future time and date, repeat a scheduled transfer one or more times, and trigger a file transfer based on certain conditions.

Transferring between data sets
You can transfer between z/OS data sets using WebSphere MQ Managed File Transfer. Review the following behavior carefully to ensure your data sets are transferred correctly.

z/OS

WebSphere MQ Managed File Transfer Version 7.5 is not available for z/OS. However, this topic contains information about z/OS to allow you to transfer data sets to and from a version of WebSphere MQ File Transfer Edition running on a z/OS system.

WebSphere MQ Managed File Transfer does not support uncataloged data sets either on disk or tape. Existing data sets must be cataloged and new data sets will be cataloged.

Consider the following cases:

If you copy or move a data set between z/OS systems and the destination does not exist.

By default, the destination data set is created with the identical characteristics to the source. You can specify attributes for the destination data set to override the default characteristics. If you do this, a compatibility check is performed to ensure the transfer is possible.

If you copy or move a data set between z/OS systems and the destination already exists.
- If you specify attributes for the destination data set to override the default characteristics, a compatibility check is performed to ensure the destination data set can be accessed in the required way. However, you cannot override the following attributes:
  - Base data set organization and type
  - Logical record length (LRECL)
  - Block size (BLKSIZE)

If you are transferring a data set to tape.

When you transfer a data set to tape, any existing data set that is already on the tape is replaced. The attributes for the new data set are set from attributes passed in the transfer definition. If no attributes are specified, attributes are set to the same as those for the source data set or are set to the default values when the source is a file. The attributes of an existing tape data set are ignored.

Data set compatibility

Review the following behavior and restrictions for data set compatibility:

Record format and length differences:
- Variable-format records use a 4 byte record length field in the record data. Therefore for a transfer from a fixed record to a variable record data set, the variable record length must be greater than or equal to the fixed record length plus 4. For a transfer from a variable format record data set to a fixed format record data set, the fixed format record data set record length must be greater than or equal to the variable record length minus 4.

Block size differences:
- For fixed- and variable-format record data, block size differences makes the source and destination data set layout different.
- For undefined format records, provided the destination block size is greater or equal to the source data set block size, you can transfer a data set.
- For undefined format data sets, you cannot transfer if the source block size is greater than the destination block size.

Partitioned data sets (PDS) and partitioned data set extended (PDSE) data sets

The following behavior and restrictions apply equally to PDS and PDSE:
- If you transfer a PDS or PDSE member to a destination PDS or PDSE, a member of the destination PDS or PDSE is created. If the destination PDS or PDSE member already exists, the member is overwritten. If you transfer a PDS or PDSE member to a non-PDS or non-PDSE

destination data set, the destination data set is created to contain the member data. If the destination data set already exists, the data set is overwritten.

- If you attempt to transfer a PDS or PDSE to a non-PDS or non-PDSE destination, this results in all members of the PDS or PDSE being written to the non-PDSE destination. Each subsequent member transfer overwrites the previous contents of the non-PDSE destination or fails, depending on the transfer options.
- When you transfer a PDS or PDSE to a destination PDS or PDSE, a copy of the entire PDS or PDSE is created at the destination. If the destination PDS or PDSE already exists, members from the source are added. If a PDS or PDSE member already exists at the destination, the member is overwritten.
- The transfer of a non-PDS or non-PDSE to a destination PDS or PDSE, adds the contents of the non-PDS or non-PDSE as a new member of the PDS or PDSE. If the PDS member already exists, the member is overwritten. If you do not specify a name for a new member, a name is generated from the source data set or DD name.
- There is a known limitation with transfers to PDS and PDSE data sets on systems where disk space is limited. For more details, see the topic Troubleshooting WebSphere MQ File Transfer Edition .
-

Note: When you transfer a PDS or PDSE to a destination PDS or PDSE, the member information and statistics are not preserved. For example, if you transfer a load library that is stored as a PDS, the destination PDS is not usable as a load library.

Binary and text transfers

Binary transfer for data sets is defined as the record data in its binary form, as read from the data set using the default record format (type=record). Data is read and written on a record by record basis. The system service performs the necessary record and block conversion (where the data sets have different record and block settings) and the necessary ASA and machine control code conversion. If one data set is defined for ASA format control characters and the other is not appropriate, conversion to normal control codes is performed using the C/C++ system library function behavior.

Generation data groups (GDGs)

WebSphere MQ Managed File Transfer supports generation data groups (GDGs) for source and destination data sets on z/OS. Absolute and relative GDG names are supported. When you write to a new generation, the base GDG must already exist.

Related reference:
"Guidelines for transferring files" on page 668
Depending on the operating system you are transferring from and to and whether you are transferring in binary or text mode, there are guidelines on what behavior to expect.

"Transferring generation data groups (GDGs)" on page 682
WebSphere MQ Managed File Transfer supports generation data groups (GDGs) for source and destination data sets on z/OS. Absolute and relative GDG names are supported. When you write to a new generation, the base GDG must exist.

"Transferring data sets to and from Connect:Direct nodes" on page 673
You can transfer data sets between WebSphere MQ Managed File Transfer agents and IBM Sterling Connect:Direct nodes using the Connect:Direct bridge. You can specify a data set as the transfer source, transfer destination, or both.

"Transferring files and data sets between z/OS and distributed systems" on page 669
You can transfer files and supported data set types between z/OS and distributed file systems by using WebSphere MQ Managed File Transfer. Review the following behavior carefully, which is dependent on the type of system you are transferring from and to.

Transferring data sets to and from Connect:Direct nodes

You can transfer data sets between WebSphere MQ Managed File Transfer agents and IBM Sterling Connect:Direct nodes using the Connect:Direct bridge. You can specify a data set as the transfer source, transfer destination, or both.

Specifying data set names

To specify a data set on a Connect:Direct node in a transfer request, use the syntax that is used for data set transfers between WebSphere MQ Managed File Transfer agents, but with two changes:

- You must prefix the data set name with the Connect:Direct node name and a colon (:). The syntax is as follows:

cdNode:data_set_name{;*attrib1*;...;*attribN*}

  For example, to specify a partitioned data set called OBJECT.LIB on the system where the Connect:Direct node CD_NODE1 is located, use the following syntax:

`CD_NODE1://'OBJECT.LIB';RECFM(F,B);BLKSIZE(800);LRECL(80)`

  In the this example, three optional attributes are specified by the text `RECFM(F,B);BLKSIZE(800);LRECL(80)`.

- The specified data set name is interpreted as a fully qualified data set name, regardless of whether it is enclosed by single quotation mark characters. The system never adds any prefix. If you want to specify a prefix, such as the user ID that the agent runs under, you must specify it as part of the data set name. This differs from the behavior for data set transfers that involve only WebSphere MQ Managed File Transfer agents, where if the specified data set name is not enclosed by single quotation mark characters, the system adds a prefix of the default high-level qualifier for the destination agent.

Except for these two changes, specify the data set name and any optional attributes using the same syntax that is used for data set transfers between WebSphere MQ Managed File Transfer agents, which has the following rules:

- You must prefix the data set name with two forward slash characters (//).
- If you want to specify data set attributes, provide these after the data set name, separated by semicolons. Attributes must be provided in the format *key(value)*, which is suitable for BPXWDYN.

For more information about specifying data sets in a transfer request, see "**fteCreateTransfer** (create new file transfer)" on page 461 and "**fteCreateTemplate** (create new file transfer template)" on page 449.

Parameters to use in your transfer request

For most transfer requests that involve data sets on Connect:Direct nodes, you can specify the source and destination data sets in the same way as you would for a data set transfer that involves only WebSphere MQ Managed File Transfer agents. Use the **source_specification**, **-ds**, and **-dp** parameters with the **fteCreateTransfer** or **fteCreateTemplate** commands. This syntax is supported for the following scenarios:

- All the agents involved in the transfer are v7.0.4 or later
- The source agent is the Connect:Direct bridge agent, and is therefore v7.0.4 or later, and the destination agent is v7.0.3 or earlier

If the destination agent is the Connect:Direct bridge agent, and the source agent is v7.0.3 or earlier, you must make the following changes to your transfer request:

- To specify a sequential data set or partitioned data set (PDS) member as the destination of a transfer, use the **-df** parameter.
- To specify a PDS as the destination of a transfer, use the **-dd** parameter.

You can also use this syntax as an alternative to the usual **-ds** and **-dp** parameters for transfers where the source agent is v7.0.4 or later. For example, if you want to use a consistent syntax across all your scenarios and some scenarios involve a source agent that is v7.0.3 or earlier, use the **-df** and **-dd** parameters.

Note: If the destination of the transfer is a PDS and the destination agent is the Connect:Direct bridge agent, you must specify the **-de** parameter with the value of overwrite.

Specifying data set attributes

Certain data set attributes are set by WebSphere MQ Managed File Transfer and passed through as parameters to the Connect:Direct **COPY** process. You can also supply certain attributes in the transfer request, by specifying the appropriate BPXWDYN key. The Connect:Direct bridge converts keys that have equivalent Connect:Direct properties to the format that is required by Connect:Direct. For example, in the data set specification CD_NODE1://'OBJECT.LIB';RECFM(F,B);BLKSIZE(800);LRECL(80), the attributes RECFM(F,B);BLKSIZE(800);LRECL(80) are converted to DCB=(RECFM=FB,BLKSIZE=800,LRECL=80).

For details of the mappings between these two types of parameter, including details of the BPXWDYN keys that are supported for use with a Connect:Direct transfer, see "Mappings between Connect:Direct process statement parameters and BPXWDYN keys." Not all BPXWDYN keys have an equivalent Connect:Direct process parameter, and not all Connect:Direct process parameters have an equivalent BPXWDYN key.

Additional considerations

- If your transfer destination is a partitioned data set at a Connect:Direct node, you must create the partitioned data set before the transfer, because the Connect:Direct node does not create it for you.

Related concepts:

"The Connect:Direct bridge" on page 263
You can transfer files to and from an existing IBM Sterling Connect:Direct network. Use the Connect:Direct bridge, which is a component of WebSphere MQ Managed File Transfer, to transfer files between MQMFT and IBM Sterling Connect:Direct.

Related reference:

"Transferring between data sets" on page 670
You can transfer between z/OS data sets using WebSphere MQ Managed File Transfer. Review the following behavior carefully to ensure your data sets are transferred correctly.

"**fteCreateTransfer** (create new file transfer)" on page 461
The **fteCreateTransfer** command creates and starts a new file transfer from the command line. This command can start a file transfer immediately, schedule a file transfer for a future time and date, repeat a scheduled transfer one or more times, and trigger a file transfer based on certain conditions.

"**fteCreateTemplate** (create new file transfer template)" on page 449
The **fteCreateTemplate** command creates a file transfer template that you can keep for future use. The only required parameter is the **-tn** (*template_name*) parameter. All other parameters are optional, although if you specify a source file specification, you must also provide a destination file. Similarly, if you specify a destination file, you must also specify a source file specification.

"Connect:Direct file paths specified with a double forward slash" on page 394
If, as part of a file transfer, you specify a file located on a Connect:Direct node by using a file path that starts with a double forward slash (//), the file is treated as a data set.

Mappings between Connect:Direct process statement parameters and BPXWDYN keys

When you submit a transfer request for a data set where either the source or destination is a Connect:Direct node, any supported BPXWDYN keys that you provide are converted to a format that is accepted by Connect:Direct processes.

z/OS

WebSphere MQ Managed File Transfer Version 7.5 is not available for z/OS. However, this topic contains information about z/OS to allow you to transfer files to and from a version of WebSphere MQ File Transfer Edition running on a z/OS system.

For more information about IBM Sterling Connect:Direct process statements, see the Connect:Direct Process Language Reference Guide .

Table 43. Parameters to the Connect:Direct `COPY` statement, and the equivalent BPXWDYN keys used by WebSphere MQ Managed File Transfer

Parameter to Connect:Direct `COPY` statement	BPXWDYN key
DSN	DSN (valid for transfers to and from data sets). Specifying this key overrides the parameter value that is assigned by WebSphere MQ Managed File Transfer, which is based on the source or destination file specifications that are provided in the transfer request.
FILE	No mapping for data sets.
PNODE	No mapping. The primary node for the transfer is identified by WebSphere MQ Managed File Transfer. If you attempt to provide a value for this parameter, an error is produced.
SNODE	No mapping. The secondary node for the transfer is identified by WebSphere MQ Managed File Transfer. If you attempt to provide a value for this parameter, an error is produced.
DCB	See Mappings for subparameters of DCB
DISP	See Mappings for subparameters of DISP for a COPY From statement and Mappings for subparameters of DISP for a COPY To statement
RESGDG	No mapping
LABEL	See Mappings for subparameters of LABEL
MSVGP	No mapping
UNIT	UNIT
VOL	See Mappings for subparameters of VOL
ALIAS	No mapping
EXCLUDE	No mapping
PDS.DIR	No mapping. WebSphere MQ Managed File Transfer sets the value of this process parameter to N, so no user-related information that is in the directory is sent.
REPLACE | NOREPLACE	No BPXWDYN equivalent. The behavior when a destination data set already exists on the destination system is defined by the value of the **-de (destination_file_behavior)** parameter in the transfer request. For more information about the default behavior of WebSphere MQ Managed File Transfer when a destination data set already exists, see "Transferring between data sets" on page 670.
SELECT	No BPXWDYN equivalent. The data set members that are selected for copying are defined by the source file specification in the transfer request.
BUFND	No mapping

Table 43. Parameters to the Connect:Direct COPY statement, and the equivalent BPXWDYN keys used by WebSphere MQ Managed File Transfer (continued)

Parameter to Connect:Direct COPY statement	BPXWDYN key
IOEXIT	No mapping
DATAEXIT	No mapping
SYSOPTS	See Mappings for subparameters of SYSOPTS
TYPE	No mapping
AVGREC	No mapping
DATACLAS	DATACLAS
DSNTYPE	DSNTYPE. Specifying a value of PDS for this key overrides the parameter value that is assigned by WebSphere MQ Managed File Transfer, which is LIBRARY. There are no mappings for any other value - EXTPREF, EXTREQ, BASIC, or LARGE. Specifying any of these unsupported values produces an error. Specifying PDS or LIBRARY for a sequential data set produces an error.
KEYLEN	No mapping
KEYOFF	No mapping
LIKE	LIKE
LRECL	No mapping
MGMTCLAS	MGMTCLAS
RECORG	No mapping
SECMODEL	No mapping
STORCLAS	STORCLAS
SPACE	See Mappings for subparameters of SPACE
SYSOUT	No mapping
CKPT	No mapping
COMPRESS	No mapping
SECURE	No mapping

Table 44. Subparameters of the DCB parameter for the Connect:Direct COPY statement, and the equivalent BPXWDYN keys used by WebSphere MQ Managed File Transfer

Subparameters of the DCB parameter	BPXWDYN key
model-file-name	No mapping
BLKSIZE	BLKSIZE
NCP	BUFNO
DEN	No mapping
DSORG	DSORG
KEYLEN	No mapping
LIMCT	No mapping
LRECL	LRECL
OPTCD	No mapping
RECFM	RECFM
RKP	No mapping

Table 44. Subparameters of the **DCB** *parameter for the Connect:Direct* **COPY** *statement, and the equivalent BPXWDYN keys used by WebSphere MQ Managed File Transfer (continued)*

Subparameters of the **DCB** parameter	BPXWDYN key
TRTCH	TRTCH

Table 45. Subparameters of the **DISP** *parameter for the Connect:Direct* **COPY From** *statement, and the equivalent BPXWDYN keys used by WebSphere MQ Managed File Transfer*

Subparameters of the **DISP** parameter for a **COPY From** statement	BPXWDYN key	Details
[OLD ∣ SHR]	[OLD ∣ SHR]	Specifies the status of the data set before the transfer.WebSphere MQ Managed File Transfer sets this subparameter to **SHR**.
[KEEP ∣ DELETE]	[KEEP ∣ DELETE] or PATHDISP	Specifies the status of the data set after the transfer has completed successfully. The value set by WebSphere MQ Managed File Transfer depends on the source file disposition, defined by the **-sd** parameter.
[KEEP ∣ DELETE]	[KEEP ∣ DELETE] or PATHDISP	Specifies the status of the data set after the transfer has completed abnormally. WebSphere MQ Managed File Transfer sets this subparameter to **KEEP**.

Table 46. Subparameters of the **DISP** *parameter for the Connect:Direct* **COPY To** *statement, and the equivalent BPXWDYN keys used by WebSphere MQ Managed File Transfer*

Subparameters of the **DISP** parameter for a **COPY To** statement	BPXWDYN key	Details
[NEW ∣ OLD ∣ MOD ∣ RPL ∣ SHR]	[NEW ∣ OLD ∣ MOD ∣ SHR]	Specifies the status of the data set before the transfer. The value set by WebSphere MQ Managed File Transfer depends on the value of the **-de (destination_file_behavior)** parameter in the transfer request. If the destination data set does not already exist, the subparameter value is **NEW**. If the data set already exists, the subparameter value is **RPL**. WebSphere MQ Managed File Transfer does not support the key **RPL** being provided in a transfer request.
[KEEP ∣ CATLG]	[KEEP ∣ CATLOG] or PATHDISP	Specifies the status of the data set after the transfer has completed successfully. WebSphere MQ Managed File Transfer sets this subparameter to **CATLOG**.
[KEEP ∣ CATLG ∣ DELETE]	[KEEP ∣ DELETE] or PATHDISP	Specifies the status of the data set after the transfer has completed abnormally. WebSphere MQ Managed File Transfer sets this subparameter to **DELETE**.

Table 47. Subparameters of the **LABEL** *parameter for the Connect:Direct* **COPY** *statement, and the equivalent BPXWDYN keys used by WebSphere MQ Managed File Transfer*

Subparameters of the **LABEL** parameter for a **COPY** statement	BPXWDYN key	Details
file-sequence-number	SEQUENCE	
[SL | AL | BLP | LTM | NL]	LABEL(*type*)	The possible values of *type* are NL, SL, NSL, SUL, BLP, LTM, AL, and AUL. Connect:Direct accepts a subset of these values. If you specify a value that is not supported by Connect:Direct, Connect:Direct produces an error message.
[PASSWORD | NOPWREAD]	No mapping	
[IN | OUT]	No mapping	
[RETPD | EXPDT]	RETPD	EXPDT not supported

Table 48. Subparameters of the **VOL** *parameter for the Connect:Direct* **COPY** *statement, and the equivalent BPXWDYN keys used by WebSphere MQ Managed File Transfer*

Subparameters of the **VOL** parameter for a **COPY** statement	BPXWDYN key
PRIVATE	No mapping
RETAIN	No mapping
volume-sequence-no	No mapping
volume-count	MAXVOL
SER	VOL
REF	No mapping

Table 49. Subparameters of the **SYSOPTS** *parameter for the Connect:Direct* **COPY** *statement, and the equivalent BPXWDYN keys used by WebSphere MQ Managed File Transfer*

Subparameters of the **SYSOPTS** parameter for a **COPY** statement	BPXWDYN key
DBCS	No mapping
CODEPAGE	Value is dependent on WebSphere MQ Managed File Transfer transfer options. For more information, see "Transferring text files" on page 680.
DATATYPE	No mapping. WebSphere MQ Managed File Transfer sets this value to TEXT for text transfers to or from a data set, and otherwise to BINARY.
XLATE	No mapping. WebSphere MQ Managed File Transfer sets this value to NO when the value of **DATATYPE** is TEXT.
STRIP.BLANKS	No mapping. WebSphere MQ Managed File Transfer sets this value to YES when the value of **DATATYPE** is TEXT.
PERMISS	No mapping
PRECOMP	No mapping
UNIQUE	No mapping
SYSOUT	No mapping

Table 50. Subparameters of the SPACE *parameter for the Connect:Direct* COPY *statement, and the equivalent BPXWDYN keys used by WebSphere MQ Managed File Transfer*

Subparameters of the SPACE parameter for a COPY statement	BPXWDYN key
CYL	CYL
TRK	TRACKS
blk	BLOCKS
av-rec-len	No mapping
prim, [sec], [dir]	SPACE(prim[,sec]), DIR
RLSE	RELEASE
CONTIG	No mapping
ROUND	No mapping

Related concepts:

"The Connect:Direct bridge" on page 263
You can transfer files to and from an existing IBM Sterling Connect:Direct network. Use the Connect:Direct bridge, which is a component of WebSphere MQ Managed File Transfer, to transfer files between MQMFT and IBM Sterling Connect:Direct.

Related reference:

Transferring data sets to and from Connect:Direct nodes
You can transfer data sets between WebSphere MQ Managed File Transfer agents and IBM Sterling Connect:Direct nodes using the Connect:Direct bridge. You can specify a data set as the transfer source, transfer destination, or both.

BPXWDYN properties you must not use with WebSphere MQ Managed File Transfer

Some BPXWDYN options must not be specified when using the `fteCreateTemplate` command, the `fteCreateTransfer` command or the `bpxwdynAllocAdditionalProperties` property in the `agent.properties` file.

z/OS

WebSphere MQ Managed File Transfer Version 7.5 is not available for z/OS. However, this topic contains information about z/OS to allow you to transfer data sets for example, to and from a version of WebSphere MQ File Transfer Edition running on a z/OS system.

There are a number of BPXWDYN options that must not be specified with WebSphere MQ Managed File Transfer because they are used by the agent or they are not supported. If you use these options they can cause unpredictable behavior; the options are listed in the following table.

BPXWDYN options	Description
DA DSN	Specifies the data set name to allocate.
FI DD	Specifies the ddname to allocate.
FILEDATA	Specifies, to the sequential access method services, whether the data is treated as text or binary.
OLD SHR MOD NEW SYSOUT	Specifies the data set status.
REUSE	Specifies that the named data set is freed before the function is performed.
HOLD	Specifies that the output data set is to be held until released by the user or operator.
KEEP DELETE CATALOG UNCATALOG	Specifies the data set disposition after it is freed.

BPXWDYN options	Description
RECORG(LS)	Creates a VSAM linear data set.
MSG	Directs allocation messages. **Note:** This option can be used, but because WebSphere MQ Managed File Transfer uses this option to direct error information to the transfer log, using it can cause unpredictable behavior.

Transferring text files

Text file transfer involves converting the code page of a file from one code page to another. Text file transfer also involves converting CRLF (carriage return-line feed) characters between systems. This topic summarizes text file transfer behavior of WebSphere MQ Managed File Transfer.

Unless you specify otherwise, conversion is from the default code page of the file's source system to the default code page of its destination system. Additionally, text file transfer performs new line conversion, which means that new line characters for the destination file are those native to its destination platform. You can override the use of the default code pages on a system by specifying the code page to use for reading the source file and writing the destination file. You can also specify the end-of-line character sequence to use for the destination file. For more information, see the topics "**fteCreateTransfer** (create new file transfer)" on page 461 and "Using transfer definition files" on page 206.

Text file transfers perform simple code point substitutions between code pages. Text file transfers do not perform complex transfers or translations of data, for example, conversions between visual and logical forms of bidi data or text shaping.

Table 51. Text file transfer behavior for all platforms

Area	Default behavior	Can you change this behavior?
Source file encoding	Source platform encoding	Yes When you specify source file encoding and the source is a data set, the encoding must be an EBCDIC code page, otherwise the transfer fails. Similarly, if the destination is a data set, the destination encoding must be an EBCDIC code page.
Source file end of line character sequence	Convert a single (LF) or (CRLF) sequence to the destination end of line character sequence	No
Destination file encoding	Destination platform encoding	Yes When you specify source file encoding and the source is a data set, the encoding must be an EBCDIC code page, otherwise the transfer fails. Similarly, if the destination is a data set, the destination encoding must be an EBCDIC code page.

Table 51. Text file transfer behavior for all platforms (continued)

Area	Default behavior	Can you change this behavior?
Destination file end of line character sequence	Destination platform EOL	Yes
Text replacement character sequence for unmappable or malformed characters in the source or destination	Blank, meaning the transfer fails if unmappable characters or malformed characters are present. You can use the textReplacementCharacterSequence property to specify the replacement text, which is described in "The agent.properties file" on page 548.	Yes

Related reference:

"Guidelines for transferring files" on page 668
Depending on the operating system you are transferring from and to and whether you are transferring in binary or text mode, there are guidelines on what behavior to expect.

"Transferring text files between Connect:Direct and WebSphere MQ Managed File Transfer"
Text transfer involves converting the code page of a file from one code page to another. Text file transfer also involves converting CRLF (carriage return line feed) characters between systems. This topic summarizes text file transfer behavior in transfers between an MQMFT agent and a Connect:Direct node.

"Available code pages" on page 707
This reference topic lists all character encoding formats available for text file conversion on the various platforms supported by WebSphere MQ Managed File Transfer.

Transferring text files between Connect:Direct and WebSphere MQ Managed File Transfer

Text transfer involves converting the code page of a file from one code page to another. Text file transfer also involves converting CRLF (carriage return line feed) characters between systems. This topic summarizes text file transfer behavior in transfers between an MQMFT agent and a Connect:Direct node.

For information about the behavior of text transfers in WebSphere MQ Managed File Transfer, see "Transferring text files" on page 680.

- Ensure that the network map of the Connect:Direct bridge node and any Connect:Direct nodes that are used as a transfer destination include the correct platform description.
  - If your Connect:Direct bridge node is on a Windows system, ensure that for each remote node in your network map you select the correct value from the **Operating System** list.
    - If the remote node is on a Windows system, select `Windows`.
    - If the remote node is on a UNIX or Linux system, select `UNIX`.

    Transfers to remote nodes on other operating systems are not supported by the Connect:Direct bridge.
  - Ensure that for each remote node you transfer a file to or from, you specify the operating system type of the remote Connect:Direct node in the `ConnectDirectNodeProperties.xml` file in the Connect:Direct bridge agent configuration directory. For more information, see "Configure the `ConnectDirectNodeProperties.xml` file to include information about the remote Connect:Direct nodes" on page 192 and "Connect:Direct node properties file format" on page 582.

  Connect:Direct uses the network map information to determine which line ending to use.

- If the destination of a transfer is an MQMFT agent, this MQMFT agent performs the line ending conversion.
- If the destination of a transfer is a Connect:Direct node, the Connect:Direct bridge agent performs the line ending conversion.

Related reference:

"Transferring text files" on page 680
Text file transfer involves converting the code page of a file from one code page to another. Text file transfer also involves converting CRLF (carriage return-line feed) characters between systems. This topic summarizes text file transfer behavior of WebSphere MQ Managed File Transfer.

"What to do if text transfers to or from Connect:Direct nodes are not converting the data correctly" on page 394
When you transfer files in text mode between an MQMFT agent and a Connect:Direct node, code page and end-of-line character conversion is performed. The transfer uses the operating system information in the network map of the Connect:Direct bridge node to determine the end-of-line characters of a remote node. If the information in the network map is incorrect, the end-of-line character conversion might be performed incorrectly.

Transferring files to or from protocol bridge agents

You can transfer files to and from an FTP or SFTP file server outside your WebSphere MQ Managed File Transfer network using a protocol bridge agent.

When you transfer files using the protocol bridge, the bridge must have permission to read the source or destination directory containing the files you want to transfer. For example, if you want to transfer files from the directory /home/fte/bridge that has only execute permissions (d--x--x--x), any transfers you attempt from this directory fail with the following error message:

```
BFGBR0032E: Attempt to read filename from the protocol file server has failed with server error 550
Failed to open file.
```

During file transfer, files are typically written as temporary files at the destination and are then renamed when the transfer is complete. However, if the transfer destination is a protocol file server that is configured as limited write (users can upload files to the protocol file server but cannot change those uploaded files in any way; effectively users can write once only), transferred files are written to the destination directly. This means that if a problem occurs during the transfer, the partially-written files remain on the destination protocol file server and WebSphere MQ Managed File Transfer cannot delete or edit these files. In this situation the transfer fails.

Ensure that you have another agent in your WebSphere MQ Managed File Transfer network in addition to the protocol bridge agent. The protocol bridge agent is a bridge to the FTP or SFTP server only and does not write transferred files to the local disk. If you want to transfer files to or from the FTP or SFTP server you must use the protocol bridge agent as the destination or source for the file transfer (representing the FTP or SFTP server) and another standard agent as the corresponding source or destination.

Related concepts:

"The protocol bridge" on page 248
The protocol bridge enables your WebSphere MQ Managed File Transfer (MQMFT) network to access files stored on a file server outside your MQMFT network, either in your local domain or a remote location. This file server can use the FTP, FTPS, or SFTP network protocols. Each file server needs at least one dedicated agent. The dedicated agent is known as the protocol bridge agent. A bridge agent can interact with multiple file servers.

Related reference:

"Guidelines for transferring files" on page 668
Depending on the operating system you are transferring from and to and whether you are transferring in binary or text mode, there are guidelines on what behavior to expect.

Transferring generation data groups (GDGs)

WebSphere MQ Managed File Transfer supports generation data groups (GDGs) for source and destination data sets on z/OS. Absolute and relative GDG names are supported. When you write to a new generation, the base GDG must exist.

z/OS support

WebSphere MQ Managed File Transfer Version 7.5 is not available for z/OS. However, this topic contains information about z/OS to allow you to transfer GDGs, for example, to and from a version of WebSphere MQ File Transfer Edition running on a z/OS system.

Note: When creating a GDG entry in a batch environment using BASEGDG(+*n*), it cannot be referred to later in the same job by using the same positive generation number. Maintaining the same GDG entry numbers between steps of a job is a function of JCL and is not available to utility functions that update the GDG by using dynamic allocation. Therefore, a job that creates a new generation using BASEGDG(+1) would find the GDG updated as soon as the transfer successfully completes and would then need to refer to the same dataset as BASEGDG(0).

GDG examples

The following are examples of the **fteCreateTransfer** command using GDGs. In the examples, the name BASEGDG refers to an existing base GDG name. The name DSET refers to a sequential data set that is to be created. The name /u/user/file.dat refers to the name of a source data file.

This command copies file.dat into a new generation in BASEGDG. The absolute name of the new generation is reported in the transfer log:

```
fteCreateTransfer -sa A1 -da A2 -ds "//BASEGDG(+1)" /u/user/file.dat
```

This command copies file.dat into the generation with the absolute name specified in BASEGDG:

```
fteCreateTransfer -sa A1 -da A2 -ds "//BASEGDG.G0009V00" /u/user/file.dat
```

This command copies the most recent generation in BASEGDG to DSET. The absolute name of the generation is reported in the transfer log:

```
fteCreateTransfer -sa A1 -da A2 -ds "//DSET" "//BASEGDG(0)"
```

This command copies the next most recent generation in BASEGDG to DSET. The absolute name of the generation is reported in the transfer log:

```
fteCreateTransfer -sa A1 -da A2 -ds "//DSET" "//BASEGDG(-1)"
```

Related reference:

Depending on the operating system you are transferring from and to and whether you are transferring in binary or text mode, there are guidelines on what behavior to expect.

The **fteCreateTransfer** command creates and starts a new file transfer from the command line. This command can start a file transfer immediately, schedule a file transfer for a future time and date, repeat a scheduled transfer one or more times, and trigger a file transfer based on certain conditions.

You can transfer between z/OS data sets using WebSphere MQ Managed File Transfer. Review the following behavior carefully to ensure your data sets are transferred correctly.

Using wildcard characters

You can use wildcard characters when you specify source file names and source file paths for file transfers. This allows you to select multiple files simultaneously.

Distributed platforms

You can use the following wildcard characters on distributed platforms:

?   Use the question mark (?) to represent exactly one character. All of the other characters specified are required in matching file names.

For example, ab?d.jpg matches the files abcd.jpg, abed.jpg, and abfd.jpg.

*   Use the asterisk character (*) to represent zero or more characters.

For example *.txt matches the files abc.txt and x.txt.

The pattern *txt matches the files abc.txt, x.txt, and newtxt because the period (.) in the file names is a required character.

You must enclose the asterisk character (*) in double quotation marks. If you do not, the character will be interpreted by the command shell and might cause the command to fail.

If the operating system is case-insensitive for file and path names, for example Windows, the pattern match is case-insensitive. You can use wildcard characters to specify file names only: you cannot use wildcards in directory names.

Protocol bridge agent

If you are using a protocol bridge agent to transfer files from an FTP, FTPS, or SFTP file server, wildcard matching is case sensitive, regardless of the platform that the file server is actually running on.

Connect:Direct bridge

When the source of a transfer is a Connect:Direct bridge agent that is requesting files from a Connect:Direct node, wildcards are not supported.

Directories

By default if you create a file transfer with a wildcard pattern that matches subdirectories, the subdirectories are not transferred. You can specify the **-r** parameter on the fteCreateTransfer command to include subdirectories that match the wildcard pattern. When you transfer a subdirectory, the entire contents and structure of the subdirectory are transferred: including all of its files, subdirectories, and hidden files.

For example, if you have a directory called abc, there is a difference in behavior between specifying a source file path of /opt/abc and /opt/abc/*. In the case of /opt/abc because the directory is transferred, a directory called abc is created at the destination and all of the file contents are transferred. In the case of /opt/abc/*, the contents of abc are transferred into the destination path.

Hidden files

Wildcards do not match hidden files except on UNIX-type platforms when the wildcard pattern starts with a dot character (.). For example: /opt/.* transfers all hidden files in the opt directory.

On Windows if you want to transfer a hidden file, either specify the file name exactly or transfer the directory containing the hidden file.

Symbolic links

Symbolic links are a type of file that contain a pointer to another file or directory and are known as shortcuts on Windows. You can match symbolic link files with wildcard characters. However, when a destination file is created from a source that is a symbolic link, the destination file becomes a hard link (that is, a regular file). You cannot successfully transfer symbolic links to directories because this could potentially create a recursive path.

Transferring files with wildcard characters in their file names

You can transfer a file if the file name itself contains a wildcard character. If you specify that file name exactly, only that file is transferred, and not the set of files that match the wildcard.

For example, if you have a file called /opt/abc*.txt and you create a file transfer for /opt/abc*.txt, the only file transferred is /opt/abc*.txt. But if you create a file transfer for /opt/ab*.txt, all files matching the pattern /opt/ab*.txt are transferred, including the file /opt/abc*.txt.

Transferring directory paths that contain wildcard characters

Enclose any directory path that includes a wildcard character in quotation marks (" ") or single quotation marks (' ') to avoid shell expansion. Shell expansion happens when the operating system expands the wildcard character before the character is passed to the WebSphere MQ Managed File Transfer command and this might cause unexpected behavior.

For example, if you run the following **fteCreateTransfer** command with the **-gt** parameter on UNIX, where ${...} is a variable substitution from a resource monitor:

```
fteCreateTransfer -p QM_VENUS -sa AGT.QM_JUPITER -sm QM_JUPITER -da AGT.QM_NEPTUNE -dm QM_NEPTUNE -r -sd delete
-t binary -de overwrite —jn MONTASK -gt /home/fteadmin/bin/TransferTask.xml -df "${FilePath}" "${FilePath}"
```

the shell parses ${FilePath} and does not pass it to the command. The workaround is to enclose ${FilePath} in double quotation marks, that is, "${FilePath}".

Transfer is reported as successful even though wildcard matches zero files

If you attempt to transfer a file that does not exist, WebSphere MQ Managed File Transfer treats this attempt as a failed transfer. If you specify a file name explicitly (for example, /a/missing/filename.txt) and MQMFT is unable to find that file, the following error message is reported in the log:

```
BFGIO0001E: File "/a/missing/filename.txt" does not exist
```

As part of this process the source agent, which could not find the file, notifies the destination agent that this file transfer has been canceled (because the source agent cannot find the source file to read). If you had planned to trigger an exit after the transfer at this point, the destination agent triggers its DestinationTransferEndExit with a FileExitResultCode of CANCEL_FILE for that file name.

However, if you attempt to transfer a wildcard (for example, /a/missing/*.txt) and the source agent does not find any files that match that wildcard, MQMFT reports this as a successful transfer. This is because technically the source agent was asked to transfer 0 files. The following error message is reported in the log:

```
The transfer request has successfully completed, although no files were transferred.
```

In this example, because the destination agent was never involved in the transfer, its exit is not called.

Related reference:

"Guidelines for transferring files" on page 668
Depending on the operating system you are transferring from and to and whether you are transferring in binary or text mode, there are guidelines on what behavior to expect.

"**fteCreateTransfer** (create new file transfer)" on page 461
The **fteCreateTransfer** command creates and starts a new file transfer from the command line. This command can start a file transfer immediately, schedule a file transfer for a future time and date, repeat a scheduled transfer one or more times, and trigger a file transfer based on certain conditions.

Regular expressions used by WebSphere MQ Managed File Transfer

WebSphere MQ Managed File Transfer uses regular expressions in a number of scenarios. For example, regular expressions are used to match user IDs for Connect:Direct security credentials, or to split a file

into multiple messages by creating a new message each time a regular expression is matched. The regular expression syntax used by WebSphere MQ Managed File Transfer is the syntax supported by the `java.util.regex` API. This regular expression syntax is similar to, but not the same as, the regular expression syntax used by the Perl language.

For more information about Java regular expressions, see the Java tutorial Regular Expressions.

Examples

To match all patterns, use the following regular expression:
`.*`

To match all patterns that begin with the string `fte`, use the following regular expression:
`fte.*`

To match all patterns that begin with the string `accounts` followed by a single digit, and end with `.txt`, use the following regular expression:
`accounts[0-9]\.txt`

Substitution variables for use with user-defined Connect:Direct processes

You can define values to substitute in to user-defined Connect:Direct processes by using intrinsic symbolic variables that are specific to WebSphere MQ Managed File Transfer.

To follow the Connect:Direct naming convention, all intrinsic symbolic variables used by WebSphere MQ Managed File Transfer have the format %FTE followed by five uppercase alphanumeric characters. For more information about intrinsic symbolic variables, see the Connect:Direct product documentation.

When creating a process to transfer files from a Connect:Direct node to the Connect:Direct bridge system, you must use the intrinsic variable %FTETFILE as the value of TO FILE in the Connect:Direct process. When creating a process to transfer files to a Connect:Direct node from the Connect:Direct bridge system, you must use the intrinsic variable %FTEFFILE as the value of FROM FILE in the Connect:Direct process. These variables contain the temporary file paths that the Connect:Direct bridge agent uses for transfers into and out of the WebSphere MQ Managed File Transfer network.

Table 52. Intrinsic symbolic variables used by WebSphere MQ Managed File Transfer and Connect:Direct

Variable name	Description
%FTESAGNT	The name of the WebSphere MQ Managed File Transfer source agent. This variable is set only for transfers from a WebSphere MQ Managed File Transfer agent to a Connect:Direct node.
%FTEDAGNT	The name of the WebSphere MQ Managed File Transfer destination agent. This variable is set only for transfers from a Connect:Direct node to a WebSphere MQ Managed File Transfer agent.
%FTEPNODE	The Connect:Direct primary node name. The value is always the name of the Connect:Direct node that is part of the Connect:Direct bridge.
%FTEPPLAT	The platform that the Connect:Direct primary node runs on. Possible values for this variable are UNIX and WINDOWS. This information is provided by the Connect:Direct bridge agent.
%FTEPUSER	The Connect:Direct primary node user identifier to use in the Connect:Direct process. This information is taken from the `ConnectDirectCredentials.xml` file.
%FTEPPASS	The password to use with the user name defined by the %FTEPUSER variable. This information is taken from the `ConnectDirectCredentials.xml` file.

Variable name	Description
%FTESNODE	The Connect:Direct secondary node name. The value is always the name of the Connect:Direct node that the file is transferred to or from.
%FTESPLAT	The platform that the Connect:Direct secondary node runs on. Possible values for this variable are UNIX, WINDOWS, and ZOS. This information is taken from the `ConnectDirectNodeProperties.xml` file.
%FTESUSER	The Connect:Direct secondary node user identifier to use in the Connect:Direct process. This information is taken from the `ConnectDirectCredentials.xml` file.
%FTESPASS	The password to use with the user name defined by the %FTESUSER variable. This information is taken from the `ConnectDirectCredentials.xml` file.
%FTEFFILE	The source file name. This variable is set only for Connect:Direct processes that are submitted at a per-file scope. When transferring files from a WebSphere MQ Managed File Transfer agent to a Connect:Direct node, the value is the fully qualified location of the file on the same system as the Connect:Direct bridge. When transferring files from a Connect:Direct node to a WebSphere MQ Managed File Transfer agent, the value is the name of the file that is specified as the source file in the WebSphere MQ Managed File Transfer transfer request.
%FTEFDISP	The disposition of the source file when the process is complete. The value of this variable is platform dependent and equivalent to the values for MQMFT transfer request. This variable is set only for Connect:Direct processes that are submitted at a per-file scope. When transferring files from a WebSphere MQ Managed File Transfer agent to a Connect:Direct node, the action of deleting or not deleting the source file is performed by the WebSphere MQ Managed File Transfer bridge agent. When transferring files from a Connect:Direct node to a WebSphere MQ Managed File Transfer agent, the action of deleting or not deleting the source file must be performed by the Connect:Direct process.
%FTEFCP	The code page to use for the source file. This variable is set only for Connect:Direct processes that are submitted at a per-file scope. When transferring files from a WebSphere MQ Managed File Transfer agent to a Connect:Direct node, this value is UTF-8 or, if the transfer is a binary transfer, the value is not set. When transferring files from a Connect:Direct node to a WebSphere MQ Managed File Transfer agent, this value is specified by Connect:Direct or, if the transfer is a binary transfer, the value is not set.
%FTEFSYSO	The Connect:Direct SYSOPTS for the source of the transfer. If the remote Connect:Direct node is on Linux, UNIX, or Windows, this value contains information about the code page and data type of the source of the transfer.
%FTEFNODE	Identifies the Connect:Direct node where the source file resides. This will be set to a value of either: PNODE or SNODE.
%FTETFILE	The destination file name. This variable is set only for Connect:Direct processes that are submitted at a per-file scope. When transferring files from a WebSphere MQ Managed File Transfer agent to a Connect:Direct node, the value is the name of the file that is specified as the destination file in the WebSphere MQ Managed File Transfer transfer request. When transferring files from a Connect:Direct node to a WebSphere MQ Managed File Transfer agent, the value is the fully qualified name of the location to write the file to on the same system as the Connect:Direct bridge.

Variable name	Description
%FTETDISP	The disposition of the destination file. The value of this variable is platform dependent and equivalent to the values for MQMFT transfer request. This variable is set only for Connect:Direct processes that are submitted at a per-file scope.
	When transferring files from a WebSphere MQ Managed File Transfer agent to a Connect:Direct node, the action of creating a file or replacing an existing file must be performed by the Connect:Direct process.
	When transferring files from a Connect:Direct node to a WebSphere MQ Managed File Transfer agent, the action of creating a file or replacing an existing file is performed by the WebSphere MQ Managed File Transfer bridge agent.
%FTETCP	The code page to use for the destination file. This variable is set only for Connect:Direct processes that are submitted at a per-file scope.
	When transferring files from a WebSphere MQ Managed File Transfer agent to a Connect:Direct node, this value is specified by Connect:Direct or, if the transfer is a binary transfer, the value is not set.
	When transferring files from a Connect:Direct node to a WebSphere MQ Managed File Transfer agent, this value is UTF-8 or, if the transfer is a binary transfer, the value is not set.
%FTETSYSO	The Connect:Direct SYSOPTS for the destination of the transfer. If the remote Connect:Direct node is on Linux, UNIX, or Windows, this value contains information about the code page and data type of the destination of the transfer.
%FTETNODE	Identifies the Connect:Direct node where the destination file is to reside. This will be set to a value of either: PNODE or SNODE.
%FTEDTYPE	The data type or mode of the transfer. Possible values for this variable are `text` or `binary`. This variable is set only for Connect:Direct processes that are submitted at a per-file scope.
%FTETRID	The 48-character hexadecimal transfer ID from the WebSphere MQ Managed File Transfer transfer.
%FTEJOBN	The job name from the WebSphere MQ Managed File Transfer transfer request. The value of this variable is truncated to 256 characters and can be used in the process accounting data.
%FTEPNAME	The Connect:Direct process name generated by the WebSphere MQ Managed File Transfer bridge agent. The value of this variable is 8 alphanumeric characters. The value always starts with an alphabetic character.
%FTEMETA(*key*)	A metadata from the WebSphere MQ Managed File Transfer transfer request. The value of *key* is the key of the metadata. The value of *key* is not case sensitive. A key of `ABC` is the treated the same as a key of `abc`. If both `ABC` and `abc` are defined as metadata keys, the value of the second metadata defined overwrites the value of the first metadata defined.

Example of a Connect:Direct process file that calls the ftecxfer command

An example Connect:Direct process file that calls the WebSphere MQ Managed File Transfer **ftetag** command and the **ftecxfer** command.

In this example, the following actions occur:

1. A Connect:Direct COPY statement transfers the file from `C:\test\from\sent.txt` on the system where the secondary node runs to `C:\test\tmp\midpoint.txt` on the system where the primary node runs.
2. The Connect:Direct process calls the **ftetag** command to create audit information in MQMFT.

3. The Connect:Direct process calls the **ftecxfer** command.
4. The **ftecxfer** command transfers the file from `C:\test\tmp\midpoint.txt` on the system where the primary node runs and the agent CD_BRIDGE runs to `/test/to/arrived.txt` on the system where the agent LINUX_AGENT is located.

```
/*BEGIN_REQUESTER_COMMENTS
    $PNODE$="cd_win01" $PNODE_OS$="Windows"
    $SNODE$="CD_WIN01" $SNODE_OS$="Windows"
    $OPTIONS$="WDOS"
  END_REQUESTER_COMMENTS*/

TESTPRO PROCESS
 SNODE=CD_WIN01

 COPY
 FROM (
  FILE=C:\test\from\sent.txt
  SNODE
 )
 TO (
  FILE=C:\test\tmp\midpoint.txt
  PNODE
  DISP=RPL
 )
 COMPRESS Extended

 RUN TASK PNODE
 SYSOPTS="pgm(C:\wmqfte\bin\ftetag) args(C:\test\tmp\midpoint.txt)"

 RUN TASK PNODE
 SYSOPTS="pgm(C:\wmqfte\bin\ftecxfer) args(-qmgrname QM_CDBA -connname fish.example.com(1441) -channelname SYSTEM.DEF.SVRCONN
-sa CD_BRIDGE -da LINUX_AGENT -sm QM_CDBA -dm QM_LINUX -de overwrite -df /test/to/arrived.txt C:\test\tmp\midpoint.txt"

PEND
```

Related concepts:

"Using Connect:Direct processes to submit WebSphere MQ Managed File Transfer transfer requests" on page 273
You can submit a transfer request to the Connect:Direct bridge agent from a Connect:Direct process. WebSphere MQ Managed File Transfer provides commands that can be called from a **RUN TASK** statement in a Connect:Direct process.

Related tasks:

"Creating and submitting a Connect:Direct process that calls WebSphere MQ Managed File Transfer by using the Connect:Direct Requester" on page 274
The Connect:Direct Requester is a graphical user interface that you can use to create and submit a Connect:Direct process that calls WebSphere MQ Managed File Transfer.

Restrictions of the Connect:Direct bridge agent

The Connect:Direct bridge agent is configured to transfer files to and from Connect:Direct nodes. There are some functions that the Connect:Direct bridge agent is not capable of performing.

- The Connect:Direct bridge agent cannot read messages from a queue or write messages to a queue. It cannot act as the destination agent in a file-to-message transfer or as the source agent in a message-to-file transfer.
- You cannot define a resource monitor on the Connect:Direct bridge agent.
- You cannot have a Connect:Direct bridge agent as both the source and destination of a transfer. You cannot transfer from Connect:Direct node to Connect:Direct node through the Connect:Direct bridge.
- The Connect:Direct bridge agent does not support user exits that are called before or after the transfer. The Connect:Direct bridge agent does support a credential mapping exit. For more information, see "Mapping credentials for Connect:Direct by using exit classes" on page 195.
- You cannot define presrc or postsrc program invocations for a transfer that has the Connect:Direct bridge agent as the source agent. For more information, see "Program invocation nested elements" on page 917.

- You cannot define predst or postdst program invocations for a transfer that has the Connect:Direct bridge agent as the destination agent. For more information, see "Program invocation nested elements" on page 917.
- You cannot specify a wildcard character in the source specification if the source agent is the Connect:Direct bridge agent.
- If you specify a source disposition (**-sd**) of `delete` when transferring a file from a Connect:Direct node, the behavior is different to the usual source disposition behavior. One of the following cases occurs:
  - If Connect:Direct uses a process that is generated by WebSphere MQ Managed File Transfer to move the file or data set from the source, specifying the `delete` option causes the transfer to fail. To specify that the source file is deleted, you must submit a user-defined Connect:Direct process. For more information, see "Submitting a user-defined Connect:Direct process from a file transfer request" on page 269.
  - If Connect:Direct uses a user-defined process to move the file or data set from the source, this parameter is passed to the process through the **%FTEFDISP** intrinsic symbolic variable. The user-defined process determines whether the source is deleted. The result that the transfer returns depends on the result that is returned by the user-defined process.

Related concepts:
"The Connect:Direct bridge" on page 263
You can transfer files to and from an existing IBM Sterling Connect:Direct network. Use the Connect:Direct bridge, which is a component of WebSphere MQ Managed File Transfer, to transfer files between MQMFT and IBM Sterling Connect:Direct.

FTPS server support by the protocol bridge

The protocol bridge supports a subset of the FTPS protocol as defined by RFC-2228, RFC-4217, and the Internet-Draft entitled *Secure FTP over SSL*.

For a list of valid cipher suite values for connections between protocol bridge agents and FTPS servers, see Cipher suites in the IBM SDK and Runtime Environment Java Technology Edition Version 7 product documentation.

The following features of the FTPS protocol are supported:
- Implicit and explicit modes of operation.
- Validation of the server certificate.
- Optional mutual authentication using client certificate checks.
- Optional use of a clear control channel after the initial authentication and level of protection for the data channel has been selected.

The following features of the FTPS protocol and runtime environment are not supported:
- Use of the **ADAT** command for additional security data exchange.
- Use of FTPS for channel encryption only that is, where the servers certificate is not validated.
- Selection of the `Clear`, `Secure`, or `Confidential` levels of protection using the **PROT** command.
- Encryption for each command using the **MIC**, **CONF**, and **ENC** commands.
- Fallback to the FTP protocol if the server does not support explicit FTPS. Use the FTP support provided by the protocol bridge to work with such a server.
- Use of the **FEAT** command to determine the available capabilities of the FTPS server.
- Validation of certificates using pattern matching against the DN field.
- Certificate revocation checking.
- Validation of certificates with the issuing trusted certificate authority.
- Explicit selection of the cipher suites available to the SSL negotiation phase of establishing a session.

- Restricting the encryption used to the encryption provided by a FIPS 140-2-accredited cryptographic module.

Related concepts:

"The protocol bridge" on page 248
The protocol bridge enables your WebSphere MQ Managed File Transfer (MQMFT) network to access files stored on a file server outside your MQMFT network, either in your local domain or a remote location. This file server can use the FTP, FTPS, or SFTP network protocols. Each file server needs at least one dedicated agent. The dedicated agent is known as the protocol bridge agent. A bridge agent can interact with multiple file servers.

FIPS support

WebSphere MQ Managed File Transfer supports the use of FIPS-compliant cryptography modules in client connections from agents, commands, and the IBM WebSphere MQ Explorer to queue managers. All SSL connections to the queue manager use the TLS protocol only. Support is provided for JKS and PKCS#12 keystore types.

Specify whether you want to enable FIPS support for an agent, a coordination queue manager, or a command queue manager as follows:

- If you want to enable FIPS for a specific agent, set the appropriate agentSsl properties in the `agent.properties` file for that agent. For more information, see "SSL properties" on page 598.
- If you want to enable FIPS for a specific coordination queue manager, set the appropriate coordinationSsl properties in the `coordination.properties` file for that coordination queue manager. For more information, see "SSL properties" on page 598.
- If you want to enable FIPS for a specific command queue manager, set the appropriate connectionSsl properties in the `command.properties` file for that command queue manager. For more information, see "SSL properties" on page 598.

FIPS is not supported on connections to or from a protocol bridge to an SFTP server, or a Connect:Direct bridge.

For more information about IBM WebSphere MQ and FIPS and the configuration steps required, see Federal Information Processing Standards (FIPS).

If you want to use FIPS, the CipherSuite must be FIPS-compliant or the connection fails. For more information about the CipherSpecs supported by IBM WebSphere MQ, see SSL CipherSpecs and CipherSuites in WebSphere MQ classes for Java.

FIPS support to and from FTPS servers

You can enable FIPS support for connections between protocol bridge agents and FTPS servers by setting the agentSslFipsRequired property to true. For more information, see SSL properties.

Database tables used by the logger

When you have installed and configured the logger, the following database tables are created:

AUTH_EVENT

An event related to authority checking, typically the rejection of a request due to insufficient privileges.

- **ID:** Row ID.
- **ACTION:** The type of action that took place.
- **COMMAND_ID:** The WebSphere MQ message ID of the original message that requested the event. In the case of a transfer request, this will also be the transfer ID.
- **TIME:** The time at which the event occurred.

- **ORIGINATOR_MQ_USER:** The user ID contained in the WebSphere MQ message, against which the authority check was performed.
- **AUTHORITY:** The authority that was required for the requested action.
- **ORIGINAL_XML_REQUEST:** The payload of the command message, indicating what action was refused.
- **RESULTCODE:** The numeric code identifying the result.
- **RESULT_TEXT:** A message explaining the result of the authority event.

CALL

The remote running of an operating system command, or Ant script managed by WebSphere MQ Managed File Transfer. Calls can be embedded in transfers, or referred to by call_request rows.

A CALL (that is, a row in this table) can either be part of a normal transfer (in which case TRANSFER_CALLS is used to link it to the relevant entry in TRANSFERS) or it can be a stand-alone managed call on its own (available only from Ant or by directly inserting messages). In the latter case, the CALL_REQUEST table is used instead of the TRANSFERS table; an equivalent to TRANSFER_CALLS is not needed because there can be only one call per call request.

- **ID:** Row ID.
- **COMMAND:** The command that was run. This field does not include any arguments passed to the command or the path where the command is located.
- **TYPE:** The type of command, such as Ant or JCL.
- **RETRIES:** The number of retries that were requested.
- **RETRY_WAIT:** The interval to wait between retries as originally requested, in seconds.
- **SUCCESS_RC:** The return code that indicates a successful completion of the command. If any other code is received, the run is reported to have failed.
- **EXECUTED_COMMAND:** The full name of the command that was run, including path.
- **CAPPED_RETRIES:** The number of retries available; this number might be less than requested if the retry limit of the agent is lower than the number of retries requested.
- **CAPPED_RETRY_WAIT:** The interval between retries that is used; this number might be less than requested if the configured limit of the agent is lower than the retry wait requested.
- **OUTCOME:** Whether the call was successful overall. If there were multiple tries the outcome of each one is recorded separately in the CALL_RESULT table.
- **PRIORITY:** The application priority that is given to the background application when the type for this call is os4690background.
- **MESSAGE:** The initial status message for the background application when the type for this call is os4690background. Contains NULL if the type is not os4690background.

CALL_ARGUMENT

An argument or parameter supplied to a command that is called.
- **ID:** Row ID.
- **CALL_ID:** The call that the argument is associated with.
- **KEY:** Where the argument is of a key-value-pair kind, the key, or name.
- **TYPE:** The type of the argument: some are position parameters to operating system commands and others are named properties used with Ant.
- **VALUE:** The value of the argument.

CALL_REQUEST

The vehicle for a command call that is not part of a file transfer. You can submit ManagedCall messages using Ant and using direct XML injection.

- **ID:** The hexadecimal ID of the managed call request.
- **CALL_ID:** The database ID of the row in the CALL table describing this call.
- **ACTION_TIME:** The time that the action occurred.
- **AGENT:** The agent that the command is run on.
- **AGENT_QM:** The queue manager used by the agent that the command is run on.
- **ARCHITECTURE:** The machine architecture of the system that the agent runs on.
- **OS_NAME:** The name of the operating system that the agent is running on.
- **OS_VERSION:** The version of the operating system.
- **ORIGINATOR_HOST:** The host name of the machine that the call request was submitted from.
- **ORIGINATOR_USER:** The name of the user who submitted the call request, as reported in the request XML.
- **ORIGINATOR_MQ_USER:** The name of the user who submitted the call request, as contained in the WebSphere MQ message descriptor of the request.
- **JOB_NAME:** A user-specified job name.
- **RESULTCODE:** The overall result code for the call.
- **RESULTTEXT:** The overall result message for the call.

CALL_RESULT

The detailed result of calling a command. A call can have multiple results if retries were enabled.

- **ID:** Row ID.
- **CALL_ID:** The database ID of the row in the CALL table that this result applies to.
- **SEQUENCE:** Which attempt this result applies to, where there have been multiple attempts.
- **OUTCOME:** The outcome (for example, success or failure) of the command.
- **RETURN_CODE:** The command return code.
- **TIME:** The time that the command completed.
- **STDOUT:** The standard output stream from the command, if it was started.
- **STDERR:** The standard error stream from the command, if it was started.
- **ERROR:** If the command could not be started, an error message produced by WebSphere MQ Managed File Transfer explaining the problem.

FILE_SPACE_ENTRY

Each row represents a file that has been sent to the named file space.

- **ID:** The ID of the file space entry.
- **FILE_SPACE_NAME:** The name of the file space. This is the name of the user that the file space belongs to.
- **TRANSFER_ITEM_ID:** The ID of the transfer item that this row relates to.
- **ALIAS:** The alias name for this file space entry. Typically this alias name is the name of the source file for the transfer.
- **DELETED:** The time when the file was deleted from the file space. If the file has not been deleted the value is `null`.

METADATA

Metadata associated with a transfer.

- **ID:** Row ID.
- **TRANSFER_EVENT_ID:** The transfer_event row that this metadata is associated with, if it relates to a transfer. This field is null if the metadata is associated with a stand-alone managed call.
- **STANDALONE_CALL_ID:** If the metadata is associated with a stand-alone managed call, the ID of the managed call request concerned.
- **KEY:** The name of the metadata item.
- **VALUE:** The value of the metadata item.

MONITOR

Resource monitors that trigger WebSphere MQ Managed File Transfer operations based on external conditions.

- **AGENT:** The agent that the monitor runs on.
- **ID:** The hexadecimal ID of the monitor.
- **NAME:** The name of the monitor.
- **QMGR:** The queue manager of the agent where the monitor runs.

MONITOR_ACTION

Each row represents an action (for example, creation and triggering) occurring in respect of a monitor

- **ID:** Row ID.
- **ACTION:** The type of action that took place.
- **JOB_NAME:** The name of the submitted job, where applicable.
- **MONITOR:** The monitor that this action occurred on. Might be null if the action failed because it was requested for a monitor that does not exist.
- **ORIGINAL_XML_REQUEST:** If this action was a *create* or *triggerSatisfied* action, the XML request that is started when the monitor is triggered.
- **ORIGINATOR_MQ_USER:** The user ID contained in the WebSphere MQ message that initiated the action
- **ORIGINATOR_USER:** The user name that submitted the request to perform the action.
- **ORIGINATOR_HOST:** The machine from which the user submitted the request to perform the action.
- **TIME:** The time that the action occurred.
- **UPDATED_XML_REQUEST:** If the action is *triggerSatisfied*, the XML request that was started. This request might vary from the XML request that was originally made because of variable substitution.

MONITOR_EXIT_RESULT

The result of running a resource monitor exit.

- **ID:** Row ID.
- **ACTION_ID:** The monitor action that the result is associated with.
- **EXIT_NAME:** The name of the exit that produced this result.
- **RESULTCODE:** The numeric result code from the exit.
- **RESULTTEXT:** The text output from the exit, if provided.

MONITOR_METADATA

Items of metadata associated with a resource monitor.
- **ID:** Row ID.
- **ACTION_ID:** The monitor_action that the metadata is associated with.
- **KEY:** The name of the metadata item.
- **PHASE:** Whether this metadata item represents the data that was originally submitted or the updated version after variable substitution.
- **VALUE:** The value of the metadata item.

SCHEDULE

A transfer schedule registered with an agent.
- **AGENT:** The name of the agent that has this schedule.
- **CREATION_DATE:** The point in time that this schedule was created.
- **ID:** The unique database (not agent) ID for the schedule.
- **ID_ON_AGENT:** The ID that the agent uses for the database ID. This ID is not unique across agents and might not even be unique in an agent if the persistent state of the agent is reset.
- **LATEST_ACTION:** The most recent action that modified the state of this schedule.

SCHEDULE_ACTION

When an event occurs that modifies the schedule state, an action is recorded.
- **ACTION_TYPE:** The action that occurred.
- **ID:** Row ID
- **ORIGINATOR_HOST:** The machine that the request that caused the change was submitted from.
- **ORIGINATOR_USER:** The user whose name the request that caused the change was submitted in.
- **SCHEDULE_ID:** The schedule that this action applies to.
- **SPEC_AFTERWARDS:** The schedule_spec that represents the state of this schedule after the action occurred.
- **STATUS_CODE:** A numeric return code describing the outcome of the action
- **STATUS_TEXT:** A text description of the outcome of the action. Typically null if the action succeeded.
- **TIME:** The point in time that the action occurred

SCHEDULE_SPEC

The details of an individual scheduled transfer.
- **ID:** Row ID.
- **DESTINATION_AGENT:** The agent that the files are transferred to.
- **DESTINATION_QM:** The queue manager used by the destination agent.
- **REPEAT_COUNT:** How many times to repeat if the schedule repeats and is bound by the number of occurrences rather than an end time.
- **REPEAT_FREQUENCY:** How many repeat_intervals there are between scheduled transfers.
- **REPEAT_INTERVAL:** If the transfer repeats, what interval to repeat at (for example, minutes or weeks).
- **SOURCE_AGENT:** The agent that the files are transferred from.
- **SOURCE_QM:** The queue manager used by the source agent.
- **START_TIME:** The time that the first transfer in the schedule will take place.

- **START_TIMEBASE:** The time base for the times associated with the transfer. For example, whether to operate from the time zone of the agent or the time zone of the administrator.
- **START_TIMEZONE:** The time zone that the time base corresponds to and which will be used in operating the schedule.

SCHEDULE_ITEM

Each file (or pattern to match at transfer time) is represented by a schedule_item.

- **ID:** Row ID.
- **CHECKSUM_METHOD:** How the checksum for the file is calculated
- **DESTINATION_EXISTS_ACTION:** What action the destination agent takes if the file already exists at the destination.
- **DESTINATION_FILENAME:** The file or directory that the files are transferred into.
- **DESTINATION_TYPE:** Whether the destination_filename column refers to a file or directory.
- **FILE_MODE:** The mode (for example, *text* or *binary*) that the file is transferred in.
- **RECURSIVE:** When the agent creates the transfer according to the schedule, whether the agent recurses (*Y*) or not (*N*) the source directory.
- **SCHEDULE_SPEC_ID:** The schedule_spec that this item is associated with.
- **SOURCE_DISPOSITION:** What action to perform on source files after the transfer completes.
- **SOURCE_FILENAME:** The source file, directory name, or pattern.

TRANSFER

A single transfer of one or more files.

- **TRANSFER_ID:** The hexadecimal ID for the transfer.
- **JOB_NAME:** A user-specified job name for the transfer.
- **SCHEDULE_ID:** If this transfer is the result of a schedule, the database row ID of the schedule concerned.
- **START_ID:** The row ID of the transfer_event that represents the start of the transfer.
- **COMPLETE_ID:** The row ID of the transfer_event that represents the end of the transfer.
- **RESULTCODE:** The overall result code for the transfer. The possible values for this column are listed in the following topic: "Return codes for WebSphere MQ Managed File Transfer" on page 373. These codes apply to the transfer as a whole; see TRANSFER_ITEM.RESULTCODE for the status of each individual item.
- **RESULTTEXT:** The overall result text for the transfer, if any.
- **STATUS:** The status of a transfer. The possible values for this column are started, success, partial success, failure, and cancelled.
- **RELATED_TRANSFER_ID:** The hexadecimal ID of a previous transfer that is related to this transfer. For example, if the transfer is a file download using the Web Gateway, this field will refer to the transfer that uploaded the file.

TRANSFER_CALLS

Links runnable command calls to transfers

- **ID:** Row ID.
- **POST_DESTINATION_CALL:** The call made at the destination after the transfer is complete.
- **POST_SOURCE_CALL:** The call made at the source agent after the transfer is complete.
- **PRE_DESTINATION_CALL:** The call made at the destination agent before the transfer starts.
- **PRE_SOURCE_CALL:** The call made at the source agent before the transfer starts.

- **TRANSFER_ID:** The transfer that the calls in this row are associated with.

TRANSFER_CD_NODE

Information about Connect:Direct nodes that are used in a transfer.
- **PNODE:** The primary node in the transfer.
- **SNODE:** The secondary node in the transfer.
- **BRIDGE_IS_PNODE:** Character indicating which node is the node that is part of the Connect:Direct bridge. If this value is Y, the primary node is the bridge node. If this value is N, the secondary node is the bridge node.
- **ID:** The ID of this row.

TRANSFER_CORRELATOR

Each row contains a correlation string and a number associated with a transfer item.
- **CORRELATION_BOOLEAN:** A boolean correlation value. Represented by a single character of Y for true and N for false.
- **CORRELATION_STRING:** A string correlation value.
- **CORRELATION_NUMBER:** A numeric correlation value.
- **ID:** The ID of this row.

TRANSFER_EVENT

An event (start or end) related to a transfer.
- **ID:** Row ID.
- **ACTION_TIME:** The time that the transfer action took place.
- **SOURCE_AGENT:** The name of the agent that the files are transferred from.
- **SOURCE_AGENT_TYPE:** The type of agent that the files are transferred from. The following values are possible: 1 = STANDARD, 2 = BRIDGE, 3 = WEB_GATEWAY, 4 = EMBEDDED, 5 = CD_BRIDGE, 6 = SFG.
- **SOURCE_QM:** The queue manager used by the source agent.
- **SOURCE_ARCHITECTURE:** The machine architecture of the system hosting the source agent.
- **SOURCE_OS_NAME:** The operating system of the source agent machine.
- **SOURCE_OS_VERSION:** The version of operating system of the source agent machine.
- **SOURCE_BRIDGE_URL:** If the source agent is a protocol bridge agent, the URL of the data source to which it forms a bridge.
- **SOURCE_WEB_GATEWAY:** The name of the Web Gateway that the files are transferred from.
- **SOURCE_CD_NODE_ID:** The Connect:Direct node that is the source of the transfer.
- **DESTINATION_AGENT:** The name of the agent that the files are transferred to.
- **DESTINATION_AGENT_TYPE:** The type of agent that the files are transferred to. The following values are possible: 1 = STANDARD, 2 = BRIDGE, 3 = WEB_GATEWAY, 4 = EMBEDDED, 5 = CD_BRIDGE, 6 = SFG.
- **DESTINATION_QM:** The queue manager used by the destination agent.
- **DESTINATION_BRIDGE_URL:** If the destination agent is a bridge agent, the URL of the data source to which it forms a bridge.
- **DESTINATION_WEB_GATEWAY:** The name of the Web Gateway that the files are transferred to.
- **DESTINATION_CD_NODE_ID:** The Connect:Direct node that is the destination of the transfer.
- **ORIGINATOR_HOST:** The host name of the machine that the transfer request was submitted from.

- **ORIGINATOR_USER:** The name of the user who submitted the transfer request, as reported by the `fteCreateTransfer` command.
- **ORIGINATOR_MQ_USER:** The name of the user who submitted the transfer request, as contained in the WebSphere MQ message descriptor of the request.
- **ORIGINATOR_WEB_USER:** The name of the Web Gateway user, which is configured in your application server environment, who submitted the request.
- **TRANSFERSET_TIME:** The time that the transfer set was created.
- **TRANSFERSET_SIZE:** The number of items being transferred.
- **TRIGGER_LOG:** For transfer definitions involving a trigger, whether to log trigger evaluations that did not result in a transfer.

TRANSFER_EXIT

Each row represents a transfer exit which was executed as part of a file transfer.
- **ID:** Row ID.
- **EXIT_NAME:** The name of the exit.
- **TRANSFER_ID:** The ID of the completed or canceled transfer that this exit applies to.
- **TYPE:** The type of exit. This can be one of the following values: *SourceStart*, *SourceEnd*, *DestinationStart* or *DestinationEnd*.
- **STATUS:** The value that the exit returned. This can be *cancel* or *proceed*.
- **SUPPLEMENT:** An optional message explaining the status of the exit.

TRANSFER_ITEM

Each row represents a file that is sent as part of the transfer.
- **DESTINATION_CHECKSUM_METHOD:** The algorithm used to calculate a checksum of the destination file. Might be null if no checksum was calculated because the transfer did not complete successfully.
- **DESTINATION_CHECKSUM_VALUE:** The checksum value of the destination file. The value might be null if checksumming was disabled.
- **DESTINATION_ENCODING:** The character encoding used on the destination file, if the destination file is transferred as text.
- **DESTINATION_EXISTS_ACTION:** The action to perform if the file exists at the destination.
- **DESTINATION_FILE_SIZE:** The size of the file name to use at the destination.
- **DESTINATION_FILENAME:** The file name to use at the destination.
- **DESTINATION_LINEEND:** The line-end format used in the destination file, if the destination file is transferred as text.
- **DESTINATION_MESSAGE_QUEUE_NAME:** The destination queue for the messages that are produced from the source file during a file to message transfer.
- **DESTINATION_MESSAGE_GROUP_ID:** If more than one message is produced, the group ID used for the messages that are produced from the source file during a file to message transfer.
- **DESTINATION_MESSAGE_MESSAGE_ID:** If only one message is produced, The message ID of the message that is produced from the source file during a file to message transfer.
- **DESTINATION_MESSAGE_COUNT:** The number of messages that the source file was split into during a file to message transfer.
- **DESTINATION_MESSAGE_LENGTH:** The length of the message that is produced from the source file during a file to message transfer, in bytes. This value is only set if you specify a length for the output messages, for example by using the `-qs` option of the `fteCreateTransfer` command. If you specify `-qs 20K` and the size of your source file is 50 KB, the resulting three messages are 20 KB, 20 KB, and 10 KB in size. In this case the value of DESTINATION_MESSAGE_LENGTH is set to 20480.

- **DESTINATION_CORRELATOR_ID:** The ID of the correlator information for the destination.
- **FILE_MODE:** The file transfer mode, for example *text* or *binary*.
- **ID:** Row ID
- **RESULTCODE:** A numeric code indicating the outcome of the transfer of this item. The possible values for this column are listed in the following topic: "Return codes for files in a transfer" on page 378. These codes apply to the individual items in the transfer; see TRANSFER.RESULTCODE for the result of the transfer as a whole.
- **RESULT_TEXT:** A textual explanation of the result of the transfer. Typically null if the transfer was successful.
- **SOURCE_CHECKSUM_METHOD:** The algorithm used to calculate a checksum of the source file.
- **SOURCE_CHECKSUM_VALUE:** The checksum value of the source file. The value might be null if checksumming was disabled.
- **SOURCE_DISPOSITION:** The action to perform on the source file when the transfer is complete.
- **SOURCE_ENCODING:** The character encoding used on the source file, if the source file is transferred as text.
- **SOURCE_FILE_SIZE:** The size of the file name to use at the source.
- **SOURCE_FILENAME:** The source file name.
- **SOURCE_LINEEND:** The line-end format used in the source file, if the source file is transferred as text.
- **SOURCE_MESSAGE_QUEUE_NAME:** The source queue for the messages that are included in the destination file for a message to file transfer.
- **SOURCE_MESSAGE_GROUP_ID:** The group ID of the messages that are included in the destination file for a message to file transfer.
- **SOURCE_MESSAGE_COUNT:** The number of messages that are included in the destination file for a message to file transfer.
- **SOURCE_CORRELATOR_ID:** The ID of the correlator information for the source.
- **TRANSFER_ID:** The transfer that this item is part of.

TRANSFER_ITEM_ATTRIBUTES

Each row represents an attribute name-value pair associated with a row in the TRANSFER_ITEM table.
- **ID:** Row ID.
- **TRANSFER_ITEM ID:** The TRANSFER_ITEM row associated with this attribute name-value pair.
- **ATTRIBUTE_NAME:** The name of the attribute. For example, DIST
- **ATTRIBUTE_VALUE:** The value of the attribute, if any. For example, MIRRORED,CLOSE or 3

For more information about distribution attributes for WebSphere MQ Managed File Transfer on IBM 4690, see "File distribution attributes" on page 87.

TRANSFER_STATS

A set of statistics generated at the end of a transfer.
- **ID:** Row ID.
- **TRANSFER_ID:** The transfer to which the statistics refer.
- **START_TIME:** The time at which the transfer started. In a system that is busy or has intermittent connectivity, this time might be later than the time reported in the Started message, as that time represents the point at which initial processing began rather than the point at which the successful transfer of data began.
- **RETRY_COUNT:** The number of times that the transfer had to be retried because of load or availability issues.

- **FILE_FAILURES:** The number of files that failed to be transferred.
- **FILE_WARNINGS:** The number of files that had warnings reported for them when they were transferred.

TRIGGER_CONDITION

One condition in a basic WebSphere MQ Managed File Transfer conditional transfer. For example, "file `example.file` exists".

- **ID:** Row ID.
- **TRANSFER_EVENT_ID:** The transfer event that the trigger is related to.
- **CONDITION_TYPE:** The type of check used in the trigger. For example, the existence of a file or the size of a file.
- **COMPARISON:** The specific comparison to make. For example "greater than or equal to".
- **VALUE:** The value to compare against.
- **FILENAME:** The file name to examine.

Related concepts:

"Configuring a WebSphere MQ Managed File Transfer logger" on page 133

Related reference:

"fteStartLogger (start a logger)" on page 531
The **fteStartLogger** command starts a WebSphere MQ Managed File Transfer logging application.

"fteModifyLogger (run a WebSphere MQ Managed File Transfer logging application as a Windows service)" on page 509
Use the **fteModifyLogger** command to modify a logger so that it can be run as a Windows service. You can use this command only on Windows platforms, and you must first stop the logger by using the **fteStopLogger** command.

"fteStopLogger (stop a logger)" on page 536
The **fteStopLogger** command stops a logger.

Authorities for the logger

The operating system user who runs the logger requires certain WebSphere MQ authorities on the logger queues and the SYSTEM.FTE topic.

The operating system user who runs the logger requires the following WebSphere MQ authorities:

- CONNECT and INQUIRE on the coordination queue manager.
- SUBSCRIBE permission on the SYSTEM.FTE topic.
- PUT permission on the SYSTEM.FTE.LOG.RJCT.*logger_name* queue.
- GET permission on the SYSTEM.FTE.LOG.CMD.*logger_name* queue.

Related reference:

"Group authorities for resources specific to WebSphere MQ Managed File Transfer" on page 405
Instead of granting authority to individual users for all of the various objects that might be involved, configure two security groups for the purposes of administering WebSphere MQ Managed File Transfer access control: FTEUSER and FTEAGENT. It is the responsibility of the WebSphere MQ administrator to create and populate these groups. The administrator can choose to extend or modify the proposed configuration described here.

"User authorities on WebSphere MQ Managed File Transfer actions" on page 409
In addition to using groups to manage access to resources, you can enable an additional level of security to restrict the agent actions that a user can take. Grant authorities on an agent authority queue to a user to give the user permission to perform specific agent actions.

WebSphere MQ message properties set on messages written to destination queues

When transferring from file to message, WebSphere MQ Managed File Transfer can set WebSphere MQ message properties on the first message written to the destination queue. Additional WebSphere MQ message properties are set when a file to message transfer has failed.

WebSphere MQ message properties allow an application to select messages to process, or to retrieve information about a message without accessing MQ Message Descriptor (MQMD) or MQRFH2 headers. See Message properties.

Standard properties

You can use the **-qmp** parameter on the **fteCreateTransfer** command or the **fteCreateTemplate** command to specify whether WebSphere MQ message properties are set on the first message written to the destination queue by the transfer. For an example of how to use this parameter, see the topic "Example: Setting WebSphere MQ message properties on a file-to-message transfer" on page 238

The WebSphere MQ message properties contain transfer metadata. The message property names are prefixed with **usr.WMQFTE**. The **usr.** prefix makes these message properties available to JMS applications.

usr.WMQFTETransferId
> The unique hexadecimal transfer ID.

usr.WMQFTETransferMode
> The type of file transfer: binary mode or text mode.

usr.WMQFTESourceAgent
> The name of the source agent.

usr.WMQFTEDestinationAgent
> The name of the destination agent.

usr.WMQFTEFileName
> The name of the source file.

usr.WMQFTEFileSize
> The size of the source file in bytes.

usr.WMQFTEFileLastModified
> The last modified time of the source file. This value is in units of milliseconds, measured from 00:00:00 UTC, January 1, 1970.

usr.WMQFTEFileIndex
> The index of the current file in the list of files that are being transferred. The first file in the list has index 0.

usr.WMQFTEMqmdUser
> The MQMD user ID of the user that submitted the transfer request.

Failure properties

When a file to message transfer fails after the destination agent has written at least one message to the destination queue, WebSphere MQ Managed File Transfer writes a blank message to the destination queue. If the **-qmp** parameter is set to true, this blank message has two WebSphere MQ message properties set. For an example of a file to message transfer failure, see "Failure of a file to message transfer" on page 241.

When a file to message transfer fails completely, WebSphere MQ Managed File Transfer writes a blank message to the destination queue. If the **-qmp** parameter is set to true, and the length of the message data is greater than the maxInputOutputMessageLength value, the following error message is displayed at the command line.

```
Name WMQFTEResultCode
Value 40
Name WMQFTESupplement
Value BFGTR0072E: The transfer failed to complete due to the exception BFGIO0205E:The message data length 1290843 being wri
to the output queue "M2F@q2" is greater than the maximum allowed 1048576.
```

The WebSphere MQ message properties contain information about the failure. As with the standard message properties, the message property names are prefixed with **usr.WMQFTE** and are available to JMS applications.

usr.WMQFTEReturnCode
> The return code of the transfer. For a list of possible values for this return code, see the topic "Return codes for WebSphere MQ Managed File Transfer" on page 373.

usr.WMQFTESupplement
> A supplementary message describing in more detail why the transfer failed.

User-defined properties

Metadata specified using the **-md** parameter with the **fteCreateTransfer** command can be set as WebSphere MQ message properties. If the **-qmp** parameter is set to true, any metadata specified by the user will be added to the message header of the first message.

The metadata name is prefixed by **usr.**. For example, if the metadata is department=accounts, the WebSphere MQ message header is set to usr.department=accounts.

You cannot use metadata to specify headers that begin with usr.WMQFTE or usr.com.ibm.wmqfte. If you specify metadata with a name beginning with WMQFTE or com.ibm.wmqfte this metadata is not used in the message properties and is ignored.

Related concepts:

"Transfer data from files to messages" on page 233
You can use the file-to-message feature of WebSphere MQ Managed File Transfer to transfer data from a file to a single message, or multiple messages, on a WebSphere MQ queue.

Related tasks:

"Example: Setting WebSphere MQ message properties on a file-to-message transfer" on page 238
You can use the **-qmp** parameter on the **fteCreateTransfer** command to specify whether WebSphere MQ message properties are set on the first message written to the destination queue by the transfer. WebSphere MQ message properties allow an application to select messages to process, or to retrieve information about a message without accessing WebSphere MQ Message Descriptor (MQMD) or MQRFH2 headers.

Related reference:

"WebSphere MQ message properties read from messages on source queues" on page 703
The agent reading messages from a source queue in a message to file transfer reads the WebSphere MQ message properties from the message. The value of these properties can be used to determine the behavior of a transfer.

"Return codes for WebSphere MQ Managed File Transfer" on page 373
WebSphere MQ Managed File Transfer commands, Ant tasks, and log messages provide return codes to indicate whether functions have successfully completed.

"Failure of a file to message transfer" on page 241
If a file-to- message transfer fails after the agent has started writing file data to the destination queue, the agent writes a message to the queue to indicate to an application consuming the messages that a failure

has occurred.

"**fteCreateTransfer** (create new file transfer)" on page 461
The **fteCreateTransfer** command creates and starts a new file transfer from the command line. This command can start a file transfer immediately, schedule a file transfer for a future time and date, repeat a scheduled transfer one or more times, and trigger a file transfer based on certain conditions.

WebSphere MQ message properties read from messages on source queues

The agent reading messages from a source queue in a message to file transfer reads the WebSphere MQ message properties from the message. The value of these properties can be used to determine the behavior of a transfer.

Headers used to cancel message to file transfers

Set the following WebSphere MQ message properties on the last message in a group to cancel the message to file transfer of that group:

usr.UserReturnCode
> Required. The return code of the transfer. Set this header as a non-zero value to indicate that the transfer is to be canceled.

usr.UserSupplement
> Optional. Text describing why the transfer was canceled.

If the source agent of a message to file transfer reads a message from the source queue that has the **usr.UserReturnCode** message property set to a non-zero value, it stops reading messages from the queue and reports that the transfer failed in the transfer log XML. The transfer log XML contains the return code and supplementary text that is set in the message headers. If the destination agent has already written data to a temporary file this file is deleted from the destination.

Headers used by variable substitution

The value of any WebSphere MQ message property in the first message to be read from the monitored queue can be substituted into the task XML definition. User-defined message properties are prefixed with `usr.`, but do not include this prefix in the variable name. Variable names must be preceded by a dollar sign ($) character and enclosed in braces ({}). For example, ${destFileName} is replaced with the value of the `usr.destFileName` message property of the first message to be read from the source queue.

For example, the user or program putting messages to a monitored queue can set WebSphere MQ message properties on the first message in a group specifying which agent is to be used as the destination of the file transfer and what file name to transfer the data to.

For more information, see "Monitoring a queue and using variable substitution" on page 228.

Guidance for setting WebSphere MQ attributes and WebSphere MQ Managed File Transfer properties associated with message size

You can change WebSphere MQ attributes and WebSphere MQ Managed File Transfer properties to affect the behavior of WebSphere MQ Managed File Transfer when reading or writing messages of various sizes.

If the size of messages being read from a source queue or written to a destination queue exceeds 1048576 bytes (1 MB), you must increase the value of the WebSphere MQ Managed File Transfer agent property `maxInputOutputMessageLength` to a value that is greater than or equal to the maximum message size to be read or written.

If the messages on the source queue are greater than 1048576 bytes, you must set the **maxInputOutputMessageLength** property on the source agent. If the messages on the destination queue are greater than 1048576 bytes you must set the **maxInputOutputMessageLength** property on the destination agent. For more information about the **maxInputOutputMessageLength** property, see Advanced agent properties.

- If the queue that the agent is writing to or reading from is local to the agent queue manager, you might have to change the WebSphere MQ queue manager, queue, and channel **MAXMSGL** attributes.

  Ensure that the value of the maximum message size of the source or destination queue is greater than or equal to the value of the **maxInputOutputMessageLength** agent property.

  Ensure that the value of each of the following WebSphere MQ attributes, in bytes:

  – The maximum message size of the agent queue manager

  – The maximum message size of the SYSTEM.FTE.STATE.*<agent_name>* queue

  – The client channel maximum message size, if your agent connects to the queue manager in client mode

  is greater than or equal to the result of the following calculation:

  For a file-to-message transfer:
  > The value of **maxInputOutputMessageLength**

  For a message-to-file transfer:
  > The value of 3 * (**maxInputOutputMessageLength**) + 1048576

  (This calculation is derived from the fact that three checkpoints can be stored in a state message and each checkpoint might have to buffer up to the maximum size of a message amount of data.)

- If the queue that the agent is writing to is a remote queue, you might have to change the WebSphere MQ queue manager, queue, and channel **MAXMSGL** attributes.

  Ensure that the value of each of the following WebSphere MQ attributes is greater than or equal to the value of the **maxInputOutputMessageLength** agent property:

  – The maximum message size of the remote queue manager transmission queue on the agent queue manager

  – The maximum message size of the channel from the agent queue manager to the remote queue manager

  – The maximum message size of the destination queue on the remote queue manager

  – The maximum message size of the remote queue manager

  Ensure that the value of each of the following WebSphere MQ attributes, in bytes:

  – The maximum message size of the agent queue manager

  – The maximum message size of the SYSTEM.FTE.STATE.*<agent_name>* queue

  – The client channel maximum message size, if your agent connects to the queue manager in client mode

  is greater than or equal to the result of the following calculation:

  For a file-to-message transfer:
  > The value of **maxInputOutputMessageLength**

  For a message-to-file transfer:
  > The value of 3 * (**maxInputOutputMessageLength**) + 1048576

  (This calculation is derived from the fact that three checkpoints can be stored in a state message and each checkpoint might have to buffer up to the maximum size of a message amount of data.)

If you exceed the value of one of these properties, the agent stops with the following error in the agent event log:

```
BFGUT0002E: An internal error has occurred. Product failure data was captured in file
"FFDC.FTE.20100928170828514.8172766022149157013.log".
BFGSS0025E: An internal error has occurred. The exception is: cc=2 rc=2010 op=put - MQPUT to SYSTEM.FTE.STATE.<agent_name>
BFGAG0061E: The agent ended abnormally
```

The following WebSphere MQ reason codes might be included in this message in the agent event log:

- `rc=2010` This reason code maps to MQRC_DATA_LENGTH_ERROR and indicates that the value of the client channel maximum message size was exceeded. To resolve this problem ensure that the client channel maximum message size of the agent queue manager is greater than or equal to the result of the following calculation:

  `3 * (maxInputOutputMessageLength) + 1048576`

- `rc=2030` This reason code maps to MQRC_MSG_TOO_BIG_FOR_Q and indicates that the value of the maximum message size of the SYSTEM.FTE.STATE.<agent_name> queue was exceeded. To resolve this problem ensure that the maximum message size of the SYSTEM.FTE.STATE.<agent_name> queue is greater than or equal to the result of the following calculation:

  `3 * (maxInputOutputMessageLength) + 1048576`

- `rc=2031` This reason code maps to MQRC_MSG_TOO_BIG_FOR_Q_MGR and indicates that the value of the maximum message size of the agent queue manager was exceeded. To resolve this problem ensure that the maximum message size of the agent queue manager is greater than or equal to the result of the following calculation:

  `3 * (maxInputOutputMessageLength) + 1048576`

If you are transferring many small messages

If the average size of the messages that the agent is reading from or writing to a queue is less than 1310 bytes and the agent is reading or writing more than 10000 messages, you must increase the maximum number of uncommitted messages property on the queue manager or reduce the amount of data in a checkpoint interval.

When the agent is reading messages from or writing messages to a queue the corresponding **GET**s or **PUT**s are grouped together into transactions. The number of **GET**s or **PUT**s in a transaction is determined by the number required to process all of the data within a checkpoint interval. The approximate amount of the data in a checkpoint interval is determined from agent properties using the following calculation:

```
Checkpoint interval data size (in bytes) = agentCheckpointInterval * agentFrameSize *
                                           agentWindowSize * agentChunkSize.
```

The default checkpoint data size is 1 * 5 * 10 * 262144 bytes = 13107200 bytes (12.5MB). The maximum number of uncommitted messages in a transaction that a queue manager supports is controlled by the **MaxUncommittedMsgs** queue manager attribute. The default value of this attribute is 10000 messages. If the average message size is less than approximately 1310 bytes the default maximum number of uncommitted messages is exceeded if there are more than 10000 messages to be written.

If you exceed the **MaxUncommittedMsgs** limit, the agent stops with the following error in the agent event log:

```
BFGSS0024E: The agent has received a reason code of '2024' from the message queue interface (MQI).
The agent cannot continue processing and will now end.
BFGAG0139I: The agent has suspended its current transfers and is now stopping.
```

The reason code 2024 maps to: MQRC_SYNCPOINT_LIMIT_REACHED.

To resolve this problem perform one of the following actions

- Increase the value of the **MaxUncommittedMsgs** queue manager property of the queue manager that the agent reading from or writing to a queue connects to. See MaxUncommittedMsgs (MQLONG).
- Reduce the amount of data in a checkpoint interval. To do this, decrease the value of one or more of the following agent properties:
  - agentCheckpointInterval

– agentFrameSize
– agentWindowSize
– agentChunkSize

For information about these agent properties, see Advanced agent properties.

If you are writing messages to a queue persistently

If you are transferring to a queue and writing the messages to the queue persistently, you might have to increase the size of the queue manager log file space to be able to log all of the data in a checkpoint interval.

If you exceed the queue manager log file space, the agent stops with the following error in the agent event log:

```
BFGSS0024E: The agent has received a reason code of '2102' from the message queue interface (MQI).
The agent cannot continue processing and will now end.
BFGAG0062E: The agent has received MQI reason code '2102'.  The agent cannot continue processing and will now end.
BFGAG0061E: The agent ended abnormally
```

The reason code '2102' maps to: MQRC_RESOURCE_PROBLEM.

To resolve this problem increase the size of the destination agent queue manager log file space.

Guidance for specifying a wait time on a message-to-file transfer

When specifying a message-to-file transfer you can optionally specify a wait time on the transfer using the **-sqwt** parameter. The value of **-sqwt** is the amount of time that the source agent waits either for a message to appear on the source queue if the source queue is empty or becomes empty, or for a complete group to appear on the source queue if the **-sqgi** attribute is specified.

If the value of the **-sqwt** parameter is greater than or equal to the amount of time the destination agent waits for the transfer to be completed by the source agent, the transfer does not complete. The amount of time the destination agent waits for the transfer to complete is given by the following calculation:

```
transferAckTimeout * transferAckTimeoutRetries
```

The properties transferAckTimeout and transferAckTimeoutRetries are set in the destination agent agent.properties file. For more information about these agent properties, see "The agent.properties file" on page 548.

To prevent transfers from failing to complete, you must perform one of the following steps:
* Reduce the value of the **-sqwt** parameter so that it is less than the value of the destination agent transferAckTimeout property.

  Note: The default value of the transferAckTimeout property is 60,000 milliseconds. The value of the **-sqwt** parameter is given in seconds, set the value to 59 or less.
* Increase the value of the destination agent transferAckTimeout property so that it is greater than the value of the **-sqwt** parameter.

  Note: The value of the transferAckTimeout property is given in milliseconds. The value of the **-sqwt** parameter is given in seconds.

Related reference:

"**fteCreateTransfer** (create new file transfer)" on page 461
The **fteCreateTransfer** command creates and starts a new file transfer from the command line. This command can start a file transfer immediately, schedule a file transfer for a future time and date, repeat a scheduled transfer one or more times, and trigger a file transfer based on certain conditions.

"The agent.properties file" on page 548
Each agent has its own properties file, `agent.properties`, that must contain the information that an agent uses to connect to its queue manager. The `agent.properties` file can also contain properties that alter the behavior of the agent.

Available code pages

This reference topic lists all character encoding formats available for text file conversion on the various platforms supported by WebSphere MQ Managed File Transfer.

Common Encodings

These character encoding formats are available on all supported platforms. If your source file is encoded using one of the formats in this table, and you want to use another of the formats in this table to write the destination file, you can do so without any consideration of platform. You can use either the canonical name or any of the aliases to specify an encoding format.

Canonical Name	Aliases
windows-1256	ibm-1256, Cp1256
windows-1255	ibm-1255, Cp1255
windows-1254	Cp1254, ibm-1254
windows-1253	Cp1253, ibm-1253
windows-1252	ibm-1252, Cp1252
windows-1251	ibm-1251, Cp1251
windows-1250	Cp1250, ibm-1250
UTF-8	UTF_8, UTF8
UTF-16LE	X-UTF-16LE, UTF16LE, UTF_16LE, UnicodeLittleUnmarked
UTF-16BE	UTF16BE, UnicodeBigUnmarked, ISO-10646-UCS-2, UTF_16BE, X-UTF-16BE
US-ASCII	Cp367, iso-ir-6, ANSI_X3.4-1968, ANSI_X3.4-1986, default, ASCII, us, iso-646.irv:1983, csASCII, 646, ascii7, ISO646-US, ibm-367, ISO-646.irv:1991, direct
TIS-620	tis620, tis620.2533
IBM-1122	Cp1122, ibm1122
IBM-1006	Cp1006, ibm1006
IBM-037	ibm-37
GB18030	windows-54936, gb18030-2000, ibm-1392
EUC-TW	x-euc-tw, euctw, cns11643, euc_tw
EUC-KR	ibm-euckr, euc_kr, ksc_5601, ks_c_5601-1987, ksc5601_1987, euckr, ksc5601-1987, ibm-970, Cp970, 5601
EUC-JP	x-euc-jp, euc_jp, eucjp, x-eucjp, euc_jp_linux, euc-jp-linux
EUC-CN	x-euc-cn, ibm-euccn, euc_cn, euccn
Big5	big5-0, big5, Big5-HKSCS
IBM-1025	Cp1025, ibm1025
IBM-1026	ibm1026, Cp1026
IBM-1046	Cp1046, ibm1046
IBM-1097	Cp1097, ibm1097
IBM-1098	Cp1098, ibm1098

Canonical Name	Aliases
IBM-1112	ibm1112, Cp1112
IBM-1383	Cp1383, ibm1383
IBM-273	Cp273, ibm273
IBM-277	Cp277, ibm277
IBM-278	Cp278, ibm278
IBM-280	ibm280, Cp280
IBM-284	ibm284, Cp284
IBM-285	Cp285, ibm285
IBM-297	ibm297, Cp297
IBM-420	Cp420, ibm420
IBM-860	Cp860, ibm860
IBM-861	ibm861, Cp861
IBM-862	Cp862, ibm862
IBM-863	Cp863, ibm863
IBM-864	Cp864, ibm864
IBM-865	ibm865, Cp865
windows-1257	Cp1257, ibm-1257
windows-1258	Cp1258, ibm-1129, ibm-1258
windows-31j	ms_kanji, cswindows31j, MS932, windows-932
windows-874	MS874
windows-936	MS936, x-mswin-936, 936
windows-949	MS949, Cp1361, ibm-1361, ibm1361, ms1361, ksc5601-1992, x-windows-949
windows-950	MS950, x-windows-950
IBM-857	ibm857, Cp857, csibm857
IBM-856	Cp856, ibm856
IBM-855	Cp855, ibm855
IBM-852	cspcp852, ibm852, Cp852
IBM-850	Cp850, ibm850, cspc850multilingual
IBM-838	Cp838, ibm838
IBM-834	Cp834, ibm834
IBM-775	ibm775, Cp775
IBM-737	Cp737, ibm737
IBM-500	Cp500, ibm500
IBM-437	ibm437, Cp437, cspc8codepage437
IBM-424	ibm424, Cp424
IBM-1123	Cp1123, ibm1123
IBM-1124	Cp1124, ibm1124
IBM-1381	Cp1381, ibm1381
IBM-866	Cp866, ibm866
IBM-868	Cp868, ibm868
IBM-869	ibm869, Cp869
IBM-870	Cp870, ibm870
IBM-871	ibm871, Cp871
IBM-874	ibm874, Cp874
IBM-875	Cp875, ibm875
IBM-921	Cp921, ibm921

Canonical Name	Aliases
IBM-922	Cp922, ibm922
IBM-933	Cp933, ibm933
IBM-935	Cp935, ibm935
IBM-937	Cp937, ibm937
IBM-942	Cp942, ibm942
IBM-943	Cp943, ibm943
IBM-948	ibm948, Cp948
IBM-949	ibm949, Cp949
IBM-950	ibm950, Cp950
ISCII91	iscii
ISO-2022-CN	iso2022-cn-cns, iso2022cn-cns, iso-2022-cn-cns, iso2022cn, iso2022-cn
ISO-2022-CN-GB	iso2022-cn-gb, iso2022cn-gb
ISO-2022-JP	iso2022jp, jis, iso2022-jp, iso-2022-jp2, csiso2022jp2, csjisencoding, jis-encoding
ISO-2022-KR	csiso2022kr, iso2022-kr, iso2022kr
ISO-8859-1	iso8859_1, iso8859-1, ibm819, l1, csisolatin1, Cp819, iso-ir-100, iso-8859-1:1987, ibm-819, latin1, 8859-1
ISO-8859-13	iso8859-13, 8859-13, iso8859_13
ISO-8859-15	csisolatin9, iso8859-15, ibm923, latin9, ibm-923, l9, iso8859_15, iso8859_15_fdis, Cp923, latin0
ISO-8859-2	Cp912, ibm912, iso8859-2, iso-8859-2:1987, l2, iso8859_2, csisolatin2, latin2, ibm-912, 8859-2, iso-ir-101
ISO-8859-3	iso8859-3, Cp913, l3, iso8859_3, iso-ir-109, iso-8859-3:1988, latin3, ibm-913, 8859-3, csisolatin3
ISO-8859-4	Cp914, latin4, iso8859_4, l4, iso-8859-4:1988, ibm-914, iso8859-4, 8859-4, csisolatin4, iso-ir-110
ISO-8859-5	csisolatincyrillic, iso-ir-144, cyrillic, iso8859_5, iso-8859-5:1988, ibm-915, 8859-5, Cp915, ibm915, iso8859-5
ISO-8859-6	csisolatinarabic, Cp1089, iso-8859-6:1987, ecma-114, iso-ir-127, asmo-708, iso8859_6, 8859-6, ibm1089, arabic, iso8859-6, ibm-1089
ISO-8859-7	ecma-118, ibm813, csisolatingreek, elot-928, iso-ir-126, Cp813, 8859-7, iso-8859-7:1987, iso8859_7, greek, greek8, ibm-813, iso8859-7
ISO-8859-8	iso-ir-138, iso-8859-8:1988, csisolatinhebrew, hebrew, iso8859-8, 8859-8, ibm-916, iso8859_8, Cp916, ibm916
ISO-8859-9	ibm-920, ibm920, latin5, 8859-9, Cp920, l5, iso8859-9, iso8859_9, csisolatin5, iso-ir-148
JIS0212	
KOI8-R	koi8, ibm-878, cskoi8r, koi8_r
MacArabic	
MacCentralEurope	ibm-1282
MacCroatian	ibm-1284
MacCyrillic	ibm-1283
MacGreek	ibm-1280
MacIceland	ibm-1286
MacRoman	ibm-1275
MacRomania	ibm-1285
MacSymbol	Adobe-Symbol-Encoding, ibm-1038
MacTurkish	ibm-1281

Source Platform Default Encodings

If you do not specify an encoding for the source file or for the destination file, the default encoding for that platform will be used. The conversion is performed by the destination agent, and both source and destination encodings must be supported on the destination agent's platform for the conversion to take place. The destination default encoding will always be supported on the destination agent, so it is always safe to leave this unspecified. However, it might not be safe to use a default source encoding, because the destination agent might not support the source's default.

If you are using default source encodings, you should use the tables in this topic to make sure that the combination will be supported.

Platform	Default Encoding
Solaris	ISO-8859-1
SUSE Linux Enterprise Server on System x	UTF-8
HP-UX (Itanium)	ISO-8859-1
Linux for System z®	UTF-8
AIX	ISO-8859-1
Microsoft Windows	windows-1252
Red Hat Enterprise Linux on System x	UTF-8
Linux for System P	UTF-8
HP (PA-RISC)	ISO-8859-1

Platform-specific Encodings

Note: The following two tables contain the same information. It is organized in two different ways to help you find the correct information, depending whether you are looking up by platform or by encoding.

Encodings by Platform

Canonical names are listed in bold, followed by aliases in parentheses.

Platforms that support only encodings already listed in the Common Encodings table are not listed here.

Platform	Supported Encodings (not in Common Encodings table)
Solaris	**x-IBM33722** (ibm33722, 33722, ibm-33722_vascii_vpua, ibm-5050, ibm-33722, cp33722) **x-IBM930** (cp930, ibm930, ibm-930, 930) **x-IBM939** (ibm-939, ibm939, cp939, 939) **x-IBM964** (964, cp964, ibm-964, ibm964) **x-ISO-2022-CN-CNS** (ISO-2022-CN-CNS, ISO2022CN_CNS) **x-iso-8859-11** (iso-8859-11, iso8859_11) **x-JISAutoDetect** (JISAutoDetect) **x-MS932_0213** () **x-MS950-HKSCS** (MS950_HKSCS) **x-PCK** (pck) **x-SJIS_0213** () **X-UTF-32BE-BOM** (UTF_32BE_BOM, UTF-32BE-BOM) **x-MacUkraine** (macukraine) **x-MacThai** (macthai) **x-MacHebrew** (machebrew) **x-MacDingbat** (macdingbat) **x-KSC5601** (ksc5601) **x-JIS0208** (jis_c6226-1983, jis_x0208-1983, csiso87jisx0208, x0208, iso-ir-87, jis0208) **x-IBM949C** (ibm949c, cp949c, 949c, ibm-949c) **x-IBM943C** (cp943c, 943c, ibm-943c, ibm943c) **JIS_X0201** (jis_x0201, x0201, cshalfwidthkatakana, jis0201) **x-windows-iso2022jp** (windows-iso2022jp) **x-windows-50221** (ms50221, cp50221) **x-windows-50220** (cp50220, ms50220) **X-UTF-32LE-BOM** (UTF_32LE_BOM, UTF-32LE-BOM) **x-eucJP-Open** (EUC_JP_Solaris, eucJP-open) **x-Big5-Solaris** (Big5_Solaris) **ISO-2022-JP-2** (csISO2022JP2, iso2022jp2) **IBM918** (cp918, ebcdic-cp-ar2, ibm-918, 918) **IBM1047** (cp1047, 1047, ibm-1047) **IBM01149** (cp1149, cp01149, ccsid01149, 1149) **IBM01148** (cp1148, ccsid01148, 1148, cp01148) **IBM01147** (ccsid01147, cp1147, 1147, cp01147) **IBM01146** (ccsid01146, cp01146, cp1146, 1146) **IBM01145** (cp1145, cp01145, ccsid01145, 1145) **IBM01144** (cp01144, cp1144, ccsid01144, 1144) **IBM01143** (cp01143, 1143, ccsid01143, cp1143) **IBM01142** (cp01142, cp1142, 1142, ccsid01142) **IBM01141** (cp1141, ccsid01141, cp01141, 1141) **IBM01140** (ccsid01140, cp01140, 1140, cp1140) **IBM00858** (cp858, ccsid00858, 858, cp00858) **X-UnicodeLittle** (UnicodeLittle) **X-UnicodeBig** (UnicodeBig) **COMPOUND_TEXT** (x-compound-text, x11-compound-text) **IBM-942C** (Cp942C, ibm942C) **KOI8-U** (koi8_u, ibm-1167) **UTF-32** (UCS-4, UTF32, ISO-10646-UCS-4) **UTF-32BE** (UTF_32BE, X-UTF-32BE, UTF32BE) **UTF-32LE** (UTF_32LE, X-UTF-32LE, UTF32LE)

Platform	Supported Encodings (not in Common Encodings table)
SUSE Linux Enterprise Server on System x	**windows-1256S** (Cp1256s, ibm-1256s) **UTF-8J** (UTF8J) **UTF-32LE** (UTF_32LE, X-UTF-32LE, UTF32LE) **UTF-32BE** (UTF_32BE, X-UTF-32BE, UTF32BE) **UTF-32** (UCS-4, UTF32, ISO-10646-UCS-4) **PTCP154** (PT154, IBM-1169, Cyrillic-Asian, csPTCP154) **KOI8-RU** (ibm-1168, koi8_ru) **ISO-8859-16** (8859-16, iso8859_16, iso8859-16) **ISO-8859-14** (ISO_8859-14:1998, 8859-14, latin8, iso-ir-199, iso8859-14, l8, isoceltic, iso8859_14) **IBM01141** (cp1141, ccsid01141, cp01141, 1141) **IBM01142** (cp01142, cp1142, 1142, ccsid01142) **IBM01143** (cp01143, 1143, ccsid01143, cp1143) **IBM01144** (cp01144, cp1144, ccsid01144, 1144) **IBM01145** (cp1145, cp01145, ccsid01145, 1145) **IBM01146** (ccsid01146, cp01146, cp1146, 1146) **IBM01147** (ccsid01147, cp1147, 1147, cp01147) **IBM01148** (cp1148, ccsid01148, 1148, cp01148) **IBM01149** (cp1149, cp01149, ccsid01149, 1149) **IBM1047** (cp1047, 1047, ibm-1047) **IBM918** (cp918, ebcdic-cp-ar2, ibm-918, 918) **ISO-2022-JP-2** (csISO2022JP2, iso2022jp2) **x-Big5-Solaris** (Big5_Solaris) **x-eucJP-Open** (EUC_JP_Solaris, eucJP-open) **x-IBM33722** (ibm33722, 33722, ibm-33722_vascii_vpua, ibm-5050, ibm-33722, cp33722) **x-IBM930** (cp930, ibm930, ibm-930, 930) **x-IBM939** (ibm-939, ibm939, cp939, 939) **x-IBM964** (964, cp964, ibm-964, ibm964) **x-ISO-2022-CN-CNS** (ISO-2022-CN-CNS, ISO2022CN_CNS) **x-iso-8859-11** (iso-8859-11, iso8859_11) **x-JISAutoDetect** (JISAutoDetect) **x-MS932_0213** () **x-MS950-HKSCS** (MS950_HKSCS) **x-PCK** (pck) **x-IBM1363C** (ibm1363c, cp1363c, ibm-1363c) **x-IBM420S** (420s, ibm-420s, csibm420s, ibm420s, cp420s) **x-IBM864S** (csibm864s, ibm864s, cp864s, 864s, ibm-864s) **x-IBM943C** (cp943c, 943c, ibm-943c, ibm943c) **x-IBM949C** (ibm949c, cp949c, 949c, ibm-949c) **x-IBM954C** (cp954c, 954c, ibm-954c, ibm954c) **x-ISO-8859-6S** (8859_6s, iso8859-6s, iso8859_6s, iso-8859-6s) **x-JIS0208** (jis_c6226-1983, jis_x0208-1983, csiso87jisx0208, x0208, iso-ir-87, jis0208) **x-KSC5601** (ksc5601) **x-MacDingbat** (macdingbat) **x-MacHebrew** (machebrew) **x-MacThai** (macthai) **x-MacUkraine** (macukraine) **x-IBM1046S** (ibm-1046s, 1046s, cp1046s, ibm1046s) **x-IBM-udcJP** (IBM-udcJP) **JIS_X0201** (jis_x0201, x0201, cshalfwidthkatakana, jis0201) **IBM-939A** (Cp939A, ibm939A) **IBM-930A** (ibm930A, Cp930A) **IBM-33722A** (Cp33722A, ibm33722A) **x-windows-iso2022jp** (windows-iso2022jp) **x-windows-50221** (ms50221, cp50221) **x-windows-50220** (cp50220, ms50220) **X-UTF-32LE-BOM** (UTF_32LE_BOM, UTF-32LE-BOM) **X-UTF-32BE-BOM** (UTF_32BE_BOM, UTF-32BE-BOM) **x-SJIS_0213** () **IBM01140** (ccsid01140, cp01140, 1140, cp1140) **IBM00858** (cp858, ccsid00858, 858, cp00858) **X-UnicodeLittle** (UnicodeLittle) **X-UnicodeBig** (UnicodeBig) **IBM-859** (Cp859, ibm859)

Platform	Supported Encodings (not in Common Encodings table)
SUSE Linux Enterprise Server on System x	**IBM-837** (ibm837, Cp837) **IBM-836** (ibm836, Cp836) **IBM-835** (ibm835, Cp835) **IBM-833** (ibm833, Cp833) **IBM-808** (Cp808, ibm808) **IBM-720** (Cp720, ibm720) **IBM-33722C** (ibm-eucjp, Cp33722c) **IBM-301** (Cp301, ibm301) **IBM-300** (Cp300, ibm300) **IBM-290** (ibm290, Cp290) **IBM-1399** (ibm1399, Cp1399) **IBM-1390** (Cp1390, ibm1390) **IBM-1388** (Cp1388, ibm1388) **IBM-1385** (Cp1385, ibm1385) **IBM-1382** (ibm1382, Cp1382) **IBM-1088** (Cp1088, ibm1088) **IBM-1043** (Cp1043, ibm1043) **IBM-1041** (Cp1041, ibm1041) **IBM-1027** (Cp1027, ibm1027) **CESU-8** (CESU8) **COMPOUND_TEXT** (x-compound-text, x11-compound-text) **GB2312** (gb2312-1980, gb2312-80) **GBK** (GBK) **hp-roman8** (roman8, ibm-1051, r8, Cp1051) **IBM-1114** (Cp1114, ibm1114) **IBM-1115** (Cp1115, ibm1115) **IBM-1351** (Cp1351, ibm1351) **IBM-1362** (Cp1362, ibm1362) **IBM-1363** (ibm1363, Cp1363) **IBM-1364** (Cp1364, ibm1364) **IBM-1370** (Cp1370, ibm1370) **IBM-1371** (Cp1371, ibm1371) **IBM-1380** (Cp1380, ibm1380) **IBM-867** (Cp867, ibm867) **IBM-897** (Cp897, ibm897) **IBM-924** (Cp924, ibm924) **IBM-927** (ibm927, Cp927) **IBM-932** (ibm932, Cp932) **IBM-947** (Cp947, ibm947) **IBM-951** (Cp951, ibm951) **IBM-954** (ibm954, Cp954) **IBM-971** (Cp971, ibm971) **ISO-8859-10** (latin6, 8859-10, ISO_8859-10:1992, iso8859_10, iso-ir-157, ibm-919, iso8859-10, l6, csisolatin6)

Platform	Supported Encodings (not in Common Encodings table)
HP-UX (Itanium)	**UTF-16** (UTF16, Unicode, UTF_16, UCS-2) **MacUkraine** () **MacThai** () **MacHebrew** () **MacDingbat** () **JIS0208** () **JIS0201** () **IBM-949C** (Cp949C, ibm949C) **IBM-943C** (ibm943C, Cp943C) **IBM-942C** (Cp942C, ibm942C) **IBM00858** (cp858, ccsid00858, 858, cp00858) **IBM01140** (ccsid01140, cp01140, 1140, cp1140) **x-eucJP-Open** (EUC_JP_Solaris, eucJP-open) **x-IBM33722** (ibm33722, 33722, ibm-33722_vascii_vpua, ibm-5050, ibm-33722, cp33722) **x-IBM930** (cp930, ibm930, ibm-930, 930) **x-IBM939** (ibm-939, ibm939, cp939, 939) **x-IBM964** (964, cp964, ibm-964, ibm964) **x-ISO-2022-CN-CNS** (ISO-2022-CN-CNS, ISO2022CN_CNS) **x-iso-8859-11** (iso-8859-11, iso8859_11) **x-JISAutoDetect** (JISAutoDetect) **x-MS950-HKSCS** (MS950_HKSCS) **x-PCK** (pck) **x-windows-50220** (cp50220, ms50220) **x-windows-50221** (ms50221, cp50221) **x-windows-iso2022jp** (windows-iso2022jp) **x-Big5-Solaris** (Big5_Solaris) **IBM918** (cp918, ebcdic-cp-ar2, ibm-918, 918) **IBM1047** (cp1047, 1047, ibm-1047) **IBM01149** (cp1149, cp01149, ccsid01149, 1149) **IBM01148** (cp1148, ccsid01148, 1148, cp01148) **IBM01147** (ccsid01147, cp1147, 1147, cp01147) **IBM01146** (ccsid01146, cp01146, cp1146, 1146) **IBM01145** (cp1145, cp01145, ccsid01145, 1145) **IBM01144** (cp01144, cp1144, ccsid01144, 1144) **IBM01143** (cp01143, 1143, ccsid01143, cp1143) **IBM01142** (cp01142, cp1142, 1142, ccsid01142) **IBM01141** (cp1141, ccsid01141, cp01141, 1141)

Platform	Supported Encodings (not in Common Encodings table)
Linux for System z	**windows-1256S** (Cp1256s, ibm-1256s) **UTF-8J** (UTF8J) **UTF-32LE** (UTF_32LE, X-UTF-32LE, UTF32LE) **UTF-32BE** (UTF_32BE, X-UTF-32BE, UTF32BE) **UTF-32** (UCS-4, UTF32, ISO-10646-UCS-4) **PTCP154** (PT154, IBM-1169, Cyrillic-Asian, csPTCP154) **KOI8-RU** (ibm-1168, koi8_ru) **ISO-8859-16** (8859-16, iso8859_16, iso8859-16) **ISO-8859-14** (ISO_8859-14:1998, 8859-14, latin8, iso-ir-199, iso8859-14, l8, isoceltic, iso8859_14) **IBM01141** (cp1141, ccsid01141, cp01141, 1141) **IBM01142** (cp01142, cp1142, 1142, ccsid01142) **IBM01143** (cp01143, 1143, ccsid01143, cp1143) **IBM01144** (cp01144, cp1144, ccsid01144, 1144) **IBM01145** (cp1145, cp01145, ccsid01145, 1145) **IBM01146** (ccsid01146, cp01146, cp1146, 1146) **IBM01147** (ccsid01147, cp1147, 1147, cp01147) **IBM01148** (cp1148, ccsid01148, 1148, cp01148) **IBM01149** (cp1149, cp01149, ccsid01149, 1149) **IBM1047** (cp1047, 1047, ibm-1047) **IBM918** (cp918, ebcdic-cp-ar2, ibm-918, 918) **ISO-2022-JP-2** (csISO2022JP2, iso2022jp2) **x-Big5-Solaris** (Big5_Solaris) **x-eucJP-Open** (EUC_JP_Solaris, eucJP-open) **x-IBM33722** (ibm33722, 33722, ibm-33722_vascii_vpua, ibm-5050, ibm-33722, cp33722) **x-IBM930** (cp930, ibm930, ibm-930, 930) **x-IBM939** (ibm-939, ibm939, cp939, 939) **x-IBM964** (964, cp964, ibm-964, ibm964) **x-ISO-2022-CN-CNS** (ISO-2022-CN-CNS, ISO2022CN_CNS) **x-iso-8859-11** (iso-8859-11, iso8859_11) **x-JISAutoDetect** (JISAutoDetect) **x-MS932_0213** () **x-MS950-HKSCS** (MS950_HKSCS) **x-PCK** (pck) **x-IBM1363C** (ibm1363c, cp1363c, ibm-1363c) **x-IBM420S** (420s, ibm-420s, csibm420s, ibm420s, cp420s) **x-IBM864S** (csibm864s, ibm864s, cp864s, 864s, ibm-864s) **x-IBM943C** (cp943c, 943c, ibm-943c, ibm943c) **x-IBM949C** (ibm949c, cp949c, 949c, ibm-949c) **x-IBM954C** (cp954c, 954c, ibm-954c, ibm954c) **x-ISO-8859-6S** (8859_6s, iso8859-6s, iso8859_6s, iso-8859-6s) **x-JIS0208** (jis_c6226-1983, jis_x0208-1983, csiso87jisx0208, x0208, iso-ir-87, jis0208) **x-KSC5601** (ksc5601) **x-MacDingbat** (macdingbat) **x-MacHebrew** (machebrew) **x-MacThai** (macthai) **x-MacUkraine** (macukraine) **x-IBM1046S** (ibm-1046s, 1046s, cp1046s, ibm1046s) **x-IBM-udcJP** (IBM-udcJP) **JIS_X0201** (jis_x0201, x0201, cshalfwidthkatakana, jis0201) **IBM-939A** (Cp939A, ibm939A) **IBM-930A** (ibm930A, Cp930A) **IBM-33722A** (Cp33722A, ibm33722A) **x-windows-iso2022jp** (windows-iso2022jp) **x-windows-50221** (ms50221, cp50221) **x-windows-50220** (cp50220, ms50220) **X-UTF-32LE-BOM** (UTF_32LE_BOM, UTF-32LE-BOM) **X-UTF-32BE-BOM** (UTF_32BE_BOM, UTF-32BE-BOM) **x-SJIS_0213** () **IBM01140** (ccsid01140, cp01140, 1140, cp1140) **IBM00858** (cp858, ccsid00858, 858, cp00858) **X-UnicodeLittle** (UnicodeLittle) **X-UnicodeBig** (UnicodeBig) **IBM-859** (Cp859, ibm859)

Platform	Supported Encodings (not in Common Encodings table)
Linux for System z	**IBM-837** (ibm837, Cp837)
	IBM-836 (ibm836, Cp836)
	IBM-835 (ibm835, Cp835)
	IBM-833 (ibm833, Cp833)
	IBM-808 (Cp808, ibm808)
	IBM-720 (Cp720, ibm720)
	IBM-33722C (ibm-eucjp, Cp33722c)
	IBM-301 (Cp301, ibm301)
	IBM-300 (Cp300, ibm300)
	IBM-290 (ibm290, Cp290)
	IBM-1399 (ibm1399, Cp1399)
	IBM-1390 (Cp1390, ibm1390)
	IBM-1388 (Cp1388, ibm1388)
	IBM-1385 (Cp1385, ibm1385)
	IBM-1382 (ibm1382, Cp1382)
	IBM-1088 (Cp1088, ibm1088)
	IBM-1043 (Cp1043, ibm1043)
	IBM-1041 (Cp1041, ibm1041)
	IBM-1027 (Cp1027, ibm1027)
	CESU-8 (CESU8)
	COMPOUND_TEXT (x-compound-text, x11-compound-text)
	GB2312 (gb2312-1980, gb2312-80)
	GBK (GBK)
	hp-roman8 (roman8, ibm-1051, r8, Cp1051)
	IBM-1114 (Cp1114, ibm1114)
	IBM-1115 (Cp1115, ibm1115)
	IBM-1351 (Cp1351, ibm1351)
	IBM-1362 (Cp1362, ibm1362)
	IBM-1363 (ibm1363, Cp1363)
	IBM-1364 (Cp1364, ibm1364)
	IBM-1370 (Cp1370, ibm1370)
	IBM-1371 (Cp1371, ibm1371)
	IBM-1380 (Cp1380, ibm1380)
	IBM-867 (Cp867, ibm867)
	IBM-897 (Cp897, ibm897)
	IBM-924 (Cp924, ibm924)
	IBM-927 (ibm927, Cp927)
	IBM-932 (ibm932, Cp932)
	IBM-947 (Cp947, ibm947)
	IBM-951 (Cp951, ibm951)
	IBM-954 (ibm954, Cp954)
	IBM-971 (Cp971, ibm971)
	ISO-8859-10 (latin6, 8859-10, ISO_8859-10:1992, iso8859_10, iso-ir-157, ibm-919, iso8859-10, l6, csisolatin6)

Platform	Supported Encodings (not in Common Encodings table)
AIX	**windows-1256S** (Cp1256s, ibm-1256s)
	UTF-8J (UTF8J)
	UTF-32LE (UTF_32LE, X-UTF-32LE, UTF32LE)
	UTF-32BE (UTF_32BE, X-UTF-32BE, UTF32BE)
	UTF-32 (UCS-4, UTF32, ISO-10646-UCS-4)
	UTF-16 (UTF16, Unicode, UTF_16, UCS-2)
	Shift_JIS ()
	PTCP154 (PT154, IBM-1169, Cyrillic-Asian, csPTCP154)
	MacUkraine ()
	MacThai ()
	MacHebrew ()
	MacDingbat ()
	KSC5601 ()
	KOI8-U (koi8_u, ibm-1167)
	KOI8-RU (ibm-1168, koi8_ru)
	Johab (x-johab)
	JIS0208 ()
	JIS0201 ()
	ISO-8859-6S (iso8859-6S, iso8859_6S)
	ISO-8859-16 (8859-16, iso8859_16, iso8859-16)
	ISO-8859-14 (ISO_8859-14:1998, 8859-14, latin8, iso-ir-199, iso8859-14, l8, isoceltic, iso8859_14)
	ISO-8859-10 (latin6, 8859-10, ISO_8859-10:1992, iso8859_10, iso-ir-157, ibm-919, iso8859-10, l6, csisolatin6)
	IBM-971 (Cp971, ibm971)
	IBM-964 (ibm-euctw, Cp964)
	IBM-954C (Cp954c)
	IBM-954 (ibm954, Cp954)
	IBM-951 (Cp951, ibm951)
	IBM-949C (Cp949C, ibm949C)
	IBM-947 (Cp947, ibm947)
	IBM-943C (ibm943C, Cp943C)
	IBM-942C (Cp942C, ibm942C)
	IBM-939 (Cp5035, 5035)
	IBM-932 (ibm932, Cp932)
	IBM-930 (Cp5026, 5026)
	IBM-927 (ibm927, Cp927)
	IBM-924 (Cp924, ibm924)
	IBM-918 (ibm918, Cp918)
	IBM-897 (Cp897, ibm897)
	IBM-867 (Cp867, ibm867)
	IBM-1380 (Cp1380, ibm1380)
	IBM-1371 (Cp1371, ibm1371)
	IBM-1370 (Cp1370, ibm1370)
	IBM-1364 (Cp1364, ibm1364)
	IBM-1363C (ibm1363C, Cp1363C)
	IBM-1047 (Cp1047, ibm1047)
	IBM-1088 (Cp1088, ibm1088)
	IBM-1382 (ibm1382, Cp1382)
	IBM-1385 (Cp1385, ibm1385)
	IBM-1386 (ibm1386, Cp1386)
	IBM-1388 (Cp1388, ibm1388)
	IBM-1390 (Cp1390, ibm1390)
	IBM-1399 (ibm1399, Cp1399)
	IBM-290 (ibm290, Cp290)
	IBM-300 (Cp300, ibm300)
	IBM-301 (Cp301, ibm301)
	IBM-33722 (5050, Cp5050)
	X-UnicodeLittle (UnicodeLittle)
	X-UnicodeBig (UnicodeBig)
	IBM-864S (ibm864S, Cp864S)
	IBM-859 (Cp859, ibm859)
	IBM-858 (Cp858, ibm858)

Platform	Supported Encodings (not in Common Encodings table)
AIX	**IBM-837** (ibm837, Cp837) **IBM-836** (ibm836, Cp836) **IBM-835** (ibm835, Cp835) **IBM-833** (ibm833, Cp833) **IBM-808** (Cp808, ibm808) **IBM-720** (Cp720, ibm720) **IBM-420S** (Cp420S, ibm420S) **IBM-33722C** (ibm-eucjp, Cp33722c) **IBM-1046S** (ibm1046S, Cp1046S) **IBM-1043** (Cp1043, ibm1043) **IBM-1041** (Cp1041, ibm1041) **IBM-1027** (Cp1027, ibm1027) **CESU-8** (CESU8) **COMPOUND_TEXT** (x-compound-text, x11-compound-text) **GB2312** (gb2312-1980, gb2312-80) **GBK** (GBK) **hp-roman8** (roman8, ibm-1051, r8, Cp1051) **IBM-1114** (Cp1114, ibm1114) **IBM-1115** (Cp1115, ibm1115) **IBM-1140** (ibm1140, Cp1140) **IBM-1141** (Cp1141, ibm1141) **IBM-1142** (Cp1142, ibm1142) **IBM-1143** (Cp1143, ibm1143) **IBM-1144** (ibm1144, Cp1144) **IBM-1145** (Cp1145, ibm1145) **IBM-1146** (Cp1146, ibm1146) **IBM-1147** (Cp1147, ibm1147) **IBM-1148** (ibm1148, Cp1148) **IBM-1149** (Cp1149, ibm1149) **IBM-1351** (Cp1351, ibm1351) **IBM-1362** (Cp1362, ibm1362) **IBM-1363** (ibm1363, Cp1363)

Platform	Supported Encodings (not in Common Encodings table)
Microsoft Windows	**windows-1256S** (Cp1256s, ibm-1256s) **UTF-8J** (UTF8J) **UTF-32LE** (UTF_32LE, X-UTF-32LE, UTF32LE) **UTF-32BE** (UTF_32BE, X-UTF-32BE, UTF32BE) **PTCP154** (PT154, IBM-1169, Cyrillic-Asian, csPTCP154) **KOI8-RU** (ibm-1168, koi8_ru) **ISO-8859-16** (8859-16, iso8859_16, iso8859-16) **ISO-8859-14** (ISO_8859-14:1998, 8859-14, latin8, iso-ir-199, iso8859-14, l8, isoceltic, iso8859_14) **IBM01141** (cp1141, ccsid01141, cp01141, 1141) **IBM01142** (cp01142, cp1142, 1142, ccsid01142) **IBM01143** (cp01143, 1143, ccsid01143, cp1143) **IBM01144** (cp01144, cp1144, ccsid01144, 1144) **IBM01145** (cp1145, cp01145, ccsid01145, 1145) **IBM01146** (ccsid01146, cp01146, cp1146, 1146) **IBM01147** (ccsid01147, cp1147, 1147, cp01147) **IBM01148** (cp1148, ccsid01148, 1148, cp01148) **IBM01149** (cp1149, cp01149, ccsid01149, 1149) **IBM1047** (cp1047, 1047, ibm-1047) **ISO-2022-JP-2** (csISO2022JP2, iso2022jp2) **x-Big5-Solaris** (Big5_Solaris) **x-eucJP-Open** (EUC_JP_Solaris, eucJP-open) **x-IBM33722** (ibm33722, 33722, ibm-33722_vascii_vpua, ibm-5050, ibm-33722, cp33722) **x-IBM930** (cp930, ibm930, ibm-930, 930) **x-IBM939** (ibm-939, ibm939, cp939, 939) **x-IBM964** (964, cp964, ibm-964, ibm964) **x-ISO-2022-CN-CNS** (ISO-2022-CN-CNS, ISO2022CN_CNS) **x-iso-8859-11** (iso-8859-11, iso8859_11) **x-JISAutoDetect** (JISAutoDetect) **x-MS932_0213** () **x-MS950-HKSCS** (MS950_HKSCS) **x-PCK** (pck) **x-IBM1363C** (ibm1363c, cp1363c, ibm-1363c) **x-IBM420S** (420s, ibm-420s, csibm420s, ibm420s, cp420s) **x-IBM864S** (csibm864s, ibm864s, cp864s, 864s, ibm-864s) **x-IBM943C** (cp943c, 943c, ibm-943c, ibm943c) **x-IBM949C** (ibm949c, cp949c, 949c, ibm-949c) **x-IBM954C** (cp954c, 954c, ibm-954c, ibm954c) **x-ISO-8859-6S** (8859_6s, iso8859-6s, iso8859_6s, iso-8859-6s) **x-JIS0208** (jis_c6226-1983, jis_x0208-1983, csiso87jisx0208, x0208, iso-ir-87, jis0208) **x-KSC5601** (ksc5601) **x-MacDingbat** (macdingbat) **x-MacHebrew** (machebrew) **x-MacThai** (macthai) **x-MacUkraine** (macukraine) **x-IBM1046S** (ibm-1046s, 1046s, cp1046s, ibm1046s) **x-IBM-udcJP** (IBM-udcJP) **JIS_X0201** (jis_x0201, x0201, cshalfwidthkatakana, jis0201) **IBM-939A** (Cp939A, ibm939A) **IBM-930A** (ibm930A, Cp930A) **IBM-33722A** (Cp33722A, ibm33722A) **x-windows-iso2022jp** (windows-iso2022jp) **x-windows-50221** (ms50221, cp50221) **x-windows-50220** (cp50220, ms50220) **X-UTF-32LE-BOM** (UTF_32LE_BOM, UTF-32LE-BOM) **X-UTF-32BE-BOM** (UTF_32BE_BOM, UTF-32BE-BOM) **x-SJIS_0213** () **IBM01140** (ccsid01140, cp01140, 1140, cp1140) **IBM00858** (cp858, ccsid00858, 858, cp00858) **X-UnicodeLittle** (UnicodeLittle) **X-UnicodeBig** (UnicodeBig) **IBM-859** (Cp859, ibm859) **IBM-837** (ibm837, Cp837)

Platform	Supported Encodings (not in Common Encodings table)
Microsoft Windows	**IBM-836** (ibm836, Cp836) **IBM-835** (ibm835, Cp835) **IBM-833** (ibm833, Cp833) **IBM-808** (Cp808, ibm808) **IBM-720** (Cp720, ibm720) **IBM-33722C** (ibm-eucjp, Cp33722c) **IBM-301** (Cp301, ibm301) **IBM-300** (Cp300, ibm300) **IBM-290** (ibm290, Cp290) **IBM-1399** (ibm1399, Cp1399) **IBM-1390** (Cp1390, ibm1390) **IBM-1388** (Cp1388, ibm1388) **IBM-1385** (Cp1385, ibm1385) **IBM-1382** (ibm1382, Cp1382) **IBM-1088** (Cp1088, ibm1088) **IBM-1043** (Cp1043, ibm1043) **IBM-1041** (Cp1041, ibm1041) **IBM-1027** (Cp1027, ibm1027) **CESU-8** (CESU8) **COMPOUND_TEXT** (x-compound-text, x11-compound-text) **GB2312** (gb2312-1980, gb2312-80) **GBK** (GBK) **hp-roman8** (roman8, ibm-1051, r8, Cp1051) **IBM-1115** (Cp1115, ibm1115) **IBM-1351** (Cp1351, ibm1351) **IBM-1362** (Cp1362, ibm1362) **IBM-1363** (ibm1363, Cp1363) **IBM-1364** (Cp1364, ibm1364) **IBM-1370** (Cp1370, ibm1370) **IBM-1371** (Cp1371, ibm1371) **IBM-1380** (Cp1380, ibm1380) **IBM-867** (Cp867, ibm867) **IBM-897** (Cp897, ibm897) **IBM-924** (Cp924, ibm924) **IBM-927** (ibm927, Cp927) **IBM-932** (ibm932, Cp932) **IBM-947** (Cp947, ibm947) **IBM-951** (Cp951, ibm951) **IBM-954** (ibm954, Cp954) **IBM-971** (Cp971, ibm971) **ISO-8859-10** (latin6, 8859-10, ISO_8859-10:1992, iso8859_10, iso-ir-157, ibm-919, iso8859-10, l6, csisolatin6)

Platform	Supported Encodings (not in Common Encodings table)
Red Hat Enterprise Linux on System x	**windows-1256S** (Cp1256s, ibm-1256s) **UTF-8J** (UTF8J) **UTF-32LE** (UTF_32LE, X-UTF-32LE, UTF32LE) **UTF-32BE** (UTF_32BE, X-UTF-32BE, UTF32BE) **UTF-32** (UCS-4, UTF32, ISO-10646-UCS-4) **PTCP154** (PT154, IBM-1169, Cyrillic-Asian, csPTCP154) **KOI8-RU** (ibm-1168, koi8_ru) **ISO-8859-16** (8859-16, iso8859_16, iso8859-16) **ISO-8859-14** (ISO_8859-14:1998, 8859-14, latin8, iso-ir-199, iso8859-14, l8, isoceltic, iso8859_14) **IBM01141** (cp1141, ccsid01141, cp01141, 1141) **IBM01142** (cp01142, cp1142, 1142, ccsid01142) **IBM01143** (cp01143, 1143, ccsid01143, cp1143) **IBM01144** (cp01144, cp1144, ccsid01144, 1144) **IBM01145** (cp1145, cp01145, ccsid01145, 1145) **IBM01146** (ccsid01146, cp01146, cp1146, 1146) **IBM01147** (ccsid01147, cp1147, 1147, cp01147) **IBM01148** (cp1148, ccsid01148, 1148, cp01148) **IBM01149** (cp1149, cp01149, ccsid01149, 1149) **IBM1047** (cp1047, 1047, ibm-1047) **IBM918** (cp918, ebcdic-cp-ar2, ibm-918, 918) **ISO-2022-JP-2** (csISO2022JP2, iso2022jp2) **x-Big5-Solaris** (Big5_Solaris) **x-eucJP-Open** (EUC_JP_Solaris, eucJP-open) **x-IBM33722** (ibm33722, 33722, ibm-33722_vascii_vpua, ibm-5050, ibm-33722, cp33722) **x-IBM930** (cp930, ibm930, ibm-930, 930) **x-IBM939** (ibm-939, ibm939, cp939, 939) **x-IBM964** (964, cp964, ibm-964, ibm964) **x-ISO-2022-CN-CNS** (ISO-2022-CN-CNS, ISO2022CN_CNS) **x-iso-8859-11** (iso-8859-11, iso8859_11) **x-JISAutoDetect** (JISAutoDetect) **x-MS932_0213** () **x-MS950-HKSCS** (MS950_HKSCS) **x-PCK** (pck) **x-IBM1363C** (ibm1363c, cp1363c, ibm-1363c) **x-IBM420S** (420s, ibm-420s, csibm420s, ibm420s, cp420s) **x-IBM864S** (csibm864s, ibm864s, cp864s, 864s, ibm-864s) **x-IBM943C** (cp943c, 943c, ibm-943c, ibm943c) **x-IBM949C** (ibm949c, cp949c, 949c, ibm-949c) **x-IBM954C** (cp954c, 954c, ibm-954c, ibm954c) **x-ISO-8859-6S** (8859_6s, iso8859-6s, iso8859_6s, iso-8859-6s) **x-JIS0208** (jis_c6226-1983, jis_x0208-1983, csiso87jisx0208, x0208, iso-ir-87, jis0208) **x-KSC5601** (ksc5601) **x-MacDingbat** (macdingbat) **x-MacHebrew** (machebrew) **x-MacThai** (macthai) **x-MacUkraine** (macukraine) **x-IBM1046S** (ibm-1046s, 1046s, cp1046s, ibm1046s) **x-IBM-udcJP** (IBM-udcJP) **JIS_X0201** (jis_x0201, x0201, cshalfwidthkatakana, jis0201) **IBM-939A** (Cp939A, ibm939A) **IBM-930A** (ibm930A, Cp930A) **IBM-33722A** (Cp33722A, ibm33722A) **x-windows-iso2022jp** (windows-iso2022jp) **x-windows-50221** (ms50221, cp50221) **x-windows-50220** (cp50220, ms50220) **X-UTF-32LE-BOM** (UTF_32LE_BOM, UTF-32LE-BOM) **X-UTF-32BE-BOM** (UTF_32BE_BOM, UTF-32BE-BOM) **x-SJIS_0213** () **IBM01140** (ccsid01140, cp01140, 1140, cp1140) **IBM00858** (cp858, ccsid00858, 858, cp00858) **X-UnicodeLittle** (UnicodeLittle) **X-UnicodeBig** (UnicodeBig) **IBM-859** (Cp859, ibm859)

Platform	Supported Encodings (not in Common Encodings table)
Red Hat Enterprise Linux on System x	**IBM-837** (ibm837, Cp837) **IBM-836** (ibm836, Cp836) **IBM-835** (ibm835, Cp835) **IBM-833** (ibm833, Cp833) **IBM-808** (Cp808, ibm808) **IBM-720** (Cp720, ibm720) **IBM-33722C** (ibm-eucjp, Cp33722c) **IBM-301** (Cp301, ibm301) **IBM-300** (Cp300, ibm300) **IBM-290** (ibm290, Cp290) **IBM-1399** (ibm1399, Cp1399) **IBM-1390** (Cp1390, ibm1390) **IBM-1388** (Cp1388, ibm1388) **IBM-1385** (Cp1385, ibm1385) **IBM-1382** (ibm1382, Cp1382) **IBM-1088** (Cp1088, ibm1088) **IBM-1043** (Cp1043, ibm1043) **IBM-1041** (Cp1041, ibm1041) **IBM-1027** (Cp1027, ibm1027) **CESU-8** (CESU8) **COMPOUND_TEXT** (x-compound-text, x11-compound-text) **GB2312** (gb2312-1980, gb2312-80) **GBK** (GBK) **hp-roman8** (roman8, ibm-1051, r8, Cp1051) **IBM-1114** (Cp1114, ibm1114) **IBM-1115** (Cp1115, ibm1115) **IBM-1351** (Cp1351, ibm1351) **IBM-1362** (Cp1362, ibm1362) **IBM-1363** (ibm1363, Cp1363) **IBM-1364** (Cp1364, ibm1364) **IBM-1370** (Cp1370, ibm1370) **IBM-1371** (Cp1371, ibm1371) **IBM-1380** (Cp1380, ibm1380) **IBM-867** (Cp867, ibm867) **IBM-897** (Cp897, ibm897) **IBM-924** (Cp924, ibm924) **IBM-927** (ibm927, Cp927) **IBM-932** (ibm932, Cp932) **IBM-947** (Cp947, ibm947) **IBM-951** (Cp951, ibm951) **IBM-954** (ibm954, Cp954) **IBM-971** (Cp971, ibm971) **ISO-8859-10** (latin6, 8859-10, ISO_8859-10:1992, iso8859_10, iso-ir-157, ibm-919, iso8859-10, l6, csisolatin6)

Platform	Supported Encodings (not in Common Encodings table)
Linux for System P	**windows-1256S** (Cp1256s, ibm-1256s) **UTF-8J** (UTF8J) **UTF-32LE** (UTF_32LE, X-UTF-32LE, UTF32LE) **UTF-32BE** (UTF_32BE, X-UTF-32BE, UTF32BE) **UTF-32** (UCS-4, UTF32, ISO-10646-UCS-4) **UTF-16** (UTF16, Unicode, UTF_16, UCS-2) **Shift_JIS** () **PTCP154** (PT154, IBM-1169, Cyrillic-Asian, csPTCP154) **MacUkraine** () **MacThai** () **MacHebrew** () **MacDingbat** () **KSC5601** () **KOI8-U** (koi8_u, ibm-1167) **KOI8-RU** (ibm-1168, koi8_ru) **Johab** (x-johab) **JIS0208** () **JIS0201** () **ISO-8859-6S** (iso8859-6S, iso8859_6S) **ISO-8859-16** (8859-16, iso8859_16, iso8859-16) **ISO-8859-14** (ISO_8859-14:1998, 8859-14, latin8, iso-ir-199, iso8859-14, l8, isoceltic, iso8859_14) **ISO-8859-10** (latin6, 8859-10, ISO_8859-10:1992, iso8859_10, iso-ir-157, ibm-919, iso8859-10, l6, csisolatin6) **IBM-971** (Cp971, ibm971) **IBM-964** (ibm-euctw, Cp964) **IBM-954C** (Cp954c) **IBM-954** (ibm954, Cp954) **IBM-951** (Cp951, ibm951) **IBM-949C** (Cp949C, ibm949C) **IBM-947** (Cp947, ibm947) **IBM-943C** (ibm943C, Cp943C) **IBM-942C** (Cp942C, ibm942C) **IBM-939** (Cp5035, 5035) **IBM-932** (ibm932, Cp932) **IBM-930** (Cp5026, 5026) **IBM-927** (ibm927, Cp927) **IBM-924** (Cp924, ibm924) **IBM-918** (ibm918, Cp918) **IBM-897** (Cp897, ibm897) **IBM-867** (Cp867, ibm867) **IBM-1380** (Cp1380, ibm1380) **IBM-1371** (Cp1371, ibm1371) **IBM-1370** (Cp1370, ibm1370) **IBM-1364** (Cp1364, ibm1364) **IBM-1363C** (ibm1363C, Cp1363C) **IBM-1047** (Cp1047, ibm1047) **IBM-1088** (Cp1088, ibm1088) **IBM-1382** (ibm1382, Cp1382) **IBM-1385** (Cp1385, ibm1385) **IBM-1386** (ibm1386, Cp1386) **IBM-1388** (Cp1388, ibm1388) **IBM-1390** (Cp1390, ibm1390) **IBM-1399** (ibm1399, Cp1399) **IBM-290** (ibm290, Cp290) **IBM-300** (Cp300, ibm300) **IBM-301** (Cp301, ibm301) **IBM-33722** (5050, Cp5050) **X-UnicodeLittle** (UnicodeLittle) **X-UnicodeBig** (UnicodeBig) **IBM-864S** (ibm864S, Cp864S) **IBM-859** (Cp859, ibm859) **IBM-858** (Cp858, ibm858)

Platform	Supported Encodings (not in Common Encodings table)
Linux for System P	**IBM-837** (ibm837, Cp837) **IBM-836** (ibm836, Cp836) **IBM-835** (ibm835, Cp835) **IBM-833** (ibm833, Cp833) **IBM-808** (Cp808, ibm808) **IBM-720** (Cp720, ibm720) **IBM-420S** (Cp420S, ibm420S) **IBM-33722C** (ibm-eucjp, Cp33722c) **IBM-1046S** (ibm1046S, Cp1046S) **IBM-1043** (Cp1043, ibm1043) **IBM-1041** (Cp1041, ibm1041) **IBM-1027** (Cp1027, ibm1027) **CESU-8** (CESU8) **COMPOUND_TEXT** (x-compound-text, x11-compound-text) **GB2312** (gb2312-1980, gb2312-80) **GBK** (GBK) **hp-roman8** (roman8, ibm-1051, r8, Cp1051) **IBM-1114** (Cp1114, ibm1114) **IBM-1115** (Cp1115, ibm1115) **IBM-1140** (ibm1140, Cp1140) **IBM-1141** (Cp1141, ibm1141) **IBM-1142** (Cp1142, ibm1142) **IBM-1143** (Cp1143, ibm1143) **IBM-1144** (ibm1144, Cp1144) **IBM-1145** (Cp1145, ibm1145) **IBM-1146** (Cp1146, ibm1146) **IBM-1147** (Cp1147, ibm1147) **IBM-1148** (ibm1148, Cp1148) **IBM-1149** (Cp1149, ibm1149) **IBM-1351** (Cp1351, ibm1351) **IBM-1362** (Cp1362, ibm1362) **IBM-1363** (ibm1363, Cp1363)

Platform	Supported Encodings (not in Common Encodings table)
HP (PA-RISC)	**UTF-32LE** (UTF_32LE, X-UTF-32LE, UTF32LE) **UTF-32BE** (UTF_32BE, X-UTF-32BE, UTF32BE) **IBM01147** (ccsid01147, cp1147, 1147, cp01147) **IBM01148** (cp1148, ccsid01148, 1148, cp01148) **IBM01149** (cp1149, cp01149, ccsid01149, 1149) **IBM1047** (cp1047, 1047, ibm-1047) **IBM918** (cp918, ebcdic-cp-ar2, ibm-918, 918) **ISO-2022-JP-2** (csISO2022JP2, iso2022jp2) **Roman9** (Roman9) **x-Big5-Solaris** (Big5_Solaris) **x-eucJP-Open** (EUC_JP_Solaris, eucJP-open) **x-IBM33722** (ibm33722, 33722, ibm-33722_vascii_vpua, ibm-5050, ibm-33722, cp33722) **x-IBM930** (cp930, ibm930, ibm-930, 930) **x-IBM939** (ibm-939, ibm939, cp939, 939) **x-windows-iso2022jp** (windows-iso2022jp) **x-windows-50221** (ms50221, cp50221) **x-windows-50220** (cp50220, ms50220) **X-UTF-32LE-BOM** (UTF_32LE_BOM, UTF-32LE-BOM) **X-UTF-32BE-BOM** (UTF_32BE_BOM, UTF-32BE-BOM) **x-SJIS_0213** () **x-PCK** (pck) **x-MS950-HKSCS** (MS950_HKSCS) **x-MS932_0213** () **x-JISAutoDetect** (JISAutoDetect) **x-iso-8859-11** (iso-8859-11, iso8859_11) **x-ISO-2022-CN-CNS** (ISO-2022-CN-CNS, ISO2022CN_CNS) **x-IBM964** (964, cp964, ibm-964, ibm964) **IBM01146** (ccsid01146, cp01146, cp1146, 1146) **IBM01145** (cp1145, cp01145, ccsid01145, 1145) **IBM01144** (cp01144, cp1144, ccsid01144, 1144) **IBM01143** (cp01143, 1143, ccsid01143, cp1143) **IBM01142** (cp01142, cp1142, 1142, ccsid01142) **IBM01141** (cp1141, ccsid01141, cp01141, 1141) **IBM01140** (ccsid01140, cp01140, 1140, cp1140) **IBM00858** (cp858, ccsid00858, 858, cp00858) **X-UnicodeLittle** (UnicodeLittle) **X-UnicodeBig** (UnicodeBig) **COMPOUND_TEXT** (x-compound-text, x11-compound-text) **hp-roman8** (roman8, ibm-1051, r8, Cp1051) **IBM-1364** (Cp1364, ibm1364) **IBM-942C** (Cp942C, ibm942C) **IBM-943C** (ibm943C, Cp943C) **IBM-949C** (Cp949C, ibm949C) **JIS0201** () **JIS0208** () **KOI8-U** (koi8_u, ibm-1167) **MacDingbat** () **MacHebrew** () **MacThai** () **MacUkraine** () **UTF-32** (UCS-4, UTF32, ISO-10646-UCS-4)

Platforms by Encoding

Encoding	Aliases	Platforms on which this encoding is supported
x-MacUkraine	macukraine	Solaris, SUSE Linux Enterprise Server on System x, Linux for System z, Microsoft Windows, Red Hat Enterprise Linux on System x
x-MacThai	macthai	Solaris, SUSE Linux Enterprise Server on System x, Linux for System z, Microsoft Windows, Red Hat Enterprise Linux on System x
x-MacHebrew	machebrew	Solaris, SUSE Linux Enterprise Server on System x, Linux for System z, Microsoft Windows, Red Hat Enterprise Linux on System x
x-MacDingbat	macdingbat	Solaris, SUSE Linux Enterprise Server on System x, Linux for System z, Microsoft Windows, Red Hat Enterprise Linux on System x
x-KSC5601	ksc5601	Solaris, SUSE Linux Enterprise Server on System x, Linux for System z, Microsoft Windows, Red Hat Enterprise Linux on System x
x-JIS0208	jis_c6226-1983, jis_x0208-1983, csiso87jisx0208, x0208, iso-ir-87, jis0208	Solaris, SUSE Linux Enterprise Server on System x, Linux for System z, Microsoft Windows, Red Hat Enterprise Linux on System x

Encoding	Aliases	Platforms on which this encoding is supported
x-ISO-8859-6S	8859_6s, iso8859-6s, iso8859_6s, iso-8859-6s	SUSE Linux Enterprise Server on System x, Linux for System z, Microsoft Windows, Red Hat Enterprise Linux on System x
x-IBM954C	cp954c, 954c, ibm-954c, ibm954c	SUSE Linux Enterprise Server on System x, Linux for System z, Microsoft Windows, Red Hat Enterprise Linux on System x
x-IBM949C	ibm949c, cp949c, 949c, ibm-949c	Solaris, SUSE Linux Enterprise Server on System x, Linux for System z, Microsoft Windows, Red Hat Enterprise Linux on System x
x-IBM943C	cp943c, 943c, ibm-943c, ibm943c	Solaris, SUSE Linux Enterprise Server on System x, Linux for System z, Microsoft Windows, Red Hat Enterprise Linux on System x
x-IBM864S	csibm864s, ibm864s, cp864s, 864s, ibm-864s	SUSE Linux Enterprise Server on System x, Linux for System z, Microsoft Windows, Red Hat Enterprise Linux on System x
x-IBM420S	420s, ibm-420s, csibm420s, ibm420s, cp420s	SUSE Linux Enterprise Server on System x, Linux for System z, Microsoft Windows, Red Hat Enterprise Linux on System x
x-IBM1363C	ibm1363c, cp1363c, ibm-1363c	SUSE Linux Enterprise Server on System x, Linux for System z, Microsoft Windows, Red Hat Enterprise Linux on System x

Encoding	Aliases	Platforms on which this encoding is supported
x-IBM1046S	ibm-1046s, 1046s, cp1046s, ibm1046s	SUSE Linux Enterprise Server on System x, Linux for System z, Microsoft Windows, Red Hat Enterprise Linux on System x
x-IBM-udcJP	IBM-udcJP	SUSE Linux Enterprise Server on System x, Linux for System z, Microsoft Windows, Red Hat Enterprise Linux on System x
JIS_X0201	jis_x0201, x0201, cshalfwidthkatakana, jis0201	Solaris, SUSE Linux Enterprise Server on System x, Linux for System z, Microsoft Windows, Red Hat Enterprise Linux on System x
IBM-939A	Cp939A, ibm939A	SUSE Linux Enterprise Server on System x, Linux for System z, Microsoft Windows, Red Hat Enterprise Linux on System x
IBM-930A	ibm930A, Cp930A	SUSE Linux Enterprise Server on System x, Linux for System z, Microsoft Windows, Red Hat Enterprise Linux on System x,
IBM-33722A	Cp33722A, ibm33722A	SUSE Linux Enterprise Server on System x, Linux for System z, Microsoft Windows, Red Hat Enterprise Linux on System x
x-windows-iso2022jp	windows-iso2022jp	Solaris, SUSE Linux Enterprise Server on System x, HP-UX (Itanium), Linux for System z, Microsoft Windows, Red Hat Enterprise Linux on System x, HP (PA-RISC)

Encoding	Aliases	Platforms on which this encoding is supported
x-windows-50221	ms50221, cp50221	Solaris, SUSE Linux Enterprise Server on System x, HP-UX (Itanium), Linux for System z, Microsoft Windows, Red Hat Enterprise Linux on System x, HP (PA-RISC)
x-windows-50220	cp50220, ms50220	Solaris, SUSE Linux Enterprise Server on System x, HP-UX (Itanium), Linux for System z, Microsoft Windows, Red Hat Enterprise Linux on System x, HP (PA-RISC)
X-UTF-32LE-BOM	UTF_32LE_BOM, UTF-32LE-BOM	Solaris, SUSE Linux Enterprise Server on System x, Linux for System z, Microsoft Windows, Red Hat Enterprise Linux on System x, HP (PA-RISC)
X-UTF-32BE-BOM	UTF_32BE_BOM, UTF-32BE-BOM	Solaris, SUSE Linux Enterprise Server on System x, Linux for System z, Microsoft Windows, Red Hat Enterprise Linux on System x, HP (PA-RISC)
x-SJIS_0213		Solaris, SUSE Linux Enterprise Server on System x, Linux for System z, Microsoft Windows, Red Hat Enterprise Linux on System x, HP (PA-RISC)

Encoding	Aliases	Platforms on which this encoding is supported
x-PCK	pck	Solaris, SUSE Linux Enterprise Server on System x, HP-UX (Itanium), Linux for System z, Microsoft Windows, Red Hat Enterprise Linux on System x, HP (PA-RISC)
x-MS950-HKSCS	MS950_HKSCS	Solaris, SUSE Linux Enterprise Server on System x, HP-UX (Itanium), Linux for System z, Microsoft Windows, Red Hat Enterprise Linux on System x, HP (PA-RISC)
x-MS932_0213		Solaris, SUSE Linux Enterprise Server on System x, Linux for System z, Microsoft Windows, Red Hat Enterprise Linux on System x, HP (PA-RISC)
x-JISAutoDetect	JISAutoDetect	Solaris, SUSE Linux Enterprise Server on System x, HP-UX (Itanium), Linux for System z, Microsoft Windows, Red Hat Enterprise Linux on System x, HP (PA-RISC)

Encoding	Aliases	Platforms on which this encoding is supported
x-iso-8859-11	iso-8859-11, iso8859_11	Solaris, SUSE Linux Enterprise Server on System x, HP-UX (Itanium), Linux for System z, Microsoft Windows, Red Hat Enterprise Linux on System x, HP (PA-RISC)
x-ISO-2022-CN-CNS	ISO-2022-CN-CNS, ISO2022CN_CNS	Solaris, SUSE Linux Enterprise Server on System x, HP-UX (Itanium), Linux for System z, Microsoft Windows, Red Hat Enterprise Linux on System x, HP (PA-RISC)
x-IBM964	964, cp964, ibm-964, ibm964	Solaris, SUSE Linux Enterprise Server on System x, HP-UX (Itanium), Linux for System z, Microsoft Windows, Red Hat Enterprise Linux on System x, HP (PA-RISC)
x-IBM939	ibm-939, ibm939, cp939, 939	Solaris, SUSE Linux Enterprise Server on System x, HP-UX (Itanium), Linux for System z, Microsoft Windows, Red Hat Enterprise Linux on System x, HP (PA-RISC)

Encoding	Aliases	Platforms on which this encoding is supported
x-IBM930	cp930, ibm930, ibm-930, 930	Solaris, SUSE Linux Enterprise Server on System x, HP-UX (Itanium), Linux for System z, Microsoft Windows, Red Hat Enterprise Linux on System x, HP (PA-RISC)
x-IBM33722	ibm33722, 33722, ibm-33722_vascii_vpua, ibm-5050, ibm-33722, cp33722	Solaris, SUSE Linux Enterprise Server on System x, HP-UX (Itanium), Linux for System z, Microsoft Windows, Red Hat Enterprise Linux on System x, HP (PA-RISC)
x-eucJP-Open	EUC_JP_Solaris, eucJP-open	Solaris, SUSE Linux Enterprise Server on System x, HP-UX (Itanium), Linux for System z, Microsoft Windows, Red Hat Enterprise Linux on System x, HP (PA-RISC)
x-Big5-Solaris	Big5_Solaris	Solaris, SUSE Linux Enterprise Server on System x, HP-UX (Itanium), Linux for System z, Microsoft Windows, Red Hat Enterprise Linux on System x, HP (PA-RISC)
Roman9	Roman9	HP (PA-RISC)

Encoding	Aliases	Platforms on which this encoding is supported
ISO-2022-JP-2	csISO2022JP2, iso2022jp2	Solaris, SUSE Linux Enterprise Server on System x, Linux for System z, Microsoft Windows, Red Hat Enterprise Linux on System x, HP (PA-RISC)
IBM918	cp918, ebcdic-cp-ar2, ibm-918, 918	Solaris, SUSE Linux Enterprise Server on System x, HP-UX (Itanium), Linux for System z, Microsoft Windows, Red Hat Enterprise Linux on System x, HP (PA-RISC)
IBM1047	cp1047, 1047, ibm-1047	Solaris, SUSE Linux Enterprise Server on System x, HP-UX (Itanium), Linux for System z, Microsoft Windows, Red Hat Enterprise Linux on System x, HP (PA-RISC)
IBM01149	cp1149, cp01149, ccsid01149, 1149	Solaris, SUSE Linux Enterprise Server on System x, HP-UX (Itanium), Linux for System z, Microsoft Windows, Red Hat Enterprise Linux on System x, HP (PA-RISC)

Encoding	Aliases	Platforms on which this encoding is supported
IBM01148	cp1148, ccsid01148, 1148, cp01148	Solaris, SUSE Linux Enterprise Server on System x, HP-UX (Itanium), Linux for System z, Microsoft Windows, Red Hat Enterprise Linux on System x, HP (PA-RISC)
IBM01147	ccsid01147, cp1147, 1147, cp01147	Solaris, SUSE Linux Enterprise Server on System x, HP-UX (Itanium), Linux for System z, Microsoft Windows, Red Hat Enterprise Linux on System x, HP (PA-RISC)
IBM01146	ccsid01146, cp01146, cp1146, 1146	Solaris, SUSE Linux Enterprise Server on System x, HP-UX (Itanium), Linux for System z, Microsoft Windows, Red Hat Enterprise Linux on System x, HP (PA-RISC)
IBM01145	cp1145, cp01145, ccsid01145, 1145	Solaris, SUSE Linux Enterprise Server on System x, HP-UX (Itanium), Linux for System z, Microsoft Windows, Red Hat Enterprise Linux on System x, HP (PA-RISC)

Encoding	Aliases	Platforms on which this encoding is supported
IBM01144	cp01144, cp1144, ccsid01144, 1144	Solaris, SUSE Linux Enterprise Server on System x, HP-UX (Itanium), Linux for System z, Microsoft Windows, Red Hat Enterprise Linux on System x, HP (PA-RISC)
IBM01143	cp01143, 1143, ccsid01143, cp1143	Solaris, SUSE Linux Enterprise Server on System x, HP-UX (Itanium), Linux for System z, Microsoft Windows, Red Hat Enterprise Linux on System x, HP (PA-RISC)
IBM01142	cp01142, cp1142, 1142, ccsid01142	Solaris, SUSE Linux Enterprise Server on System x, HP-UX (Itanium), Linux for System z, Microsoft Windows, Red Hat Enterprise Linux on System x, HP (PA-RISC)
IBM01141	cp1141, ccsid01141, cp01141, 1141	Solaris, SUSE Linux Enterprise Server on System x, HP-UX (Itanium), Linux for System z, Microsoft Windows, Red Hat Enterprise Linux on System x, HP (PA-RISC)

Encoding	Aliases	Platforms on which this encoding is supported
IBM01140	ccsid01140, cp01140, 1140, cp1140	Solaris, SUSE Linux Enterprise Server on System x, HP-UX (Itanium), Linux for System z, Microsoft Windows, Red Hat Enterprise Linux on System x, HP (PA-RISC)
IBM00858	cp858, ccsid00858, 858, cp00858	Solaris, SUSE Linux Enterprise Server on System x, HP-UX (Itanium), Linux for System z, Microsoft Windows, Red Hat Enterprise Linux on System x, HP (PA-RISC)
X-UnicodeLittle	UnicodeLittle	Solaris, SUSE Linux Enterprise Server on System x, Linux for System z, AIX, Microsoft Windows, Red Hat Enterprise Linux on System x, Linux for System P , HP (PA-RISC)
X-UnicodeBig	UnicodeBig	Solaris, SUSE Linux Enterprise Server on System x, Linux for System z, AIX, Microsoft Windows, Red Hat Enterprise Linux on System x, Linux for System P , HP (PA-RISC)
IBM-864S	ibm864S, Cp864S	AIX, Linux for System P

Encoding	Aliases	Platforms on which this encoding is supported
IBM-859	Cp859, ibm859	SUSE Linux Enterprise Server on System x, Linux for System z, AIX, Microsoft Windows, Red Hat Enterprise Linux on System x, Linux for System P
IBM-858	Cp858, ibm858	AIX, Linux for System P
IBM-837	ibm837, Cp837	SUSE Linux Enterprise Server on System x, Linux for System z, AIX, Microsoft Windows, Red Hat Enterprise Linux on System x, Linux for System P
IBM-836	ibm836, Cp836	SUSE Linux Enterprise Server on System x, Linux for System z, AIX, Microsoft Windows, Red Hat Enterprise Linux on System x, Linux for System P
IBM-835	ibm835, Cp835	SUSE Linux Enterprise Server on System x, Linux for System z, AIX, Microsoft Windows, Red Hat Enterprise Linux on System x, Linux for System P
IBM-833	ibm833, Cp833	SUSE Linux Enterprise Server on System x, Linux for System z, AIX, Microsoft Windows, Red Hat Enterprise Linux on System x, Linux for System P

Encoding	Aliases	Platforms on which this encoding is supported
IBM-808	Cp808, ibm808	SUSE Linux Enterprise Server on System x, Linux for System z, AIX, Microsoft Windows, Red Hat Enterprise Linux on System x, Linux for System P
IBM-720	Cp720, ibm720	SUSE Linux Enterprise Server on System x, Linux for System z, AIX, Microsoft Windows, Red Hat Enterprise Linux on System x, Linux for System P
IBM-420S	Cp420S, ibm420S	AIX, Linux for System P
IBM-33722C	ibm-eucjp, Cp33722c	SUSE Linux Enterprise Server on System x, Linux for System z, AIX, Microsoft Windows, Red Hat Enterprise Linux on System x, Linux for System P
IBM-33722	5050, Cp5050	AIX, Linux for System P
IBM-301	Cp301, ibm301	SUSE Linux Enterprise Server on System x, Linux for System z, AIX, Microsoft Windows, Red Hat Enterprise Linux on System x, Linux for System P

Encoding	Aliases	Platforms on which this encoding is supported
IBM-300	Cp300, ibm300	SUSE Linux Enterprise Server on System x, Linux for System z, AIX, Microsoft Windows, Red Hat Enterprise Linux on System x, Linux for System P
IBM-290	ibm290, Cp290	SUSE Linux Enterprise Server on System x, Linux for System z, AIX, Microsoft Windows, Red Hat Enterprise Linux on System x, Linux for System P
IBM-1399	ibm1399, Cp1399	SUSE Linux Enterprise Server on System x, Linux for System z, AIX, Microsoft Windows, Red Hat Enterprise Linux on System x, Linux for System P
IBM-1390	Cp1390, ibm1390	SUSE Linux Enterprise Server on System x, Linux for System z, AIX, Microsoft Windows, Red Hat Enterprise Linux on System x, Linux for System P
IBM-1388	Cp1388, ibm1388	SUSE Linux Enterprise Server on System x, Linux for System z, AIX, Microsoft Windows, Red Hat Enterprise Linux on System x, Linux for System P
IBM-1386	ibm1386, Cp1386	AIX, Linux for System P

Encoding	Aliases	Platforms on which this encoding is supported
IBM-1385	Cp1385, ibm1385	SUSE Linux Enterprise Server on System x, Linux for System z, AIX, Microsoft Windows, Red Hat Enterprise Linux on System x, Linux for System P
IBM-1382	ibm1382, Cp1382	SUSE Linux Enterprise Server on System x, Linux for System z, AIX, Microsoft Windows, Red Hat Enterprise Linux on System x, Linux for System P
IBM-1088	Cp1088, ibm1088	SUSE Linux Enterprise Server on System x, Linux for System z, AIX, Microsoft Windows, Red Hat Enterprise Linux on System x, Linux for System P
IBM-1047	Cp1047, ibm1047	AIX, Linux for System P
IBM-1046S	ibm1046S, Cp1046S	AIX, Linux for System P
IBM-1043	Cp1043, ibm1043	SUSE Linux Enterprise Server on System x, Linux for System z, AIX, Microsoft Windows, Red Hat Enterprise Linux on System x, Linux for System P

Encoding	Aliases	Platforms on which this encoding is supported
IBM-1041	Cp1041, ibm1041	SUSE Linux Enterprise Server on System x, Linux for System z, AIX, Microsoft Windows, Red Hat Enterprise Linux on System x, Linux for System P
IBM-1027	Cp1027, ibm1027	SUSE Linux Enterprise Server on System x, Linux for System z, AIX, Microsoft Windows, Red Hat Enterprise Linux on System x, Linux for System P
CESU-8	CESU8	SUSE Linux Enterprise Server on System x, Linux for System z, AIX, Microsoft Windows, Red Hat Enterprise Linux on System x, Linux for System P
COMPOUND_TEXT	x-compound-text, x11-compound-text	Solaris, SUSE Linux Enterprise Server on System x, Linux for System z, AIX, Microsoft Windows, Red Hat Enterprise Linux on System x, Linux for System P , HP (PA-RISC)
GB2312	gb2312-1980, gb2312-80	SUSE Linux Enterprise Server on System x, Linux for System z, AIX, Microsoft Windows, Red Hat Enterprise Linux on System x, Linux for System P

Encoding	Aliases	Platforms on which this encoding is supported
GBK	GBK	SUSE Linux Enterprise Server on System x, Linux for System z, AIX, Microsoft Windows, Red Hat Enterprise Linux on System x, Linux for System P
hp-roman8	roman8, ibm-1051, r8, Cp1051	SUSE Linux Enterprise Server on System x, Linux for System z, AIX, Microsoft Windows, Red Hat Enterprise Linux on System x, Linux for System P , HP (PA-RISC)
IBM-1114	Cp1114, ibm1114	SUSE Linux Enterprise Server on System x, Linux for System z, AIX, Microsoft Windows, Red Hat Enterprise Linux on System x, Linux for System P
IBM-1115	Cp1115, ibm1115	SUSE Linux Enterprise Server on System x, Linux for System z, AIX, Microsoft Windows, Red Hat Enterprise Linux on System x, Linux for System P
IBM-1140	ibm1140, Cp1140	AIX, Linux for System P
IBM-1141	Cp1141, ibm1141	AIX, Linux for System P
IBM-1142	Cp1142, ibm1142	AIX, Linux for System P
IBM-1143	Cp1143, ibm1143	AIX, Linux for System P

Encoding	Aliases	Platforms on which this encoding is supported
IBM-1144	ibm1144, Cp1144	AIX, Linux for System P
IBM-1145	Cp1145, ibm1145	AIX, Linux for System P
IBM-1146	Cp1146, ibm1146	AIX, Linux for System P
IBM-1147	Cp1147, ibm1147	AIX, Linux for System P
IBM-1148	ibm1148, Cp1148	AIX, Linux for System P
IBM-1149	Cp1149, ibm1149	AIX, Linux for System P
IBM-1351	Cp1351, ibm1351	SUSE Linux Enterprise Server on System x, Linux for System z, AIX, Microsoft Windows, Red Hat Enterprise Linux on System x, Linux for System P
IBM-1362	Cp1362, ibm1362	SUSE Linux Enterprise Server on System x, Linux for System z, AIX, Microsoft Windows, Red Hat Enterprise Linux on System x, Linux for System P
IBM-1363	ibm1363, Cp1363	SUSE Linux Enterprise Server on System x, Linux for System z, AIX, Microsoft Windows, Red Hat Enterprise Linux on System x, Linux for System P
IBM-1363C	ibm1363C, Cp1363C	AIX, Linux for System P

Encoding	Aliases	Platforms on which this encoding is supported
IBM-1364	Cp1364, ibm1364	SUSE Linux Enterprise Server on System x, Linux for System z, AIX, Microsoft Windows, Red Hat Enterprise Linux on System x, Linux for System P , HP (PA-RISC)
IBM-1370	Cp1370, ibm1370	SUSE Linux Enterprise Server on System x, Linux for System z, AIX, Microsoft Windows, Red Hat Enterprise Linux on System x, Linux for System P
IBM-1371	Cp1371, ibm1371	SUSE Linux Enterprise Server on System x, Linux for System z, AIX, Microsoft Windows, Red Hat Enterprise Linux on System x, Linux for System P
IBM-1380	Cp1380, ibm1380	SUSE Linux Enterprise Server on System x, Linux for System z, AIX, Microsoft Windows, Red Hat Enterprise Linux on System x, Linux for System P
IBM-867	Cp867, ibm867	SUSE Linux Enterprise Server on System x, Linux for System z, AIX, Microsoft Windows, Red Hat Enterprise Linux on System x, Linux for System P

Encoding	Aliases	Platforms on which this encoding is supported
IBM-897	Cp897, ibm897	SUSE Linux Enterprise Server on System x, Linux for System z, AIX, Microsoft Windows, Red Hat Enterprise Linux on System x, Linux for System P
IBM-918	ibm918, Cp918	AIX, Linux for System P
IBM-924	Cp924, ibm924	SUSE Linux Enterprise Server on System x, Linux for System z, AIX, Microsoft Windows, Red Hat Enterprise Linux on System x, Linux for System P
IBM-927	ibm927, Cp927	SUSE Linux Enterprise Server on System x, Linux for System z, AIX, Microsoft Windows, Red Hat Enterprise Linux on System x, Linux for System P
IBM-930	Cp5026, 5026	AIX, Linux for System P
IBM-932	ibm932, Cp932	SUSE Linux Enterprise Server on System x, Linux for System z, AIX, Microsoft Windows, Red Hat Enterprise Linux on System x, Linux for System P
IBM-939	Cp5035, 5035	AIX, Linux for System P

Encoding	Aliases	Platforms on which this encoding is supported
IBM-942C	Cp942C, ibm942C	Solaris, HP-UX (Itanium), AIX, Linux for System P , HP (PA-RISC)
IBM-943C	ibm943C, Cp943C	HP-UX (Itanium), AIX, Linux for System P , HP (PA-RISC)
IBM-947	Cp947, ibm947	SUSE Linux Enterprise Server on System x, Linux for System z, AIX, Microsoft Windows, Red Hat Enterprise Linux on System x, Linux for System P
IBM-949C	Cp949C, ibm949C	HP-UX (Itanium), AIX, Linux for System P , HP (PA-RISC)
IBM-951	Cp951, ibm951	SUSE Linux Enterprise Server on System x, Linux for System z, AIX, Microsoft Windows, Red Hat Enterprise Linux on System x, Linux for System P
IBM-954	ibm954, Cp954	SUSE Linux Enterprise Server on System x, Linux for System z, AIX, Microsoft Windows, Red Hat Enterprise Linux on System x, Linux for System P
IBM-954C	Cp954c	AIX, Linux for System P

Encoding	Aliases	Platforms on which this encoding is supported
IBM-964	ibm-euctw, Cp964	AIX, Linux for System P
IBM-971	Cp971, ibm971	SUSE Linux Enterprise Server on System x, Linux for System z, AIX, Microsoft Windows, Red Hat Enterprise Linux on System x, Linux for System P
ISO-8859-10	latin6, 8859-10, ISO_8859-10:1992, iso8859_10, iso-ir-157, ibm-919, iso8859-10, l6, csisolatin6	SUSE Linux Enterprise Server on System x, Linux for System z, AIX, Microsoft Windows, Red Hat Enterprise Linux on System x, Linux for System P
ISO-8859-14	ISO_8859-14:1998, 8859-14, latin8, iso-ir-199, iso8859-14, l8, isoceltic, iso8859_14	SUSE Linux Enterprise Server on System x, Linux for System z, AIX, Microsoft Windows, Red Hat Enterprise Linux on System x, Linux for System P
ISO-8859-16	8859-16, iso8859_16, iso8859-16	SUSE Linux Enterprise Server on System x, Linux for System z, AIX, Microsoft Windows, Red Hat Enterprise Linux on System x, Linux for System P
ISO-8859-6S	iso8859-6S, iso8859_6S	AIX, Linux for System P
JIS0201		HP-UX (Itanium), AIX, Linux for System P , HP (PA-RISC)

Encoding	Aliases	Platforms on which this encoding is supported
JIS0208		HP-UX (Itanium), AIX, Linux for System P , HP (PA-RISC)
Johab	x-johab	AIX, Linux for System P
KOI8-RU	ibm-1168, koi8_ru	SUSE Linux Enterprise Server on System x, Linux for System z, AIX, Microsoft Windows, Red Hat Enterprise Linux on System x, Linux for System P
KOI8-U	koi8_u, ibm-1167	Solaris, AIX, Linux for System P , HP (PA-RISC)
KSC5601		AIX, Linux for System P
MacDingbat		HP-UX (Itanium), AIX, Linux for System P , HP (PA-RISC)
MacHebrew		HP-UX (Itanium), AIX, Linux for System P , HP (PA-RISC)
MacThai		HP-UX (Itanium), AIX, Linux for System P , HP (PA-RISC)
MacUkraine		HP-UX (Itanium), AIX, Linux for System P , HP (PA-RISC)

Encoding	Aliases	Platforms on which this encoding is supported
PTCP154	PT154, IBM-1169, Cyrillic-Asian, csPTCP154	SUSE Linux Enterprise Server on System x, Linux for System z, AIX, Microsoft Windows, Red Hat Enterprise Linux on System x, Linux for System P
Shift_JIS		AIX, Linux for System P
UTF-16	UTF16, Unicode, UTF_16, UCS-2	HP-UX (Itanium), AIX, Linux for System P
UTF-32	UCS-4, UTF32, ISO-10646-UCS-4	Solaris, SUSE Linux Enterprise Server on System x, Linux for System z, AIX, Microsoft Windows, Red Hat Enterprise Linux on System x, Linux for System P , HP (PA-RISC)
UTF-32BE	UTF_32BE, X-UTF-32BE, UTF32BE	Solaris, SUSE Linux Enterprise Server on System x, Linux for System z, AIX, Microsoft Windows, Red Hat Enterprise Linux on System x, Linux for System P , HP (PA-RISC)

Encoding	Aliases	Platforms on which this encoding is supported
UTF-32LE	UTF_32LE, X-UTF-32LE, UTF32LE	Solaris, SUSE Linux Enterprise Server on System x, Linux for System z, AIX, Microsoft Windows, Red Hat Enterprise Linux on System x, Linux for System P , HP (PA-RISC)
UTF-8J	UTF8J	SUSE Linux Enterprise Server on System x, Linux for System z, AIX, Microsoft Windows, Red Hat Enterprise Linux on System x, Linux for System P
windows-1256S	Cp1256s, ibm-1256s	SUSE Linux Enterprise Server on System x, Linux for System z, AIX, Microsoft Windows, Red Hat Enterprise Linux on System x, Linux for System P

Related concepts:

"Using transfer definition files" on page 206
You can specify a transfer definition file which can be used to create a file transfer. The transfer definition file is an XML file that defines some or all of the information required to create the transfer.

Related reference:

"Transferring text files" on page 680
Text file transfer involves converting the code page of a file from one code page to another. Text file transfer also involves converting CRLF (carriage return-line feed) characters between systems. This topic summarizes text file transfer behavior of WebSphere MQ Managed File Transfer.

"**fteCreateTransfer** (create new file transfer)" on page 461
The **fteCreateTransfer** command creates and starts a new file transfer from the command line. This command can start a file transfer immediately, schedule a file transfer for a future time and date, repeat a scheduled transfer one or more times, and trigger a file transfer based on certain conditions.

Message formats for WebSphere MQ Managed File Transfer

WebSphere MQ Managed File Transfer uses messages in XML format for a number of purposes: to interact with the Web Gateway; to command an agent; to log information about the monitors, schedules, and transfers; and to define information used for configuration. The logical structure of the XML formats used for these purposes described by XML schema.

Each version of WebSphere MQ Managed File Transfer uses an XML schema to validate messages written in XML. The agent extracts the XML schema version and determines whether the schema is supported.

After you have installed WebSphere MQ Managed File Transfer, you can find the WebSphere MQ Managed File Transfer message schema files in the following directory: `MQ_INSTALLATION_PATH`/mqft/ samples/schema. The following schemas are included:

Schemas for XML messages used by the Web Gateway
> `Filespace.xsd`
>
> `FileSpaceInfo.xsd`
>
> `UserInfo.xsd`
>
> `WebFileSpaceList.xsd`
>
> `WebTransferStatus.xsd`
>
> For more information about schemas used by the Web Gateway, see "Administration response and request formats" on page 889 and "Response formats: XML and JSON" on page 876.

Schemas for XML messages that can be put on an agent command queue
> `FileTransfer.xsd`
>
> `Internal.xsd`
>
> `Monitor.xsd`
>
> `PingAgent.xsd`
>
> For more information about putting XML messages on an agent command queue, see "Controlling WebSphere MQ Managed File Transfer by putting messages on the agent command queue" on page 342.

Schemas for XML messages that are published to the SYSTEM.FTE topic
> `MonitorList.xsd`
>
> `MonitorLog.xsd`
>
> `ScheduleList.xsd`
>
> `ScheduleLog.xsd`
>
> `TransferLog.xsd`
>
> `TransferStatus.xsd`
>
> For more information about XML messages that are published to the SYSTEM.FTE topic and the structure of the SYSTEM.FTE topic, see "The SYSTEM.FTE topic" on page 610.

Other schemas used by WebSphere MQ Managed File Transfer
> `fteutils.xsd`. This schema contains common element definitions and is included by some of the other schemas.
>
> `Notification.xsd`
>
> `ProtocolBridgeCredentials.xsd`
>
> `ProtocolBridgeProperties.xsd`
>
> `ConnectDirectCredentials.xsd`
>
> `ConnectDirectNodeProperties.xsd`
>
> `ConnectDirectProcessDefinitions.xsd`
>
> `Reply.xsd`
>
> `UserSandboxes.xsd`

Agent status message format

When an agent is created or started, the agent publishes its details to the SYSTEM.FTE topic on its coordination queue manager (on the SYSTEM.FTE/Agents/*agent name* topic).

The following information is included:
- Agent name
- Platform the agent is running on
- Agent description (if provided)
- Agent's queue manager
- Time zone that the agent is running in
- Agent version
- Agent transfer limits
- State of each of the agent's current transfers. These states are listed in Agent transfer states
- Type of agent

If the agent is a protocol bridge agent the following information is also included:
- Type of protocol bridge agent
- Host name or IP address of the protocol bridge server

If the agent is a web agent the following information is also included:
- Name of the Web Gateway the web agent connects to

The agent status is republished whenever the agent transfer states change, but by default no more than every 30 seconds. You can change this default setting using the agentStatusPublishRateLimit agent property, which is described in: Advanced agent properties.

The following example output shows the keys used for each data element in the agent status:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">
<properties>
    <entry key="agentOsName">Windows 7</entry>
    <entry key="agentDescription"/>
    <entry key="queueManager">QM1</entry>
    <entry key="agentTimeZone">Europe/London</entry>
    <entry key="agentVersion">1.00</entry>
    <entry key="agentName">FTEAGENT</entry>
    <entry key="maxDestinationTransfers">25</entry>
    <entry key="maxSourceTransfers">25</entry>
    <entry key="maxQueuedTransfers">100</entry>
    <entry key="DestinationTransferStates">414d51204d554e474f20202020202020d857374a60a72622=RunningTransfer
                                            414d51204d554e474f20202020202020d857374a69a72622=RunningTransfer
                                            414d51204d554e474f20202020202020d857374a75a72622=RunningTransfer
    </entry>
    <entry key="SourceTransferStates">414d51204d554e474f20202020202020d857374a93a72622=NegotiatingTransfer
                                      414d51204d554e474f20202020202020d857374a78a72622=RunningTransfer
                                      414d51204d554e474f20202020202020d857374aaba72622=NewSenderTransfer
                                      414d51204d554e474f20202020202020d857374a63a72622=RunningTransfer
    </entry>
</properties>
```

The following example output shows the keys used for each data element in the agent status of a protocol bridge agent:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">
<properties>
    <entry key="agentOsName">Windows 7</entry>
```

```
    <entry key="agentDescription"/>
    <entry key="queueManager">QM1</entry>
    <entry key="agentTimeZone">Europe/London</entry>
    <entry key="agentVersion">1.00</entry>
    <entry key="agentName">BRIDGE</entry>
    <entry key="protocolBridgeType">ftp</entry>
    <entry key="protocolBridgeServerHost">ftpserver.example.org</entry>
    <entry key="maxDestinationTransfers">25</entry>
    <entry key="maxSourceTransfers">25</entry>
    <entry key="maxQueuedTransfers">100</entry>
    <entry key="DestinationTransferStates">414d51204d554e474f20202020202020d857374a60a72622=RunningTransfer
    </entry>
    <entry key="SourceTransferStates">414d51204d554e474f20202020202020d857374a93a72622=NegotiatingTransfer
    </entry>
</properties>
```

Agent transfer states:

An agent that is started publishes its details to the SYSTEM.FTE topic on its coordination queue manager. These details include the states of each of the current transfers that involved that agent. The states are as follows:

Transfer state	Explanation
NewSenderTransfer	A new transfer from the source agent that the negotiation has not started for.
NewReceiverTransfer	A new transfer has been created at the destination agent as part of negotiation, but the transfer is not yet running.
NegotiatingTransfer	A source agent is in negotiation with the destination agent before running a transfer.
RunningTransfer	A transfer from either a source agent or destination agent that is in the normal running state
RecoveringTransfer	When either a source or destination agent starts the recovery process, any transfers in running state are moved into transfer state. Transfers are moved out of this state into ReSynchronisingTransfer state when a resynchronization message is sent to the peer agent. For example, if the destination agent starts the recovery process for a running transfer, the transfer is moved into the ReSynchronisingTransfer state when a resynchronization message is sent to its source agent.
ReSynchronisingTransfer	A transfer source or destination agent has found a problem and has sent a resynchronization message to its respective destination or source agent.
CompletedTransfer	A destination agent has completed the transfer and has sent a completion message to the source agent. The destination agent is waiting for an acknowledgment message from the source agent.
CompleteReceivedTransfer	A source agent has received a completion message from the destination agent and has sent a message back to the destination agent to acknowledge the completion message.
CancelledNewTransfer	A source agent has received a cancel message for a new transfer.
CancelledInProgressTransfer	A source agent has received a cancel message for an in-progress transfer.
ResumingTransfer	A source agent has received a resynchronize response message and now schedules the transfer to restart.
RestartingTransfer	A source or destination agent has received a resynchronize request message and is waiting for the respective destination or source agent to restart.

Transfer state	Explanation
WaitingForDestinationCapacity	A source agent has received a DESTINATION_CAPACITY_EXCEEDED error from the destination agent. The transfer is now in a waiting state to be retried after a period.
FailedTransferEnding	The transfer has failed but the completion log message has not been published and the transfer has not been removed from the state store. For example, this state can occur if an agent process is stopped after a failure response has been received from the destination agent but before the subsequent processing has been completed.

Monitor list message format

The XML messages that are published as retained publications to the topic string SYSTEM.FTE/monitors/*agent_name/monitor_name* conform to the MonitorList.xsd schema. Each XML message lists an active monitor belonging to that agent. This information is used by the **fteListMonitors** command and the WebSphere MQ Explorer plug-in to display a list of monitors to the user. The MonitorList.xsd schema document is located in the *MQ_INSTALLATION_PATH*/mqft/samples/schema directory. The MonitorList.xsd schema imports Monitor.xsd, which is in the same directory.

Schema

The following schema describes which elements are valid in a monitor list XML message.

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        targetNamespace="http://www.ibm.com/xmlns/wmqfte/7.0.1/MonitorDefinition"
        xmlns="http://www.ibm.com/xmlns/wmqfte/7.0.1/MonitorDefinition">

    <xsd:include schemaLocation="Monitor.xsd"/>

    <xsd:element name="monitorList">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="status"        type="monitorStatusType"        minOccurs="1" maxOccurs="1"/>
                <xsd:element name="configuration" type="monitorConfigurationType"  minOccurs="1" maxOccurs="1"/>
                <xsd:element name="pollInterval"  type="pollIntervalType"          minOccurs="1" maxOccurs="1"/>
                <xsd:element name="batch"         type="batchType"                 minOccurs="1" maxOccurs="1"/>
                <xsd:any minOccurs="0"/>
            </xsd:sequence>
            <xsd:attribute name="version" type="versionType" use="required"/>
            <xsd:attribute name="agent"   type="xsd:string"  use="required"/>
            <xsd:attribute name="monitor" type="xsd:string"  use="required"/>
        </xsd:complexType>
    </xsd:element>

    <xsd:complexType name="monitorStatusType">
        <xsd:sequence>
            <xsd:any minOccurs="0"/>
        </xsd:sequence>
        <xsd:attribute name="state"       type="xsd:token"/>
        <xsd:anyAttribute/>
    </xsd:complexType>

    <xsd:complexType name="monitorConfigurationType">
        <xsd:sequence>
            <xsd:element name="description" type="xsd:string"          minOccurs="1" maxOccurs="1" />
            <xsd:element name="resources"   type="monitorResourcesType" minOccurs="0" maxOccurs="1" />
            <xsd:element name="triggerMatch" type="triggerMatchType"     minOccurs="0" maxOccurs="1" />
            <xsd:element name="tasks"        type="monitorListTasksType" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
        <xsd:anyAttribute/>
    </xsd:complexType>

    <xsd:complexType name="monitorListTasksType">
        <xsd:sequence>
            <xsd:element name="task" type="monitorListTaskType"         minOccurs="1" maxOccurs="1" />
        </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="monitorListTaskType">
        <xsd:sequence>
            <xsd:element name="name"        type="monitorTaskNameType"  minOccurs="0" maxOccurs="1" />
            <xsd:element name="description" type="xsd:string"           minOccurs="0" maxOccurs="1" />
            <xsd:element name="taskXML"     type="xsd:string"           minOccurs="0" maxOccurs="1" />
```

```
            </xsd:sequence>
        </xsd:complexType>

</xsd:schema>
```

Understanding the monitor list message

The elements and attributes used in the monitor list messages are described in the following list:

`<monitorList>`

Group element containing the elements describe a monitor that is defined for the agent.

Attribute	Description
agent	Required. The name of the agent that the resource monitor is defined on.
monitor	Required. The name of the monitor. Unique for this agent.
version	Required. The version of the monitor list message format.

`<status>`

The status of the monitor.

Attribute	Description
state	The state of the monitor.

`<configuration>`

Group element containing the elements describe the configuration of the monitor.

> **`<description>`**
>
> A description of the monitor. (Not currently used.)

> **`<resources>`**
>
> The resource or resources being monitored.

> > **`<directory>`**
> >
> > A directory to monitor.

Attribute	Description
recursionLevel	The number of directory levels down from the top level to monitor.
id	The ID of the resource.

> > **`<queue>`**
> >
> > A queue to monitor.

Attribute	Description
id	The ID of the resource.

> **`<triggerMatch>`**
>
> Element that contains the <conditions> element.

> > **`<conditions>`**
> >
> > Element that contains the condition or conditions that the resource monitor is monitoring for. This element can contain only one of the following elements: <allOf>, <anyOf>, or <condition>.

<allOf>
> Element that contains the condition or conditions that the resource monitor is monitoring for. This element can contain one or many <condition> elements. For the resource monitor to be triggered all of the conditions inside of this element must be met.

<anyOf>
> Element that contains the condition or conditions that the resource monitor is monitoring for. This element can contain one or many <condition> elements. For the resource monitor to be triggered only one of the conditions inside of this element must be met.

<condition>
> Element that contains a single condition that the resource monitor is monitoring for. This element can contain only one of the following elements: <fileMatch>, <fileNoMatch>, <fileSize>, <queueNotEmpty>, <completeGroups>, or <fileSizeSame>. It can also contain a <name> element and a <resource> element.
>
> If the resource that is being monitored is a directory, one of the following three elements must be specified in the condition:
> * fileMatch
> * fileNoMatch
> * fileSize
>
> If the resource that is being monitored is a queue, one of the following two elements must be specified in the condition:
> * queueNotEmpty
> * completeGroups

<fileMatch>
> Group element for a file name match condition.

<pattern>
> Specifies a file name match pattern. Files on the resource must match the pattern in order to satisfy the condition. The default pattern is * (any file will match).

<fileNoMatch>
> Group element for an inverse file name match condition.

<pattern>
> Specifies an inverse file name match pattern. If no files on the monitored resource match, the condition is satisfied. The default pattern is * (the absence of any file will match).

<fileSize>
> Group element for a file size comparison.

<compare>
> Specifies a file size comparison. The value must be a non-negative integer.

Attribute	Description
operator	Comparison operator to use. Only >=' is supported.
units	Specifies file size units, which can be one of: • B - bytes • KB - kilobytes • MB - megabytes • GB - gigabytes The units value is case insensitive, so mb' works as well as MB'.

\<pattern\>

File name pattern to match. Default is * (any file will match).

\<queueNotEmpty\>

This can only be specified if the resource is a queue. Specifies that there must be a message on the queue for the monitor to be triggered.

\<completeGroups\>

This can only be specified if the resource is a queue. Specifies that there must be a complete group of messages present on the queue for the monitor to be triggered. A single transfer task is executed for each complete group on the queue.

\<name\>

Name of the condition.

\<resource\>

Identifies the resource definition to compare the condition against.

Attribute	Description
id	Unique identifier for the resource.

\<tasks\>

Group element to contain elements which specify the tasks to invoke when the monitor trigger conditions are satisfied.

\<task\>

Group element which defines an individual task that the monitor will invoke when the trigger conditions are satisfied. Currently only one task can be specified.

\<name\>

Name of the task. Accepts any alphanumeric characters.

\<description\>

Description of the task. Any text value is allowed.

\<taskXML\>

The XML message that describes the task that the monitor is to perform. The contents of this element are in an escaped XML format.

\<pollInterval\>

The time interval between each check of the resource against the trigger condition.

Attribute	Description
units	Specifies the time units for the poll interval. Valid values are: • seconds • minutes • hours • days • weeks • months • years

\<batch\>

The maximum number of trigger matches to include in a single batch.

Attribute	Description
maxSize	The maximum number of trigger matches to include in a single batch

The following XML shows an example of a retained publication which is published to the topic string SYSTEM.FTE/monitors/*agent_name*/MONITORTWO when the monitor called MONITORTWO is created on AGENT_JUPITER. The escaped XML within the <taskXML> element describes the task that is submitted when the monitor condition is met.

```
<?xml version="1.0" encoding="UTF-8"?>
<lst:monitorList xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                 xmlns:lst="http://www.ibm.com/xmlns/wmqfte/7.0.1/MonitorDefinition"
                 xsi:schemaLocation="http://www.ibm.com/xmlns/wmqfte/7.0.1/MonitorDefinition MonitorList.xsd"
                 version="4.00"
                 agent="AGENT_JUPITER"
                 monitor="MONITORTWO">
    <status state="started"/>
    <configuration>
        <description/>
        <resources>
            <directory recursionLevel="0" id="">/srv/nfs/incoming</directory>
        </resources>
        <triggerMatch>
            <conditions>
                <condition>
                    <name/>
                    <resource id=""/>
                    <fileMatch>
                        <pattern>*.completed</pattern>
                    </fileMatch>
                </condition>
            </conditions>
        </triggerMatch>
        <tasks>
            <task>
                <name/>
                <description/>
                <taskXML>&lt;?xml version="1.0" encoding="UTF-8"?&gt;&lt;request
                        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.00"
                        xsi:noNamespaceSchemaLocation="FileTransfer.xsd"&gt;&lt;managedTransfer&gt;
                        &lt;originator&gt;&lt;hostName&gt;example.com.&lt;/hostName&gt;
                        &lt;userID&gt;mqm&lt;/userID&gt;&lt;/originator&gt;
                        &lt;sourceAgent QMgr="QM_JUPITER" agent="AGENT_JUPITER"/&gt;
                        &lt;destinationAgent QMgr="QM_JUPITER" agent="AGENT_SATURN"/&gt;
                        &lt;transferSet&gt;&lt;item checksumMethod="MD5" mode="binary"&gt;
                        &lt;source disposition="leave" recursive="false"&gt;&lt;file
                        &gt;/srv/nfs/incoming/*.txt&lt;/file&gt;&lt;/source&gt;
                        &lt;destination exist="error" type="directory"&gt;
                        &lt;file&gt;/srv/backup&lt;/file&gt;&lt;/destination&gt;
                        &lt;/item&gt;&lt;/transferSet&gt;&lt;/managedTransfer&gt;
                        &lt;/request&gt;
                </taskXML>
            </task>
        </tasks>
    </configuration>
    <pollInterval units="minutes">1</pollInterval>
    <batch maxSize="1"/>
</lst:monitorList>
```

Schedule list message format

The XML message that is published to a retained publication to the topic string SYSTEM.FTE/Scheduler/ *agent_name* conforms to the ScheduleList.xsd schema. This XML message lists all active schedules belonging to that agent. This information is used by the **fteListScheduledTransfers** command and the WebSphere MQ Explorer plug-in to display a list of schedules to the user. The ScheduleList.xsd schema document is located in the *MQ_INSTALLATION_PATH*/mqft/samples/schema directory. The ScheduleList.xsd schema imports FileTransfer.xsd, which is in the same directory.

Schema

The following schema describes which elements are valid in a monitor list XML message.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

    <xsd:include schemaLocation="FileTransfer.xsd"/>

    <xsd:element name="schedules">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="managedTransfer" type="scheduledManagedTransferType" minOccurs="0" maxOccurs="unbounded"/>
            </xsd:sequence>
            <xsd:attribute name="version" type="versionType"           use="required"/>
            <xsd:attribute name="size"    type="xsd:nonNegativeInteger" use="required"/>
```

```
            <xsd:attribute name="agent"    type="xsd:string"              use="required"/>
        </xsd:complexType>
    </xsd:element>

    <xsd:complexType name="scheduledManagedTransferType">
        <xsd:sequence>
            <xsd:element name="originator"       type="origRequestType"  maxOccurs="1" minOccurs="1"/>
            <xsd:element name="schedule"         type="scheduleListType" maxOccurs="1" minOccurs="0"/>
            <xsd:element name="sourceAgent"      type="agentType"        maxOccurs="1" minOccurs="1"/>
            <xsd:element name="destinationAgent" type="agentClientType"  maxOccurs="1" minOccurs="1"/>
            <xsd:element name="trigger"          type="triggerType"      maxOccurs="1" minOccurs="0"/>
            <xsd:element name="reply"            type="replyType"        maxOccurs="1" minOccurs="0"/>
            <xsd:element name="transferSet"      type="transferSetType"  maxOccurs="1" minOccurs="1"/>
            <xsd:element name="job"              type="jobType"          maxOccurs="1" minOccurs="0" />
        </xsd:sequence>
        <xsd:attribute name="id" type="idType" use="required"/>
    </xsd:complexType>

    <xsd:complexType name="scheduleListType">
        <xsd:sequence>
            <xsd:element name="submit" type="submitType"     maxOccurs="1" minOccurs="1"/>
            <xsd:element name="repeat" type="repeatType"      maxOccurs="1" minOccurs="0"/>
            <xsd:element name="next"   type="noZoneTimeType"  maxOccurs="1" minOccurs="1"/>
        </xsd:sequence>
    </xsd:complexType>

</xsd:schema>
```

Understanding the schedule list message

The elements and attributes used in the schedule list messages are described in the following list:

`<schedules>`
Group element containing information about all of the schedules defined on a single agent.

Attribute	Description
agent	Required. The name of the source agent that the schedule is defined on.
size	Required. The number of schedules defined on this agent.
version	Required. The version of the schedule list message format.

`<managedTransfer>`
Group element containing information about a single schedule.

Attribute	Description
id	Required. The hexadecimal string ID of the schedule request message.

`<originator>`
The originator of the schedule request.

> **`<hostName>`**
> The host name of the machine that the schedule request was submitted from.

> **`<userID>`**
> The user ID of the user that submitted the schedule request.

> **`<mqmdUserID>`**
> The MQMD user ID of the user that submitted the schedule request.

> **`<webBrowser>`**
> If the schedule request was submitted through the Web Gateway, the web browser that the
> request was submitted from.

`<webUserID>`
> If the schedule request was submitted through the Web Gateway, the web user ID of the user that submitted the schedule request.

`<schedule>`
> Element that contains the elements that describe when the scheduled transfer occurs.

> **`<submit>`**
>> Specifies the date and time that the scheduled transfer is due to start.

Attribute	Description
timebase	Specifies which time zone to use. The value of this attribute can be one of the following values: • source - use the time zone of the source agent • admin - use the time zone of the administrator issuing the command • UTC - use Coordinated Universal Time
timezone	The time zone description according to the timebase value

> **`<repeat>`**
>> Group element that contains details about how often a scheduled transfer repeats, how many times a scheduled transfer repeats, and when a scheduled transfer stops repeating.

Attribute	Description
interval	The interval units, which must be one of the following values: • minutes • hours • days • weeks • months • years

> **`<frequency>`**
>> The time period that must elapse before the transfer repeats.

Attribute	Description
interval	The interval units, which must be one of the following values: • minutes • hours • days • weeks • months • years

> **`<expireTime>`**
>> Optional element that specifies the date and time that a repeating scheduled transfer stops. This element and the <expireCount> element are mutually exclusive.

> **`<expireCount>`**
>> Optional element that specifies the number of times the scheduled file transfer occurs before stopping. This element and the <expireTime> element are mutually exclusive.

`<next>`

    Specifies the date and time when the next scheduled transfer is due to start.

`<sourceAgent>`

    Specifies the name of the agent on the system where the source file is located.

Attribute	Description
agent	Specifies the name of the agent.
QMgr	The name of the agent queue manager.

`<destinationAgent>`

    Specifies the name of the agent on the system you want to transfer the file to.

Attribute	Description
agent	Specifies the name of the agent.
QMgr	The name of the agent queue manager.

`<trigger>`

    Optional element that specifies a condition that must be true for the file transfer to take place.

Attribute	Description
log	A flag indicating whether trigger failures are logged. The following are valid values: • yes - log entries are created for failed triggered transfers • no - log entries are not created for failed triggered transfers

`<reply>`

    Specifies the name of the temporary reply queue generated for synchronous file transfers (specified with the **-w** parameter on the command line). The name of the queue is defined by the key **dynamicQueuePrefix** in the `command.properties` configuration file or the default of WMQFTE.* if not specified.

Attribute	Description
QMGR	The name of the command queue manager on which the temporary dynamic queue is generated to receive replies.

`<transferSet>`

    Specifies a group of file transfers you want the scheduled transfer to perform together. During transmission <transferSet> is a group element containing <item> elements.

Attribute	Description
priority	Priority level of the transfer. Priority is a value in the range 0-9, where 0 is the lowest priority. The default priority level is 0 and by default the transfer uses the priority level of the source agent.

`<job>`

    Optional group element containing job information for the entire transfer specification. <job> is a user-defined job name identifier that is added to the log message when the transfer has started. This <job> element is the same as the <job> element that appears in the transfer log message, which is described in the following topic: "File transfer log message formats" on page 626.

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<schedules xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        size="2"
        version="4.00"
        agent="AGENT_JUPITER"
        xsi:noNamespaceSchemaLocation="ScheduleList.xsd">
    <managedTransfer id="1">
        <originator>
            <hostName>example.com.</hostName>
            <userID>mqm</userID>
        </originator>
        <schedule>
            <submit timebase="admin" timezone="Europe/London">2010-01-01T21:00+0000</submit>
            <next>2010-01-01T21:00+0000</next>
        </schedule>
        <sourceAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER"/>
        <destinationAgent agent="AGENT_SATURN" QMgr="QM_JUPITER"/>
        <reply QMGR="QM_JUPITER">WMQFTE.4D400F8B20004E06</reply>
        <transferSet>
            <item mode="binary" checksumMethod="MD5">
                <source recursive="false" disposition="leave">
                    <file>/etc/passwd</file>
                </source>
                <destination type="directory" exist="overwrite">
                    <file>/tmp</file>
                </destination>
            </item>
        </transferSet>
    </managedTransfer>
    <managedTransfer id="2">
        <originator>
            <hostName>example.com.</hostName>
            <userID>mqm</userID>
        </originator>
        <schedule>
            <submit timebase="admin" timezone="Europe/London">2010-12-31T09:00+0000</submit>
            <next>2010-12-31T09:00+0000</next>
        </schedule>
        <sourceAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER"/>
        <destinationAgent agent="AGENT_NEPTUNE" QMgr="QM_JUPITER"/>
        <reply QMGR="QM_JUPITER">WMQFTE.4D400F8B20004E09</reply>
        <transferSet>
            <item mode="binary" checksumMethod="MD5">
                <source recursive="false" disposition="leave">
                    <file>/etc/passwd</file>
                </source>
                <destination type="directory" exist="overwrite">
                    <file>/tmp</file>
                </destination>
            </item>
        </transferSet>
    </managedTransfer>
</schedules>
```

Example template XML message

When a template is created, a message is published to the SYSTEM.FTE topic with a topic string of Templates/*template_ID*. This example XML describes a single template defined in your WebSphere MQ Managed File Transfer network.

```
<?xml version="1.0" encoding="UTF-8"?>
<transferTemplate version="4.00" id="baf9df73-45c2-4bb0-a085-292232ab66bc">
    <name>BASIC_TEMPLATE</name>
    <sourceAgentName>AGENT_JUPITER</sourceAgentName>
    <sourceAgentQMgr>QM_JUPITER</sourceAgentQMgr>
    <destinationAgentName>AGENT_SATURN</destinationAgentName>
    <destinationAgentQMgr>QM_JUPITER</destinationAgentQMgr>
    <fileSpecs>
        <item mode="binary" checksumMethod="MD5">
            <source recursive="false" disposition="leave">
                <file>/etc/passwd</file>
            </source>
            <destination type="directory" exist="overwrite">
                <file>/tmp</file>
            </destination>
        </item>
    </fileSpecs>
    <priority>0</priority>
</transferTemplate>
```

Related tasks:

"Creating a file transfer template using the WebSphere MQ Explorer" on page 233
You can create a file transfer template from the WebSphere MQ Explorer or from the command line. You can then use that template to create new file transfers using the template details or submit the template to start the file transfer.

Related reference:

"**fteCreateTemplate** (create new file transfer template)" on page 449
The **fteCreateTemplate** command creates a file transfer template that you can keep for future use. The only required parameter is the **-tn** (*template_name*) parameter. All other parameters are optional, although if you specify a source file specification, you must also provide a destination file. Similarly, if you specify a destination file, you must also specify a source file specification.

File transfer status message format

Messages are published to the coordination queue manager to indicate transfer status of each file in the transfer set. Every time a request for file transfer is processed by the agent, a transaction message is published to the coordination queue manager (on its SYSTEM.FTE/Transfers/*agent_name/transfer ID* topic), which conforms to the TransferStatus.xsd XML schema. The TransferStatus.xsd file is located in the *MQ_INSTALLATION_PATH*/mqft/samples/schema directory of your WMQMFT installation.

Schema

The following schema describes which elements are valid in a transfer status XML message.

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

    <xsd:include schemaLocation="fteutils.xsd"/>

    <xsd:element name="transaction">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="sourceAgent"      type="agentType"
                             maxOccurs="1"           minOccurs="1"/>
                <xsd:element name="destinationAgent" type="agentType"
                             maxOccurs="1"           minOccurs="1"/>
                <xsd:element name="transferSet"      type="transferSetType"
                             maxOccurs="1"           minOccurs="1"/>
            </xsd:sequence>
            <xsd:attribute name="version" type="versionType"   use="required"/>
```

```
            <xsd:attribute name="ID"        type="IDType"        use="required"/>
        </xsd:complexType>
    </xsd:element>

    <xsd:complexType name="transferSetType">
        <xsd:sequence>
            <xsd:element name="stats"    type="statsType"
                        maxOccurs="1" minOccurs="1" />
            <xsd:element name="current" type="currentType"
                        maxOccurs="1" minOccurs="0" />
        </xsd:sequence>
        <xsd:attribute name="time" type="xsd:dateTime" use="required" />
    </xsd:complexType>

    <xsd:complexType name="currentType">
        <xsd:sequence>
            <xsd:element name="source"      type="fileSourceType"
                        maxOccurs="1"      minOccurs="1" />
            <xsd:element name="destination" type="fileDestinationType"
                        maxOccurs="1"      minOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="transferred" type="xsd:nonNegativeInteger" use="required" />
        <xsd:attribute name="size" type="xsd:nonNegativeInteger"  use="required" />
    </xsd:complexType>

    <xsd:complexType name="statsType">
        <xsd:attribute name="bytes"        type="xsd:nonNegativeInteger"  use="required" />
        <xsd:attribute name="seconds"      type="xsd:decimal"             use="required" />
        <xsd:attribute name="currentItem"  type="xsd:nonNegativeInteger"  use="required" />
        <xsd:attribute name="totalItems"   type="xsd:nonNegativeInteger"  use="required" />
    </xsd:complexType>

</xsd:schema>
```

Understanding the transfer status message

The elements and attributes used in the transfer status messages are described in the following list:

`<transaction>`
  Group element that contains all of the elements for the file transfers.

Attribute	Description
version	Specifies the version of this element as supplied by WebSphere MQ Managed File Transfer.
ID	The unique identifier for the file transfer.

`<sourceAgent>`
  Specifies the name of the agent on the system where the source file is located.

Attribute	Description
agent	The name of the agent.
QMgr	The name of the agent queue manager.

`<destinationAgent>`
  Specifies the name of the agent on the system you want to transfer the file to.

Attribute	Description
agent	The name of the agent.
QMgr	The name of the agent queue manager.

\<transferset\>

Specifies a group of file transfers being performed together. All of the files in the transfer must originate at the same source agent and end at the same destination agent.

Attribute	Description
time	Specifies the date and time (in date time format).

\<stats\>

Required. Defines metrics about the transfer, including the number of bytes copied so far, in the given number of seconds. Also supplies the current item number out of the total number of items in the \<transferSet\>.

Attribute	Description
bytes	Number of bytes copied so far.
seconds	Number of seconds taken to transfer those bytes.
currentItem	The index of the current item being transferred.
totalItems	The total number of items being transferred.

\<current\>

Optional element. Group element that contains elements that specify the file transfer currently in progress. The \<current\> element indicates how many bytes of data have been transferred so far for the current item and the expected total number of bytes

\<source\>

Group element that contains the element specifying the source file name.

\<file\>

Specifies the source path of the file that is being transferred. The path is as specified for the transfer. This path might differ from the path that is output as part of the transfer log, which is the absolute form the of path.

\<destination\>

Group element that contains the element specifying the destination file name or specification.

\<file\>

Specifies the destination path of the file that is being transferred. The path is as specified for the transfer. This path might differ from the path that is output as part of the transfer log, which is the absolute form the of path.

Attribute	Description
alias	Specifies an alias for the destination file. This alias is the name of the source file, excluding any directory path specified for the transfer.
filespace	Specifies the name of the file space where the destination file is written.

\<queue\>

When used with the \<destination\> element, specifies the name of the queue you want to transfer to. This name is in the format QUEUE or QUEUE@QUEUE_MANAGER.

Transfer progress message examples:

When a transfer is in progress, messages are published to the SYSTEM.FTE topic with a topic string of Transfers/*agent_name*/*transfer_ID*. The XML examples show the progress message for a single file transfer and for a multiple file transfer.

Single file transfer

The following example shows the details of a single file transfer that is in progress.

```
<?xml version="1.0" encoding="UTF-8"?>
<transaction xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             version="4.00"
             ID="414d51205553322e42494e44494e47538b0f404d223d0020"
             xsi:noNamespaceSchemaLocation="TransferStatus.xsd">
        <sourceAgent agent="US2.BINDINGS.FILE" QMgr="US2.BINDINGS"/>
        <destinationAgent agent="US2.BINDINGS.FILE" QMgr="US2.BINDINGS"/>
        <transferSet time="2011-01-26T13:03:26.542Z">
        <stats bytes="1198" seconds="0.018" currentItem="1" totalItems="1"/>
        <current transferred="1151" size="1151">
            <source>
                <file>/etc/passwd</file>
            </source>
            <destination>
                <file>/tmp/passwd</file>
            </destination>
        </current>
    </transferSet>
</transaction>
```

Multiple file transfer

If there were more files in the transfer set, the transfer status message indicates which one is being processed and how many bytes have been transferred so far.

```
<?xml version="1.0" encoding="UTF-8"?>
<transaction xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             version="4.00"
             ID="414d51205553322e42494e44494e47538b0f404d035c0020"
             xsi:noNamespaceSchemaLocation="TransferStatus.xsd">
    <sourceAgent agent="US2.BINDINGS.FILE" QMgr="US2.BINDINGS"/>
    <destinationAgent agent="US2.BINDINGS.FILE" QMgr="US2.BINDINGS"/>
    <transferSet time="2011-01-26T13:12:58.636Z">
        <stats bytes="440" seconds="0.082" currentItem="10" totalItems="10"/>
        <current transferred="0" size="0">
            <source>
                <file>/srv/nfs/incoming/file10.txt</file>
            </source>
            <destination>
                <file>/srv/nfs/outgoing/file10.txt</file>
            </destination>
        </current>
    </transferSet>
</transaction>
```

File transfer log message formats

File transfer log messages are published to the SYSTEM.FTE topic with a topic string of Log/*agent_name/transfer_ID*. These messages conform to the schema `TransferLog.xsd`, which is located in the `MQ_INSTALLATION_PATH`/mqft/samples/schema directory of your WebSphere MQ Managed File Transfer installation.

If you want to monitor file transfers or collect data about them, set up a subscription to a wildcard topic tailored to the transfers you are interested in. For example:

```
Log/#
```

or,

```
Log/FTEAGENT/#
```

This subscription can be durable or non-durable. Durable subscriptions continue to exist when a subscribing application's connection to the queue manager is closed. Non-durable subscriptions exist only as long as a subscribing application's connection to the queue manager remains open.

Schema

The following schema describes which elements are valid in a transfer log XML message.

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

    <xsd:include schemaLocation="fteutils.xsd"/>

    <xsd:element name="transaction">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="action"                type="actionType"
                             maxOccurs="1"                minOccurs="0"/>
                <xsd:element name="sourceAgent"           type="agentExitStatusType"
                             maxOccurs="1"                minOccurs="0"/>
                <xsd:element name="sourceWebGateway"      type="webGatewayType"
                             maxOccurs="1"                minOccurs="0"/>
                <xsd:element name="sourceWebUser"         type="webUserType"
                             maxOccurs="1"                minOccurs="0"/>
                <xsd:element name="destinationAgent"      type="agentExitStatusType"
                             maxOccurs="1"                minOccurs="0"/>
                <xsd:element name="destinationWebGateway" type="webGatewayType"
                             maxOccurs="1"                minOccurs="0"/>
                <xsd:element name="destinationWebUser"    type="webUserType"
                             maxOccurs="1"                minOccurs="0"/>
                <xsd:element name="agent"                 type="agentExitStatusType"
                             maxOccurs="1"                minOccurs="0"/>
                <xsd:element name="originator"            type="origRequestType"
                             maxOccurs="1"                minOccurs="1"/>
                <xsd:element name="status"                type="statusType"
                             maxOccurs="1"                minOccurs="0"/>
                <xsd:element name="trigger"               type="triggerType"
                             maxOccurs="1"                minOccurs="0" />
                <xsd:element name="transferSet"           type="transferSetType"
                             maxOccurs="1"                minOccurs="1"/>
                <xsd:element name="job"                   type="jobType"
                             maxOccurs="1"                minOccurs="0"/>
                <xsd:element name="scheduleLog"           type="scheduleLogType"
                             maxOccurs="1"                minOccurs="0"/>
                <xsd:element name="statistics"            type="statisticsType"
                             maxOccurs="1"                minOccurs="0"/>
            </xsd:sequence>
            <xsd:attribute name="version"   type="versionType"   use="required"/>
            <xsd:attribute name="ID"        type="IDType"        use="required"/>
            <xsd:attribute name="relatedID" type="IDType"        use="optional"/>
            <xsd:attribute name="agentRole" type="agentRoleType" use="optional"/>
        </xsd:complexType>
    </xsd:element>

    <xsd:complexType name="agentExitStatusType">
        <xsd:complexContent>
            <xsd:extension base="agentType">
                <xsd:sequence>
```

```
                        <xsd:element name="startExits" type="exitGroupType" minOccurs="0" maxOccurs="1"/>
                        <xsd:element name="endExits"   type="exitGroupType" minOccurs="0" maxOccurs="1"/>
                        <xsd:element name="systemInfo" type="systemInfoType" minOccurs="0" maxOccurs="1"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="transferSetType">
    <xsd:sequence>
        <xsd:element name="metaDataSet"        type="metaDataSetType"
                     maxOccurs="1"             minOccurs="0" />
        <xsd:element name="call"               type="callGroupType"
                     maxOccurs="1"             minOccurs="0"/>
        <xsd:element name="preSourceCall"      type="callGroupType"
                     maxOccurs="1"             minOccurs="0"/>
        <xsd:element name="postSourceCall"     type="callGroupType"
                     maxOccurs="1"             minOccurs="0"/>
        <xsd:element name="preDestinationCall" type="callGroupType"
                     maxOccurs="1"             minOccurs="0"/>
        <xsd:element name="postDestinationCall" type="callGroupType"
                     maxOccurs="1"             minOccurs="0"/>
        <xsd:element name="item"               type="itemType"
                     maxOccurs="unbounded"     minOccurs="0"/>
    </xsd:sequence>
    <xsd:attribute name="index"     type="xsd:nonNegativeInteger" use="optional" />
    <xsd:attribute name="size"      type="xsd:nonNegativeInteger" use="optional" />
    <xsd:attribute name="startTime" type="xsd:dateTime"           use="required" />
    <xsd:attribute name="total"     type="xsd:nonNegativeInteger" use="required" />
    <xsd:attribute name="bytesSent" type="xsd:nonNegativeInteger" use="required" />
</xsd:complexType>

<xsd:complexType name="itemType">
    <xsd:sequence>
        <xsd:element name="source"      type="fileSourceChecksumType"
                     maxOccurs="1"      minOccurs="1" />
        <xsd:element name="destination" type="fileDestinationChecksumType"
                     maxOccurs="1"      minOccurs="1" />
        <xsd:element name="status"      type="statusType"
                     maxOccurs="1"      minOccurs="1" />
    </xsd:sequence>
    <xsd:attribute name="mode" type="modeType" use="required" />
</xsd:complexType>

<xsd:complexType name="fileSourceChecksumType">
    <xsd:complexContent>
        <xsd:extension base="fileSourceType">
            <xsd:sequence>
                <xsd:element name="checksum" type="checksumType" minOccurs="0" maxOccurs="1"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="fileDestinationChecksumType">
    <xsd:complexContent>
        <xsd:extension base="fileDestinationType">
            <xsd:sequence>
                <xsd:element name="checksum" type="checksumType"
                             minOccurs="0"    maxOccurs="1"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="actionType">
    <xsd:simpleContent>
        <xsd:extension base="actionEnumType">
            <xsd:attribute name="time" type="xsd:dateTime" use="required" />
        </xsd:extension>
    </xsd:simpleContent>
</xsd:complexType>

<xsd:simpleType name="actionEnumType">
    <xsd:restriction base="xsd:token">
        <xsd:enumeration value="cancelled"/>
        <xsd:enumeration value="started"/>
        <xsd:enumeration value="progress"/>
        <xsd:enumeration value="completed"/>
```

```
            <xsd:enumeration value="malformed"/>
            <xsd:enumeration value="notAuthorized"/>
            <xsd:enumeration value="deleted"/>
        </xsd:restriction>
    </xsd:simpleType>

    <xsd:complexType name="systemInfoType">
        <xsd:attribute name="architecture" type="xsd:string" use="required"/>
        <xsd:attribute name="name"         type="xsd:string" use="required"/>
        <xsd:attribute name="version"      type="xsd:string" use="required"/>
    </xsd:complexType>

    <xsd:element name="malformed">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="action"  type="actionType"
                            maxOccurs="1"  minOccurs="1"/>
                <xsd:element name="agent"   type="agentExitStatusType"
                            maxOccurs="1"  minOccurs="0"/>
                <xsd:element name="status"  type="statusType"
                            maxOccurs="1"  minOccurs="1"/>
            </xsd:sequence>
            <xsd:attribute name="version"   type="versionType"    use="required"/>
            <xsd:attribute name="ID"        type="IDType"         use="required"/>
            <xsd:attribute name="agentRole" type="agentRoleType"  use="required"/>
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="notAuthorized">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="action"      type="actionType"
                            maxOccurs="1"      minOccurs="1"/>
                <xsd:element name="originator"  type="origRequestType"
                            maxOccurs="1"      minOccurs="1"/>
                <xsd:element name="authority"   type="xsd:string"
                            minOccurs="1"      maxOccurs="1"/>
                <xsd:element name="status"      type="statusType"
                            maxOccurs="1"      minOccurs="1"/>
            </xsd:sequence>
            <xsd:attribute name="version"   type="versionType"    use="required"/>
            <xsd:attribute name="ID"        type="IDType"         use="required"/>
            <xsd:attribute name="agentRole" type="agentRoleType"  use="required"/>
        </xsd:complexType>
    </xsd:element>

    <xsd:complexType name="statisticsType">
        <xsd:sequence>
            <xsd:element name="actualStartTime" type="xsd:dateTime"
                        maxOccurs="1"          minOccurs="0"/>
            <xsd:element name="retryCount"      type="xsd:nonNegativeInteger"
                        maxOccurs="1"          minOccurs="1"/>
            <xsd:element name="numFileFailures" type="xsd:nonNegativeInteger"
                        maxOccurs="1"          minOccurs="1"/>
            <xsd:element name="numFileWarnings" type="xsd:nonNegativeInteger"
                        maxOccurs="1"          minOccurs="1"/>
        </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="webGatewayType">
        <xsd:attribute name="webGatewayName"      type="xsd:string" use="optional" />
        <xsd:attribute name="webGatewayAgentName" type="xsd:string" use="optional" />
        <xsd:attribute name="webGatewayAgentQMgr" type="xsd:string" use="optional" />
    </xsd:complexType>

    <xsd:complexType name="webUserType">
        <xsd:attribute name="webGatewayName"      type="xsd:string" use="required" />
        <xsd:attribute name="webGatewayAgentName" type="xsd:string" use="optional" />
        <xsd:attribute name="webGatewayAgentQMgr" type="xsd:string" use="optional" />
    </xsd:complexType>

</xsd:schema>
```

Understanding the transfer log message

`<transaction>`

   Group element that specifies a group of transfers you want to perform together.

Attribute	Description
version	Specifies the version of this element as detailed by WebSphere MQ Managed File Transfer.
ID	Specifies the unique transaction ID. The ID can be a maximum of 48 alphanumeric characters.
relatedID	Optional. If the transaction is the delete or download of a file from a file space, `relatedID` specifies the transaction ID of the transfer that uploaded the file to the file space.
agentRole	Optional. Specifies whether the agent concerned is on the source or destination system
xmlns:xsi	Namespace declaration. Indicates that the elements and data types used in this schema derive from the "http://www.w3.org/2001/XMLSchema-instance" namespace.
xsi:noNamespaceSchemaLocation	Specifies the name and location of the XML schema document to validate this message against if there is no namespace declaration. The value you specify for this attribute must refer to a WebSphere MQ Managed File Transfer `TransferLog.xsd` document.

`<action>`

Describes the status of the file transfer at the time logged by the time attribute. The status can be one of the following values:

- started
- progress
- completed
- cancelled
- malformed (indicates the file transfer request message content can not be interpreted.)
- notAuthorized
- deleted

Attribute	Description
time	The time that the transfer status was captured, expressed in UTC format.

`<sourceAgent>`

Specifies the name of the agent on the system where the source file is located. Only one of <sourceAgent>, <sourceWebUser>, and <sourceWebGateway> can be specified.

<startExits>

Group element that contains one or more user exit elements. This element can occur once only.

<endExits>

Group element that contains one or more user exit elements. This element can occur once only.

<systemInfo>

Describes the system architecture, name, and version. This element can occur once only.

Attribute	Description
agent	The name of the agent on the source system.

Attribute	Description
QMgr	The name of the queue manager on the source system.
agentType	The type of the agent. Valid values are: • STANDARD - a normal agent • BRIDGE - a protocol bridge agent • CD_BRIDGE - a Connect:Direct bridge agent • EMBEDDED - an embedded agent • WEB_GATEWAY - a web agent • SFG - a Sterling File Gateway embedded agent
bridgeURL	Optional. If the agent is a protocol bridge agent, the host name of the system hosting the protocol server.
pnode	Optional. If the agent is a Connect:Direct bridge agent, the name of the Connect:Direct primary node involved in the transfer.
snode	Optional. If the agent is a Connect:Direct bridge agent, the name of the Connect:Direct secondary node involved in the transfer.
bridgeNode	Optional. If the agent is a Connect:Direct bridge agent, the name of the Connect:Direct node that is part of the Connect:Direct bridge. This is the same node as either the primary node or the secondary node.

`<sourceWebUser>`

Specifies the name of the web user that uploads the source file to the Web Gateway. Only one of <sourceAgent>, <sourceWebUser>, and <sourceWebGateway> can be specified.

Attribute	Description
webGatewayName	The name of the Web Gateway that receives the file from the web user.
webGatewayAgentName	The name of the web agent that the Web Gateway uses to send the file to the destination.
webGatewayAgentQMgr	The name of the queue manager of the web agent.

`<sourceWebGateway>`

Specifies the name of the Web Gateway that the source file is downloaded from. Only one of <sourceAgent>, <sourceWebUser>, and <sourceWebGateway> can be specified.

Attribute	Description
webGatewayName	The name of the Web Gateway that receives the file from the web user.
webGatewayAgentName	The name of the web agent that the Web Gateway uses to send the file to the destination.
webGatewayAgentQMgr	The name of the queue manager of the web agent.

`<destinationAgent>`

Specifies the name of the agent on the system the file was transferred to. Only one of <destinationAgent>, <destinationWebGateway>, and <destinationWebUser> can be specified.

Attribute	Description
agent	The name of the agent on the destination system.

Attribute	Description
QMgr	The name of the queue manager on the destination system.
agentType	The type of the agent. Valid values are: • STANDARD - a normal agent • BRIDGE - a protocol bridge agent • CD_BRIDGE - a Connect:Direct bridge agent • EMBEDDED - an embedded agent • WEB_GATEWAY - a web agent • SFG - a Sterling File Gateway embedded agent
bridgeURL	Optional. If the agent is a protocol bridge agent, the host name of the system hosting the protocol server.
pnode	Optional. If the agent is a Connect:Direct bridge agent, the name of the Connect:Direct primary node involved in the transfer.
snode	Optional. If the agent is a Connect:Direct bridge agent, the name of the Connect:Direct secondary node involved in the transfer.
bridgeNode	Optional. If the agent is a Connect:Direct bridge agent, the name of the Connect:Direct node that is part of the Connect:Direct bridge. This is the same node as either the primary node or the secondary node.

<startExits>
> Group element that contains one or more user exit elements. This element can occur once only.

<endExits>
> Group element that contains one or more user exit elements. This element can occur once only.

<systemInfo>
> Describes the system architecture, name, and version. This element can occur once only.

<destinationWebUser>
> Specifies the name of the web user who downloads the file from the Web Gateway. Only one of <destinationAgent>, <destinationWebGateway>, and <destinationWebUser> can be specified.

Attribute	Description
webGatewayName	The name of the Web Gateway that receives the file from the web user.

<destinationWebGateway>
> Specifies the name of the web user who downloads the file from the Web Gateway. Only one of <destinationAgent>, <destinationWebGateway>, and <destinationWebUser> can be specified.

Attribute	Description
webGatewayName	The name of the Web Gateway that receives the file from the web user.
webGatewayAgentName	The name of the web agent that the Web Gateway uses.
webGatewayAgentQMgr	The name of the queue manager of the web agent.

\<originator\>
Group element that contains the elements specifying the originator of the request.

> **\<hostName\>**
> The host name of the system where the source file is located.

> **\<userID\>**
> The user ID that originated the file transfer.

> **\<mqmdUserID\>**
> The WebSphere MQ user ID that was supplied in the message descriptor (MQMD)

> **\<webUserID\>**
> Optional. The user ID that was supplied to the web browser submitting the transfer request.

> **\<webBrowser\>**
> Optional. The web browser that the transfer request was submitted from.

\<status\>
The result code and supplement messages.

\<trigger\>
Group element that contains the trigger elements defined in the original transfer request. These elements can be either or both of the following:

> **\<fileExist\>**
> Trigger condition based on whether a file exists

> **\<fileSize\>**
> Trigger condition based on whether a file meets or exceeds the specified size

\<transferSet\>
Specifies a group of file transfers you want to perform together. During transmission \<transferSet\> is a group element containing \<item\> elements.

Attribute	Description
startTime	Records the time that the set of transfers started, expressed in UTC format.
total	Specifies the total number of items in this set of transfers.
index	Optional attribute. Specifies the position of the first item in progress of the transfer set. The index attribute increments from zero. For example, if the index is set to 1, the progress message is the second of two items.
size	Optional attribute. Specifies the number of items in the progress report.
priority	Optional attribute. Priority level of the transfer. Priority is a value in the range 0-9, where 0 is the lowest priority. The default priority level is 0 and by default the transfer uses the source agent priority level.

> **\<metaDataSet\>**
> Group element containing one or more of the following attributes:

\<metaData\>

Attribute	Description
key	The key half of a metadata key-value pair. The \<metaData\> element content contains the value half of the pair. For example \<metaData key="testkey1"\>testvalue1\</metaData\>

<job>

Group element that contains an element specifying job details. <job> is a user-defined job name identifier that is added to the log message when the transfer has started. This <job> element is the same as the <job> element that is included in the transfer request message, which is described in the following topic: "File transfer request message format" on page 803.

> **<name>**
> The value of name can be any string.

<scheduleLog>

Group element that contains elements specifying the source and destination file names and locations.

Attribute	Description
ID	Matches the schedule ID if the transfer is a scheduled transfer.

<item>

Group element that contains elements specifying the source and destination file names and locations.

<source>

Group element that contains the <file> element or the <queue> element, and the <checksum> element for the file on the source system.

Attribute	Description
recursive	Specifies that files are transferred recursively in subdirectories when the <source> element is a directory or contains wildcard characters.
disposition	Specifies the action that is taken on the <source> element when <source> has successfully been transferred to its destination. The valid options are as follows: • leave - the source files are left unchanged. • delete - the source files are deleted from the source system after the source file is successfully transferred.
correlationBoolean	A boolean correlation value. If the source is a Connect:Direct bridge, this specifies whether the Connect:Direct process is user-defined.
correlationString1	A string correlation value. If the source is a Connect:Direct bridge, this specifies the name of the Connect:Direct process that occurs at the destination of the transfer.
correlationNum1	A numeric correlation value. If the source is a Connect:Direct bridge, this specifies the ID number of the Connect:Direct process that occurs at the destination of the transfer.

<queue>

When used with the <source> element, specifies the name of the queue that the transferred messages were read from, which is located on the source agent queue manager.

Attribute	Description
messageCount	The number of messages that were read from the queue.
groupId	The WebSphere MQ group ID of the messages read from the queue.

`<destination>`

Group element that contains the <file> element or the <queue> element, and <checksum> element for the destination.

Only one of <file> and <queue> is present as a child element of destination.

Attribute	Description
type	The type of destination. The valid options are as follows: • file - specifies a file as the destination • directory - specifies a directory as the destination • queue- specifies a WebSphere MQ queue as the destination The options file and directory can be present only when the **<destination>** element has a child element of **<file>**. The option queue can be present only when the **<destination>** element has a child element of **<queue>**.
exist	Specifies the action that is taken if a destination file exists on the destination system. The valid options are as follows: • error - reports an error and the file is not transferred. • overwrite - overwrites the existing destination file. This attribute cannot be present if the **<destination>** element has a child element of **<queue>**.
correlationBoolean	A boolean correlation value. If the destination is a Connect:Direct bridge, this specifies whether the Connect:Direct process is user-defined.
correlationString1	A string correlation value. If the destination is a Connect:Direct bridge, this specifies the name of the Connect:Direct process that occurs at the destination of the transfer.
correlationNum1	A numeric correlation value. If the destination is a Connect:Direct bridge, this specifies the ID number of the Connect:Direct process that occurs at the destination of the transfer.

`<file>`

Specifies the absolute path of the file that was transferred (both at the source and destination). The fully-qualified path is in the format consistent with your operating system, for example `C:/from/here.txt`. File URIs are not used.

`<queue>`

When used with the <destination> element, specifies the name of the queue that was transferred to, which is located on any queue manager that is connected to the destination agent queue manager.

Attribute	Description
messageCount	The number of messages that were written to the queue.
messageLength	The length of the messages written to the queue.
groupId	If the transfer request specified that the file is split into multiple messages, the value of this attribute is the WebSphere MQ group ID of the messages written to the queue.

Attribute	Description
messageId	If the transfer request did not specify that the file is split into multiple messages, the value of this attribute is the WebSphere MQ message ID of the message written to the queue.

`<checksum>`

Optional element.

Specifies the type of hash algorithm that generated the message digest to create the digital signature. Currently WebSphere MQ Managed File Transfer supports Message Digest algorithm 5 (MD5) only. The checksum provides a way for you to confirm the integrity of transferred files is intact.

`<malformed>`

Group element for malformed messages.

Attribute	Description
version	
ID	
agentRole	Either source agent or destination agent

`<statistics>`

Group element for statistical information for the transfer (when available).

`<actualStartTime>`

The actual time that the agent started running the transfer. Typically, the time is the same as (or very close to) the start time recorded for the transfer. However, when an agent is busy submitted transfers might be queued until the agent has capacity to run the transfers.

`<retryCount>`

The number of times that the transfer went into the recovery state and was retried by the agent. A transfer can go into a recovery state because the source and destination agents lose communication, either because of a WebSphere MQ network error or because they are not receiving data or acknowledgment messages for a period. This period is determined by the agent properties: transferAckTimeout and transferAckTimeoutRetries.

`<numFileFailures>`

The number of files in the transferSet that failed to transfer successfully.

`<numFileWarnings>`

The number of files in the transferSet that generated warnings while being transferred, but otherwise transferred successfully.

Examples

Examples of XML messages that conform to this schema are provided for each of the following types of transfer:

- A transfer of a single file
- A transfer that contains multiple files
- A failed file transfer
- A transfer defined with a trigger
- A transfer started by a schedule
- A transfer that calls user exits
- A transfer requested through the Web Gateway
- A transfer through a Connect:Direct bridge node

Single transfer log message examples:

When a transfer occurs, messages are published to the SYSTEM.FTE topic with a topic string of Log/*agent_name*/*transfer_ID*. The XML examples show the log messages for a single file transfer being started, in progress, and completed.

Single file transfer - started

```
<?xml version="1.0" encoding="UTF-8"?>
<transaction xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             version="4.00"
             ID="414d51205553322e42494e44494e47538b0f404d223d0020"
             agentRole="sourceAgent"
             xsi:noNamespaceSchemaLocation="TransferLog.xsd"
             xmlns="">
    <action time="2011-01-26T13:03:26.484Z">started</action>
    <sourceAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER">
        <systemInfo architecture="x86" name="Linux" version="2.6.31-21-generic"/>
    </sourceAgent>
    <destinationAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER"/>
    <originator>
        <hostName>dhcp-9-20-240-199.hursley.ibm.com.</hostName>
        <userID>mqm</userID>
        <mqmdUserID>mqm</mqmdUserID>
    </originator>
    <transferSet startTime="2011-01-26T13:03:26.484Z" total="1" bytesSent="0">
        <metaDataSet>
            <metaData key="com.ibm.wmqfte.SourceAgent">AGENT_JUPITER</metaData>
            <metaData key="com.ibm.wmqfte.DestinationAgent">AGENT_JUPITER</metaData>
            <metaData key="com.ibm.wmqfte.MqmdUser">mqm</metaData>
            <metaData key="com.ibm.wmqfte.OriginatingUser">mqm</metaData>
            <metaData key="com.ibm.wmqfte.OriginatingHost">dhcp-9-20-240-199.hursley.ibm.com.</metaData>
            <metaData key="com.ibm.wmqfte.TransferId">414d51205553322e42494e44494e47538b0f404d223d0020</metaData>
            <metaData key="com.ibm.wmqfte.ScheduleId">3</metaData>
            <metaData key="com.ibm.wmqfte.Priority">0</metaData>
        </metaDataSet>
    </transferSet>
    <scheduleLog ID="3"/>
</transaction>
```

Single file transfer success - progress

```
<?xml version="1.0" encoding="UTF-8"?>
<transaction xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             version="4.00"
             ID="414d51205553322e42494e44494e47538b0f404d223d0020"
             agentRole="sourceAgent"
             xsi:noNamespaceSchemaLocation="TransferLog.xsd"
             xmlns="">
        <action time="2011-01-26T13:03:26.615Z">progress</action>
        <sourceAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER">
            <systemInfo architecture="x86" name="Linux" version="2.6.31-21-generic"/>
        </sourceAgent>
        <destinationAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER">
            <systemInfo architecture="x86" name="Linux" version="2.6.31-21-generic"/>
        </destinationAgent>
        <originator>
            <hostName>example.com.</hostName>
            <userID>mqm</userID>
            <mqmdUserID>mqm</mqmdUserID>
        </originator>
        <transferSet index="0" size="1" startTime="2011-01-26T13:03:26.484Z" total="1" bytesSent="1198">
        <item mode="binary">
            <source disposition="leave" type="file">
                <file size="1151" last-modified="2009-11-02T10:37:01.000Z">/etc/passwd</file>
                <checksum method="MD5">2287181c07199f879de28296371cb24c</checksum>
            </source>
            <destination type="file">
                <file size="1151" last-modified="2011-01-26T13:03:26.000Z">/tmp/passwd</file>
                <checksum method="MD5">2287181c07199f879de28296371cb24c</checksum>
            </destination>
            <status resultCode="0"/>
        </item>
    </transferSet>
</transaction>
```

Single file transfer success - completed

```xml
<?xml version="1.0" encoding="UTF-8"?>
<transaction xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             version="4.00"
             ID="414d51205553322e42494e44494e47538b0f404d223d0020"
             agentRole="sourceAgent"
             xsi:noNamespaceSchemaLocation="TransferLog.xsd"
             xmlns="">
    <action time="2011-01-26T13:03:26.622Z">completed</action>
    <sourceAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER">
        <systemInfo architecture="x86" name="Linux" version="2.6.31-21-generic"/>
    </sourceAgent>
    <destinationAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER">
        <systemInfo architecture="x86" name="Linux" version="2.6.31-21-generic"/>
    </destinationAgent>
    <originator>
        <hostName>example.com.</hostName>
        <userID>mqm</userID>
        <mqmdUserID>mqm</mqmdUserID>
    </originator>
    <status resultCode="0">
        <supplement>BFGRP0032I: The file transfer request has successfully completed.</supplement>
    </status>
    <transferSet startTime="2011-01-26T13:03:26.484Z" total="1" bytesSent="1198">
        <metaDataSet>
            <metaData key="com.ibm.wmqfte.SourceAgent">AGENT_JUPITER</metaData>
            <metaData key="com.ibm.wmqfte.DestinationAgent">AGENT_JUPITER</metaData>
            <metaData key="com.ibm.wmqfte.MqmdUser">mqm</metaData>
            <metaData key="com.ibm.wmqfte.OriginatingUser">mqm</metaData>
            <metaData key="com.ibm.wmqfte.OriginatingHost">example.com.</metaData>
            <metaData key="com.ibm.wmqfte.TransferId">414d51205553322e42494e44494e47538b0f404d223d0020</metaData>
            <metaData key="com.ibm.wmqfte.ScheduleId">3</metaData>
            <metaData key="com.ibm.wmqfte.Priority">0</metaData>
        </metaDataSet>
    </transferSet>
    <statistics>
        <actualStartTime>2011-01-26T13:03:26.541Z</actualStartTime>
        <retryCount>0</retryCount>
        <numFileFailures>0</numFileFailures>
        <numFileWarnings>0</numFileWarnings>
    </statistics>
</transaction>
```

Multiple file transfer log message examples:

Examples of the messages that are published to the SYSTEM.FTE topic with a topic string of Log/*agent_name*/*transfer_ID* when a transfer that contains multiple files occurs.

Multiple file transfer - started

```xml
<?xml version="1.0" encoding="UTF-8"?>
<transaction xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             version="4.00"
             ID="414d51205553322e42494e44494e47538b0f404d035c0020"
             agentRole="sourceAgent"
             xsi:noNamespaceSchemaLocation="TransferLog.xsd"
             xmlns="">
    <action time="2011-01-26T13:12:58.534Z">started</action>
    <sourceAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER">
        <systemInfo architecture="x86" name="Linux" version="2.6.31-21-generic"/>
    </sourceAgent>
    <destinationAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER"/>
    <originator>
        <hostName>example.com</hostName>
        <userID>mqm</userID>
        <mqmdUserID>mqm</mqmdUserID>
    </originator>
    <transferSet startTime="2011-01-26T13:12:58.534Z" total="6" bytesSent="0">
        <metaDataSet>
            <metaData key="com.ibm.wmqfte.SourceAgent">AGENT_JUPITER</metaData>
            <metaData key="com.ibm.wmqfte.DestinationAgent">AGENT_JUPITER</metaData>
            <metaData key="com.ibm.wmqfte.MqmdUser">mqm</metaData>
            <metaData key="com.ibm.wmqfte.OriginatingUser">mqm</metaData>
            <metaData key="com.ibm.wmqfte.OriginatingHost">example.com</metaData>
            <metaData key="com.ibm.wmqfte.TransferId">414d51205553322e42494e44494e47538b0f404d035c0020</metaData>
```

```
            <metaData key="com.ibm.wmqfte.Priority">0</metaData>
        </metaDataSet>
    </transferSet>
</transaction>
```

Multiple file transfer - progress

```
<?xml version="1.0" encoding="UTF-8"?>
<transaction xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             version="4.00"
             ID="414d51205553322e42494e44494e47538b0f404d035c0020"
             agentRole="sourceAgent"
             xsi:noNamespaceSchemaLocation="TransferLog.xsd"
             xmlns="">
    <action time="2011-01-26T13:12:58.753Z">progress</action>
    <sourceAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER">
        <systemInfo architecture="x86" name="Linux" version="2.6.31-21-generic"/>
    </sourceAgent>
    <destinationAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER">
        <systemInfo architecture="x86" name="Linux" version="2.6.31-21-generic"/>
    </destinationAgent>
    <originator>
        <hostName>example.com.</hostName>
        <userID>mqm</userID>
        <mqmdUserID>mqm</mqmdUserID>
    </originator>
    <transferSet index="0" size="6" startTime="2011-01-26T13:12:58.534Z" total="6" bytesSent="440">
        <item mode="binary">
            <source disposition="leave" type="file">
                <file size="0" last-modified="2011-01-26T13:10:19.000Z">/srv/nfs/incoming/file01.txt</file>
                <checksum method="MD5">d41d8cd98f00b204e9800998ecf8427e</checksum>
            </source>
            <destination type="file">
                <file size="0" last-modified="2011-01-26T13:12:58.000Z">/srv/nfs/outgoing/file01.txt</file>
                <checksum method="MD5">d41d8cd98f00b204e9800998ecf8427e</checksum>
            </destination>
            <status resultCode="0"/>
        </item>
        <item mode="binary">
            <source disposition="leave" type="file">
                <file size="0" last-modified="2011-01-26T13:10:19.000Z">/srv/nfs/incoming/file02.txt</file>
                <checksum method="MD5">d41d8cd98f00b204e9800998ecf8427e</checksum>
            </source>
            <destination type="file">
                <file size="0" last-modified="2011-01-26T13:12:58.000Z">/srv/nfs/outgoing/file02.txt</file>
                <checksum method="MD5">d41d8cd98f00b204e9800998ecf8427e</checksum>
            </destination>
            <status resultCode="0"/>
        </item>
        <item mode="binary">
            <source disposition="leave" type="file">
                <file size="0" last-modified="2011-01-26T13:10:19.000Z">/srv/nfs/incoming/file03.txt</file>
                <checksum method="MD5">d41d8cd98f00b204e9800998ecf8427e</checksum>
            </source>
            <destination type="file">
                <file size="0" last-modified="2011-01-26T13:12:58.000Z">/srv/nfs/outgoing/file03.txt</file>
                <checksum method="MD5">d41d8cd98f00b204e9800998ecf8427e</checksum>
            </destination>
            <status resultCode="0"/>
        </item>
        <item mode="binary">
            <source disposition="leave" type="file">
                <file size="0" last-modified="2011-01-26T13:10:19.000Z">/srv/nfs/incoming/file04.txt</file>
                <checksum method="MD5">d41d8cd98f00b204e9800998ecf8427e</checksum>
            </source>
            <destination type="file">
                <file size="0" last-modified="2011-01-26T13:12:58.000Z">/srv/nfs/outgoing/file04.txt</file>
                <checksum method="MD5">d41d8cd98f00b204e9800998ecf8427e</checksum>
            </destination>
            <status resultCode="0"/>
        </item>
        <item mode="binary">
            <source disposition="leave" type="file">
                <file size="0" last-modified="2011-01-26T13:10:19.000Z">/srv/nfs/incoming/file05.txt</file>
                <checksum method="MD5">d41d8cd98f00b204e9800998ecf8427e</checksum>
            </source>
            <destination type="file">
                <file size="0" last-modified="2011-01-26T13:12:58.000Z">/srv/nfs/outgoing/file05.txt</file>
```

```
            <checksum method="MD5">d41d8cd98f00b204e9800998ecf8427e</checksum>
        </destination>
        <status resultCode="0"/>
    </item>
    <item mode="binary">
        <source disposition="leave" type="file">
            <file size="0" last-modified="2011-01-26T13:10:19.000Z">/srv/nfs/incoming/file06.txt</file>
            <checksum method="MD5">d41d8cd98f00b204e9800998ecf8427e</checksum>
        </source>
        <destination type="file">
            <file size="0" last-modified="2011-01-26T13:12:58.000Z">/srv/nfs/outgoing/file06.txt</file>
            <checksum method="MD5">d41d8cd98f00b204e9800998ecf8427e</checksum>
        </destination>
        <status resultCode="0"/>
    </item>
    </transferSet>
</transaction>
```

Multiple file transfer - completed

```
<?xml version="1.0" encoding="UTF-8"?>
<transaction xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            version="4.00"
            ID="414d51205553322e42494e44494e47538b0f404d035c0020"
            agentRole="sourceAgent"
            xsi:noNamespaceSchemaLocation="TransferLog.xsd"
            xmlns="">
    <action time="2011-01-26T13:12:58.766Z">completed</action>
    <sourceAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER">
        <systemInfo architecture="x86" name="Linux" version="2.6.31-21-generic"/>
    </sourceAgent>
    <destinationAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER">
        <systemInfo architecture="x86" name="Linux" version="2.6.31-21-generic"/>
    </destinationAgent>
    <originator>
        <hostName>example.com.</hostName>
        <userID>mqm</userID>
        <mqmdUserID>mqm</mqmdUserID>
    </originator>
    <status resultCode="0">
        <supplement>BFGRP0032I: The file transfer request has successfully completed.</supplement>
    </status>
    <transferSet startTime="2011-01-26T13:12:58.534Z" total="6" bytesSent="440">
        <metaDataSet>
            <metaData key="com.ibm.wmqfte.SourceAgent">AGENT_JUPITER</metaData>
            <metaData key="com.ibm.wmqfte.DestinationAgent">AGENT_JUPITER</metaData>
            <metaData key="com.ibm.wmqfte.MqmdUser">mqm</metaData>
            <metaData key="com.ibm.wmqfte.OriginatingUser">mqm</metaData>
            <metaData key="com.ibm.wmqfte.OriginatingHost">example.com.</metaData>
            <metaData key="com.ibm.wmqfte.TransferId">414d51205553322e42494e44494e47538b0f404d035c0020</metaData>
            <metaData key="com.ibm.wmqfte.Priority">0</metaData>
        </metaDataSet>
    </transferSet>
    <statistics>
        <actualStartTime>2011-01-26T13:12:58.634Z</actualStartTime>
        <retryCount>0</retryCount>
        <numFileFailures>0</numFileFailures>
        <numFileWarnings>0</numFileWarnings>
    </statistics>
</transaction>
```

Failed transfer log message examples:

When a transfer occurs, messages are published to the SYSTEM.FTE topic with a topic string of
Log/*agent_name*/*transfer_ID*. The XML examples show the log messages for a file transfer that fails being
started, in progress, and completed.

File transfer failure - started

```
<?xml version="1.0" encoding="UTF-8"?>
<transaction xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      version="4.00"
      ID="414d51205553322e42494e44494e47538b0f404d03620020"
      agentRole="sourceAgent"
      xsi:noNamespaceSchemaLocation="TransferLog.xsd"
      xmlns="">
```

```
            <action time="2011-01-26T13:19:15.767Z">started</action>
            <sourceAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER">
                <systemInfo architecture="x86" name="Linux" version="2.6.31-21-generic"/>
            </sourceAgent>
            <destinationAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER"/>
            <originator>
                <hostName>example.com.</hostName>
                <userID>mqm</userID>
                <mqmdUserID>mqm</mqmdUserID>
            </originator>
            <transferSet startTime="2011-01-26T13:19:15.767Z" total="1" bytesSent="0">
                <metaDataSet>
                    <metaData key="com.ibm.wmqfte.SourceAgent">AGENT_JUPITER</metaData>
                    <metaData key="com.ibm.wmqfte.DestinationAgent">AGENT_JUPITER</metaData>
                    <metaData key="com.ibm.wmqfte.MqmdUser">mqm</metaData>
                    <metaData key="com.ibm.wmqfte.OriginatingUser">mqm</metaData>
                    <metaData key="com.ibm.wmqfte.OriginatingHost">example.com.</metaData>
                    <metaData key="com.ibm.wmqfte.TransferId">414d51205553322e42494e44494e47538b0f404d03620020</metaData>
                    <metaData key="com.ibm.wmqfte.Priority">0</metaData>
                </metaDataSet>
            </transferSet>
</transaction>
```

File transfer failure - progress

```
<?xml version="1.0" encoding="UTF-8"?>
<transaction xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             version="4.00"
             ID="414d51205553322e42494e44494e47538b0f404d03620020"
             agentRole="sourceAgent"
             xsi:noNamespaceSchemaLocation="TransferLog.xsd"
             xmlns="">
    <action time="2011-01-26T13:19:15.944Z">progress</action>
    <sourceAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER">
        <systemInfo architecture="x86" name="Linux" version="2.6.31-21-generic"/>
    </sourceAgent>
    <destinationAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER">
        <systemInfo architecture="x86" name="Linux" version="2.6.31-21-generic"/>
    </destinationAgent>
    <originator>
        <hostName>example.com.</hostName>
        <userID>mqm</userID>
        <mqmdUserID>mqm</mqmdUserID>
    </originator>
    <transferSet index="0" size="1" startTime="2011-01-26T13:19:15.767Z" total="1" bytesSent="0">
        <item mode="binary">
            <source disposition="leave" type="file">
                <file size="0" last-modified="2011-01-26T13:10:19.000Z">/srv/nfs/incoming/file01.txt</file>
                <checksum method="MD5">d41d8cd98f00b204e9800998ecf8427e</checksum>
            </source>
            <destination type="file">
                <file>/srv/nfs/outgoing/file01.txt</file>
            </destination>
            <status resultCode="1">
                <supplement>BFGIO0006E: File "/srv/nfs/outgoing/file01.txt" already exists.</supplement>
            </status>
        </item>
    </transferSet>
</transaction>
```

File transfer failure - completed

```
<?xml version="1.0" encoding="UTF-8"?>
<transaction xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             version="4.00"
             ID="414d51205553322e42494e44494e47538b0f404d03620020"
             agentRole="sourceAgent"
             xsi:noNamespaceSchemaLocation="TransferLog.xsd"
             xmlns="">
    <action time="2011-01-26T13:19:15.948Z">completed</action>
    <sourceAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER">
        <systemInfo architecture="x86" name="Linux" version="2.6.31-21-generic"/>
    </sourceAgent>
    <destinationAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER">
        <systemInfo architecture="x86" name="Linux" version="2.6.31-21-generic"/>
    </destinationAgent>
    <originator>
        <hostName>example.com.</hostName>
```

```
        <userID>mqm</userID>
        <mqmdUserID>mqm</mqmdUserID>
    </originator>
    <status resultCode="40">
        <supplement>BFGRP0034I: The file transfer request has
                     completed with no files being transferred.
        </supplement>
    </status>
    <transferSet startTime="2011-01-26T13:19:15.767Z" total="1" bytesSent="0">
        <metaDataSet>
            <metaData key="com.ibm.wmqfte.SourceAgent">AGENT_JUPITER</metaData>
            <metaData key="com.ibm.wmqfte.DestinationAgent">AGENT_JUPITER</metaData>
            <metaData key="com.ibm.wmqfte.MqmdUser">mqm</metaData>
            <metaData key="com.ibm.wmqfte.OriginatingUser">mqm</metaData>
            <metaData key="com.ibm.wmqfte.OriginatingHost">example.com.</metaData>
            <metaData key="com.ibm.wmqfte.TransferId">414d51205553322e42494e44494e47538b0f404d03620020</metaData>
            <metaData key="com.ibm.wmqfte.Priority">0</metaData>
        </metaDataSet>
    </transferSet>
    <statistics>
        <actualStartTime>2011-01-26T13:19:15.878Z</actualStartTime>
        <retryCount>0</retryCount>
        <numFileFailures>1</numFileFailures>
        <numFileWarnings>0</numFileWarnings>
    </statistics>
</transaction>
```

Triggered transfer message format:

When a transfer is in progress, messages are published to the SYSTEM.FTE topic with a topic string of
Log/*agent_name*/*transfer_ID*. The XML example shows the log message that is created when a file transfer
containing a trigger condition is started.

Trigger single file transfer success - started

```
<?xml version="1.0" encoding="UTF-8"?>
<transaction version="1.00"
 ID="414d5120514d31202020202020202020207e970d492000a102" agentRole="sourceAgent"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="TransferLog.xsd"
 xmlns="">
 <action time="2008-11-02T22:05:18.703Z">started</action>
 <sourceAgent agent="FTEAGENT" QMgr="QM1">
  <systemInfo architecture="x86" name="Windows 7"
   version="6.1 build 7601 Service Pack 1" />
 </sourceAgent>
 <destinationAgent agent="FTEAGENT" QMgr="QM1" />
 <originator>
  <hostName>reportserver.com</hostName>
  <userID>USER1</userID>
  <mqmdUserID>USER1    </mqmdUserID>
 </originator>
 <trigger log="yes">
  <fileExist comparison="=" value="exist">c:\trigger.txt</fileExist>
 </trigger>
 <transferSet startTime="2008-11-02T22:05:18.703Z" total="1"></transferSet>
</transaction>
```

Scheduled transfer log message examples:

When a transfer is in progress, messages are published to the SYSTEM.FTE topic with a topic string of
Log/*agent_name*/*transfer_ID*. The XML examples show the log messages that are created when a file
transfer occurs as a result of a schedule.

Schedule transfer transaction messages

When the file transfer is started as a result of the schedule entry expiring, the file transfer follows the
usual sequence of publishing transaction messages on the SYSTEM.FTE/Log/*agent_ name* topic for:

- Action started (`TransferLog.xsd`)
- Action progress (`TransferLog.xsd`)

- Action completed (`TransferLog.xsd`)

Only the log transaction message with the action of started contains the ID of the scheduled transfer, in the ID attribute of the <scheduleLog> element. This allows the schedule ID to be tied to the transfer ID throughout the lifecycle of the entire transfer.

Started:

```
<?xml version="1.0" encoding="UTF-8"?>
<transaction version="1.00"
 ID="414d5120514d3120202020202020202020248e294920004016" agentRole="sourceAgent"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="TransferLog.xsd"
 xmlns="">
 <action time="2008-11-23T21:55:03.111Z">started</action>
 .
 .
 .
 <scheduleLog ID="6" />
</transaction>
```

Progress:

```
<?xml version="1.0" encoding="UTF-8"?>
<transaction version="1.00"
 ID="414d5120514d3120202020202020202020248e294920004016" agentRole="sourceAgent"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="TransferLog.xsd"
 xmlns="">
 <action time="2008-11-23T21:55:03.377Z">progress</action>
 .
 .
 .
</transaction>
```

Completed:

```
<?xml version="1.0" encoding="UTF-8"?>
<transaction version="1.00"
 ID="414d5120514d3120202020202020202020248e294920004016" agentRole="sourceAgent"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="TransferLog.xsd"
 xmlns="">
 <action time="2008-11-23T21:55:03.424Z">completed</action>
 .
 .
 .

</transaction>
```

User exit message formats:

When a transfer is in progress, messages are published to the SYSTEM.FTE topic with a topic string of Log/*agent_name*/*transfer_ID*. The XML examples show the log messages that are created when a file transfer occurs that contains calls to user exits.

Exit single file transfer proceed - started

```
<?xml version="1.0" encoding="UTF-8"?>
<transaction version="1.00"
 ID="414d5120514d31202020202020202020207e970d492000d502" agentRole="sourceAgent"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="TransferLog.xsd"
 xmlns="">
 <action time="2008-11-02T22:36:13.046Z">started</action>
 <sourceAgent agent="FTEAGENT" QMgr="QM1">
  <systemInfo architecture="x86" name="Windows 7"
   version="6.1 build 7601 Service Pack 1" />
 </sourceAgent>
 <destinationAgent agent="FTEAGENT" QMgr="QM1" />
 <originator>
```

```
  <hostName>reportserver.com</hostName>
  <userID>USER1</userID>
  <mqmdUserID>USER1        </mqmdUserID>
 </originator>
 <transferSet startTime="2008-11-02T22:36:13.046Z" total="1">
  <metaDataSet>
   <metaData key="testkey1">testvalue1</metaData>
   <metaData key="testkey2">testvalue2</metaData>
  </metaDataSet>
 </transferSet>
</transaction>
```

Exit single file transfer proceed - completed

```
<?xml version="1.0" encoding="UTF-8"?>
<transaction version="1.00"
 ID="414d5120514d31202020202020202020207e970d492000d502"
  agentRole="sourceAgent"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="TransferLog.xsd"
xmlns="">
 <action time="2008-11-02T22:36:13.546Z">completed</action>
 <sourceAgent agent="FTEAGENT" QMgr="QM1">
  <startExits>
   <exit name="class testExits.SourceExit1">
    <status resultCode="proceed">
     <supplement>Source Start, modified metadata</supplement>
    </status>
   </exit>
  </startExits>
  <endExits>
   <exit name="class testExits.SourceExit1">
    <status>
     <supplement>Source End</supplement>
    </status>
   </exit>
  </endExits>
  <systemInfo architecture="x86" name="Windows 7"
   version="6.1 build 7601 Service Pack 1" />
 </sourceAgent>
 <destinationAgent agent="FTEAGENT" QMgr="QM1">
  <startExits>
   <exit name="class testExits.DestinationExitProceed">
    <status resultCode="proceed">
     <supplement>Destination start, with proceed</supplement>
    </status>
   </exit>
  </startExits>
  <endExits>
   <exit name="class testExits.DestinationExitProceed">
    <status>
     <supplement>destination end</supplement>
    </status>
   </exit>
  </endExits>
  <systemInfo architecture="x86" name="Windows 7"
   version="6.1 build 7601 Service Pack 1" />
 </destinationAgent>
 <originator>
  <hostName>reportserver.com</hostName>
  <userID>USER1</userID>
  <mqmdUserID>USER1        </mqmdUserID>
 </originator>
```

```
  <transferSet startTime="2008-11-02T22:36:13.046Z" total="1">
   <metaDataSet>
    <metaData key="newkey2">newvalue2</metaData>
    <metaData key="newkey1">newvalue1</metaData>
    <metaData key="newkey4">newvalue4</metaData>
    <metaData key="newkey3">newvalue3</metaData>
    <metaData key="newkey5">newvalue5</metaData>
    <metaData key="testkey1">testvalue1</metaData>
    <metaData key="testkey2">testvalue2</metaData>
   </metaDataSet>
  </transferSet>
</transaction>

<!--
 In this example the source transfer start exit has modified the
 metadata as follows:

 Added keys and values for:
 newkey1, newvalue1
 newkey2, newvalue2
 newkey3, newvalue3
 newkey4, newvalue4
 newkey5, newvalue5

 Replaced values for:
 key1 to modifiedValue1

 Deleted keys and values for:
 key2
-->
```

Exit single file transfer cancel - canceled

```
<?xml version="1.0" encoding="UTF-8"?>
<transaction version="1.00"
 ID="414d5120514d31202020202020202020207e970d492000c702" agentRole="sourceAgent"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="TransferLog.xsd"
 xmlns="">
 <action time="2008-11-02T22:25:59.328Z">cancelled</action>
 <sourceAgent agent="FTEAGENT" QMgr="QM1">
  <startExits>
   <exit name="class testExits.SourceExit1">
    <status resultCode="proceed">
     <supplement>Source Start, modified metadata</supplement>
    </status>
   </exit>
  </startExits>
  <endExits>
   <exit name="class testExits.SourceExit1">
    <status>
     <supplement>Source End</supplement>
    </status>
   </exit>
  </endExits>
  <systemInfo architecture="x86" name="Windows 7"
   version="6.1 build 7601 Service Pack 1" />
 </sourceAgent>
 <destinationAgent agent="FTEAGENT" QMgr="QM1">
  <startExits>
   <exit name="class testExits.DestinationExit1">
    <status resultCode="cancelTransfer">
     <supplement>Destination start, with cancel</supplement>
```

```
    </status>
   </exit>
  </startExits>
  <endExits>
   <exit name="class testExits.DestinationExit1">
    <status>
     <supplement>destination end</supplement>
    </status>
   </exit>
  </endExits>
  <systemInfo architecture="x86" name="Windows 7"
   version="6.1 build 7601 Service Pack 1" />
 </destinationAgent>
 <originator>
  <hostName>reportserver.com</hostName>
  <userID>USER1</userID>
  <mqmdUserID>USER1        </mqmdUserID>
 </originator>
 <transferSet startTime="2008-11-02T22:25:59.078Z" total="1" />
</transaction>
```

Additions to message formats for web-based transfers:

The Started and Completed log messages from a transfer that was requested through the WebSphere MQ
Managed File Transfer Web Gateway SupportPac include extra metadata. This metadata contains
information about the HTTP request and about the application server hosting the Web Gateway.

Definitions of web metadata

com.ibm.wmqfte.web.request.authtype
> The method of authorization used by the user who submits the request to the Web Gateway.

com.ibm.wmqfte.web.request.locale
> The locale of the user who submits the request to the Web Gateway.

com.ibm.wmqfte.web.appsrv.type
> The type of application server that hosts the Web Gateway.

com.ibm.wmqfte.web.appsrv.host
> The host name or IP address of the system where the application server that hosts the Web
> Gateway is running.

com.ibm.wmqfte.web.appsrv.port
> The port number that the application server that hosts the Web Gateway is listening on.

The metadata that is included in the log messages for a transfer that was requested through the Web
Gateway is highlighted in the following examples.

Single file transfer - success
```
<?xml version="1.0" encoding="UTF-8"?>
<transaction version="1.00"
 ID="414d5120514d31202020202020202020207e970d4920008202" agentRole="sourceAgent"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="TransferLog.xsd"
 xmlns="">
 <action time="2008-11-02T21:20:37.578Z">started</action>
 <sourceAgent agent="FTEAGENT" QMgr="QM1">
  <systemInfo architecture="x86" name="Windows 7"
   version="6.1 build 7601 Service Pack 1" />
 </sourceAgent>
 <destinationAgent agent="FTEAGENT" QMgr="QM1" />
 <originator>
```

```
  <hostName>requestor.example.com</hostName>
  <userID>USER1    </userID>
  <mqmdUserID>USER1</mqmdUserID>
 </originator>
 <transferSet startTime="2008-11-02T21:20:37.593Z" total="1">
  <metaDataSet>
   <metaData key="com.ibm.wmqfte.web.request.authtype">BASIC</metaData>
   <metaData key="com.ibm.wmqfte.web.request.locale">en_GB</metaData>
   <metaData key="com.ibm.wmqfte.web.appsrv.type">
    Apache Geronimo (Embedded Tomcat/6.0.20-20090724)
   </metaData>
   <metaData key="com.ibm.wmqfte.web.appsrv.port">8080</metaData>
   <metaData key="com.ibm.wmqfte.web.appsrv.host">gateway.example.com</metaData>
  </metaDataSet>
 </transferSet>
</transaction>
```

Single file transfer success - completed

```
<?xml version="1.0" encoding="UTF-8"?>
<transaction version="1.00"
 ID="414d5120514d3120202020202020202020207e970d4920008202" agentRole="sourceAgent"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="TransferLog.xsd"
 xmlns="">
 <action time="2008-11-02T21:20:38.234Z">completed</action>
 <sourceAgent agent="FTEAGENT" QMgr="QM1">
  <systemInfo architecture="x86" name="Windows 7"
   version="6.1 build 7601 Service Pack 1" />
 </sourceAgent>
 <destinationAgent agent="FTEAGENT" QMgr="QM1">
  <systemInfo architecture="x86" name="Windows 7"
   version="6.1 build 7601 Service Pack 1" />
 </destinationAgent>
 <originator>
  <hostName>requestor.example.com</hostName>
  <userID>USER1</userID>
  <mqmdUserID>USER1    </mqmdUserID>
 </originator>
 <transferSet startTime="2008-11-02T21:20:37.593Z" total="1">
  <metaDataSet>
   <metaData key="com.ibm.wmqfte.web.request.authtype">BASIC</metaData>
   <metaData key="com.ibm.wmqfte.web.request.locale">en_GB</metaData>
   <metaData key="com.ibm.wmqfte.web.appsrv.type">
    Apache Geronimo (Embedded Tomcat/6.0.20-20090724)
   </metaData>
   <metaData key="com.ibm.wmqfte.web.appsrv.port">8080</metaData>
   <metaData key="com.ibm.wmqfte.web.appsrv.host">gateway.example.com</metaData>
  </metaDataSet>
 </transferSet>
</transaction>
```

Note: Note: The XML message formats described here are not the same as the message formats that are returned as responses from the Web Gateway. The XML formats that are returned by the Web Gateway are documented in the following topic: "Response formats: XML and JSON" on page 876.

Related reference:

"Single transfer log message examples" on page 636
When a transfer occurs, messages are published to the SYSTEM.FTE topic with a topic string of
Log/*agent_name/transfer_ID*. The XML examples show the log messages for a single file transfer being
started, in progress, and completed.

"Triggered transfer message format" on page 641
When a transfer is in progress, messages are published to the SYSTEM.FTE topic with a topic string of
Log/*agent_name/transfer_ID*. The XML example shows the log message that is created when a file transfer
containing a trigger condition is started.

"User exit message formats" on page 643
When a transfer is in progress, messages are published to the SYSTEM.FTE topic with a topic string of
Log/*agent_name/transfer_ID*. The XML examples show the log messages that are created when a file
transfer occurs that contains calls to user exits.

"File transfer log message formats" on page 626
File transfer log messages are published to the SYSTEM.FTE topic with a topic string of
Log/*agent_name/transfer_ID*. These messages conform to the schema TransferLog.xsd, which is located in
the *MQ_INSTALLATION_PATH*/mqft/samples/schema directory of your WebSphere MQ Managed File Transfer
installation.

Connect:Direct bridge transfer message examples:

The destinationAgent or sourceAgent element contains additional attributes when the destination agent
or source agent is a Connect:Direct bridge agent. The Started log message contains only a subset of the
information about the Connect:Direct transfer. The Progress and Completed log messages contain full
information about the Connect:Direct transfer.

Source agent is Connect:Direct bridge agent

Started:

```
<?xml version="1.0" encoding="UTF-8"?>
<transaction xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             ID="414d5120514d5f696b6b796f20202020a704654d20092507"
             agentRole="sourceAgent"
             version="4.00"
             xsi:noNamespaceSchemaLocation="TransferLog.xsd"
             xmlns="">
   <action time="2011-03-07T13:05:01.838Z">started</action>
   <sourceAgent QMgr="QM_KUIPER" agent="VARUNA" agentType="CD_BRIDGE" bridgeNode="CDNODE_VARUNA">
       <systemInfo architecture="x86" name="Windows 7" version="6.1 build 7601 Service Pack 1"/>
   </sourceAgent>
   <destinationAgent QMgr="QM_KUIPER" agent="IXION"/>
   <originator>
       <hostName>kuiper.example.com.</hostName>
       <userID>sol</userID>
       <mqmdUserID>sol</mqmdUserID>
   </originator>
   <transferSet bytesSent="0" startTime="2011-03-07T13:05:01.838Z" total="1">
       <metaDataSet>
           <metaData key="com.ibm.wmqfte.SourceAgent">VARUNA</metaData>
           <metaData key="com.ibm.wmqfte.DestinationAgent">IXION</metaData>
           <metaData key="com.ibm.wmqfte.MqmdUser">sol</metaData>
           <metaData key="com.ibm.wmqfte.OriginatingUser">sol</metaData>
           <metaData key="com.ibm.wmqfte.OriginatingHost">kuiper.example.com.</metaData>
           <metaData key="com.ibm.wmqfte.TransferId">414d5120514d5f696b6b796f20202020a704654d20092507</metaData>
           <metaData key="com.ibm.wmqfte.Priority">0</metaData>
       </metaDataSet>
   </transferSet>
</transaction>
```

Progress:

```
<?xml version="1.0" encoding="UTF-8"?>
<transaction xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             ID="414d5120514d5f696b6b796f20202020a704654d20092507"
             agentRole="sourceAgent"
```

```
            version="4.00"
            xsi:noNamespaceSchemaLocation="TransferLog.xsd"
            xmlns="">
    <action time="2011-03-07T13:05:03.448Z">progress</action>
    <sourceAgent QMgr="QM_KUIPER" agent="VARUNA" agentType="CD_BRIDGE"
            bridgeNode="CDNODE_VARUNA" pnode="CDNODE_VARUNA" snode="CDNODE_ERIS">
        <systemInfo architecture="x86" name="Windows 7" version="6.1 build 7601 Service Pack 1"/>
    </sourceAgent>
    <destinationAgent QMgr="QM_KUIPER" agent="IXION" agentType="STANDARD">
        <systemInfo architecture="x86" name="Windows 7" version="6.1 build 7601 Service Pack 1"/>
    </destinationAgent>
    <originator>
        <hostName>kuiper.example.com.</hostName>
        <userID>sol</userID>
        <mqmdUserID>sol</mqmdUserID>
    </originator>
    <transferSet bytesSent="48" index="0" size="1" startTime="2011-03-07T13:05:01.838Z" total="1">
        <item mode="binary">
            <source disposition="leave" processName="f2007567" processNumber="68" type="file">
                <file last-modified="2011-03-07T13:05:02.573Z" size="4">CDNODE_ERIS:D:/AGENTS/CDNODE_ERIS/test.txt</file>
                <checksum method="MD5">098f6bcd4621d373cade4e832627b4f6</checksum>
            </source>
            <destination type="file">
                <file last-modified="2011-03-07T13:05:03.338Z" size="4">D:\AGENTS\IXION\test.txt</file>
                <checksum method="MD5">098f6bcd4621d373cade4e832627b4f6</checksum>
            </destination>
            <status resultCode="0"/>
        </item>
    </transferSet>
</transaction>
```

Completed:

```
<?xml version="1.0" encoding="UTF-8"?>
<transaction xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            ID="414d5120514d5f696b6b796f20202020a704654d20092507"
            agentRole="sourceAgent"
            version="4.00" xsi:noNamespaceSchemaLocation="TransferLog.xsd"
            xmlns="">
    <action time="2011-03-07T13:05:03.495Z">completed</action>
    <sourceAgent QMgr="QM_KUIPER" agent="VARUNA" agentType="CD_BRIDGE"
            bridgeNode="CDNODE_VARUNA" pnode="CDNODE_VARUNA" snode="CDNODE_ERIS">
        <systemInfo architecture="x86" name="Windows 7" version="6.1 build 7601 Service Pack 1"/>
    </sourceAgent>
    <destinationAgent QMgr="QM_KUIPER" agent="IXION" agentType="STANDARD">
        <systemInfo architecture="x86" name="Windows 7" version="6.1 build 7601 Service Pack 1"/>
    </destinationAgent>
    <originator>
        <hostName>kuiper.example.com.</hostName>
        <userID>sol</userID>
        <mqmdUserID>sol</mqmdUserID>
    </originator>
    <status resultCode="0">
        <supplement>BFGRP0032I: The file transfer request has successfully completed.</supplement>
    </status>
    <transferSet bytesSent="48" startTime="2011-03-07T13:05:01.838Z" total="1">
        <metaDataSet>
            <metaData key="com.ibm.wmqfte.SourceAgent">VARUNA</metaData>
            <metaData key="com.ibm.wmqfte.DestinationAgent">IXION</metaData>
            <metaData key="com.ibm.wmqfte.MqmdUser">sol</metaData>
            <metaData key="com.ibm.wmqfte.OriginatingUser">sol</metaData>
            <metaData key="com.ibm.wmqfte.OriginatingHost">kuiper.example.com.</metaData>
            <metaData key="com.ibm.wmqfte.TransferId">414d5120514d5f696b6b796f20202020a704654d20092507</metaData>
            <metaData key="com.ibm.wmqfte.Priority">0</metaData>
        </metaDataSet>
    </transferSet>
    <statistics>
        <actualStartTime>2011-03-07T13:05:02.041Z</actualStartTime>
        <retryCount>0</retryCount>
        <numFileFailures>0</numFileFailures>
        <numFileWarnings>0</numFileWarnings>
    </statistics>
</transaction>
```

Destination agent is Connect:Direct bridge agent

Started:

```
<?xml version="1.0" encoding="UTF-8"?>
<transaction xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             ID="414d5120514d5f696b6b796f20202020a704654d2008e102"
             agentRole="sourceAgent"
             version="4.00"
             xsi:noNamespaceSchemaLocation="TransferLog.xsd"
             xmlns="">
    <action time="2011-03-07T10:29:44.854Z">started</action>
    <sourceAgent QMgr="QM_ASTEROID" agent="PALLAS" agentType="STANDARD">
        <systemInfo architecture="x86" name="Windows 7" version="6.1 build 7601 Service Pack 1"/>
    </sourceAgent>
    <destinationAgent QMgr="QM_ASTEROID" agent="VESTA"/>
    <originator>
        <hostName>belt.example.com.</hostName>
        <userID>sol</userID>
        <mqmdUserID>sol</mqmdUserID>
    </originator>
    <transferSet bytesSent="0" startTime="2011-03-07T10:29:44.854Z" total="1">
        <metaDataSet>
            <metaData key="com.ibm.wmqfte.SourceAgent">PALLAS</metaData>
            <metaData key="com.ibm.wmqfte.DestinationAgent">VESTA</metaData>
            <metaData key="com.ibm.wmqfte.MqmdUser">sol</metaData>
            <metaData key="com.ibm.wmqfte.OriginatingUser">sol</metaData>
            <metaData key="com.ibm.wmqfte.OriginatingHost">belt.example.com.</metaData>
            <metaData key="com.ibm.wmqfte.TransferId">414d5120514d5f696b6b796f20202020a704654d2008e102</metaData>
            <metaData key="com.ibm.wmqfte.Priority">0</metaData>
        </metaDataSet>
    </transferSet>
</transaction>
```

Progress:

```
<?xml version="1.0" encoding="UTF-8"?>
<transaction xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             ID="414d5120514d5f696b6b796f20202020a704654d2008e102"
             agentRole="sourceAgent"
             version="4.00"
             xsi:noNamespaceSchemaLocation="TransferLog.xsd"
             xmlns="">
    <action time="2011-03-07T10:29:46.682Z">progress</action>
    <sourceAgent QMgr="QM_ASTEROID" agent="PALLAS" agentType="STANDARD">
        <systemInfo architecture="x86" name="Windows 7" version="6.1 build 7601 Service Pack 1"/>
    </sourceAgent>
    <destinationAgent QMgr="QM_ASTEROID" agent="VESTA" agentType="CD_BRIDGE"
                      bridgeNode="CDNODE_VESTA" pnode="CDNODE_VESTA" snode="CDNODE_HYGIEA">
        <systemInfo architecture="x86" name="Windows 7" version="6.1 build 7601 Service Pack 1"/>
    </destinationAgent>
    <originator>
        <hostName>belt.example.com</hostName>
        <userID>sol</userID>
        <mqmdUserID>sol</mqmdUserID>
    </originator>
    <transferSet bytesSent="48" index="0" size="1" startTime="2011-03-07T10:29:44.854Z" total="1">
        <item mode="binary">
            <source disposition="leave" type="file">
                <file last-modified="2011-03-04T14:53:28.323Z" size="4">D:\AGENTS\PALLAS\test.txt</file>
                <checksum method="MD5">098f6bcd4621d373cade4e832627b4f6</checksum>
            </source>
            <destination processName="f2006965" processNumber="59" type="file">
                <file size="4">CDNODE_VESTA:D:/AGENTS/CDNODE_VESTA/test.txt</file>
                <checksum method="MD5">098f6bcd4621d373cade4e832627b4f6</checksum>
            </destination>
            <status resultCode="0"/>
        </item>
    </transferSet>
</transaction>
```

Completed:

```
<?xml version="1.0" encoding="UTF-8"?>
<transaction xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             ID="414d5120514d5f696b6b796f20202020a704654d2008e102"
             agentRole="sourceAgent"
             version="4.00"
             xsi:noNamespaceSchemaLocation="TransferLog.xsd"
             xmlns="">
    <action time="2011-03-07T10:29:46.698Z">completed</action>
    <sourceAgent QMgr="QM_ASTEROID" agent="PALLAS" agentType="STANDARD">
```

```
            <systemInfo architecture="x86" name="Windows 7" version="6.1 build 7601 Service Pack 1"/>
    </sourceAgent>
    <destinationAgent QMgr="QM_ASTEROID" agent="VESTA" agentType="CD_BRIDGE"
                        bridgeNode="CDNODE_VESTA" pnode="CDNODE_VESTA" snode="CDNODE_HYGIEA">
            <systemInfo architecture="x86" name="Windows 7" version="6.1 build 7601 Service Pack 1"/>
    </destinationAgent>
    <originator>
        <hostName>belt.example.com</hostName>
        <userID>sol</userID>
        <mqmdUserID>sol</mqmdUserID>
    </originator>
    <status resultCode="0">
        <supplement>BFGRP0032I: The file transfer request has successfully completed.</supplement>
    </status>
    <transferSet bytesSent="48" startTime="2011-03-07T10:29:44.854Z" total="1">
        <metaDataSet>
            <metaData key="com.ibm.wmqfte.SourceAgent">PALLAS</metaData>
            <metaData key="com.ibm.wmqfte.DestinationAgent">VESTA</metaData>
            <metaData key="com.ibm.wmqfte.MqmdUser">sol</metaData>
            <metaData key="com.ibm.wmqfte.OriginatingUser">sol</metaData>
            <metaData key="com.ibm.wmqfte.OriginatingHost">belt.example.com.</metaData>
            <metaData key="com.ibm.wmqfte.TransferId">414d5120514d5f696b6b796f20202020a704654d2008e102</metaData>
            <metaData key="com.ibm.wmqfte.Priority">0</metaData>
        </metaDataSet>
    </transferSet>
    <statistics>
        <actualStartTime>2011-03-07T10:29:45.010Z</actualStartTime>
        <retryCount>0</retryCount>
        <numFileFailures>0</numFileFailures>
        <numFileWarnings>0</numFileWarnings>
    </statistics>
</transaction>
```

Scheduled transfer log message formats

Every time a request for a scheduled file transfer is processed by the agent, a schedule log message is published to the coordination queue manager (on its SYSTEM.FTE/Log/*agent name*/*schedule ID* topic). This message conforms to the ScheduleLog.xsd XML schema.

Schema

The following schema describes which elements are valid in a schedule log XML message.

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:include schemaLocation="fteutils.xsd"/>

    <xsd:element name="schedulelog">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="originator"       type="hostUserIDType"
                             maxOccurs="1"            minOccurs="0"/>
                <xsd:element name="action"           type="actionType"
                             maxOccurs="1"            minOccurs="1"/>
                <xsd:element name="schedule"         type="scheduleType"
                             maxOccurs="1"            minOccurs="0"/>
                <xsd:element name="sourceAgent"      type="agentType"
                             maxOccurs="1"            minOccurs="0"/>
                <xsd:element name="destinationAgent" type="agentClientType"
                             maxOccurs="1"            minOccurs="0" />
                <xsd:element name="status"           type="statusType"
                             maxOccurs="1"            minOccurs="0"/>
                <xsd:element name="transferSet"      type="transferSetType"
                             maxOccurs="1"            minOccurs="0"/>
                <xsd:element name="job"              type="jobType"
                             maxOccurs="1"            minOccurs="0" />
            </xsd:sequence>
            <xsd:attribute name="version"   type="versionType" use="required"/>
            <xsd:attribute name="ID"        type="xsd:string"  use="required"/>
        </xsd:complexType>
    </xsd:element>

    <xsd:complexType name="actionType">
```

```
        <xsd:simpleContent>
            <xsd:extension base="actionEnumType">
                <xsd:attribute name="time" type="xsd:dateTime" use="required" />
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>

    <xsd:simpleType name="actionEnumType">
        <xsd:restriction base="xsd:token">
            <xsd:enumeration value="submit"/>
            <xsd:enumeration value="delete"/>
            <xsd:enumeration value="expire"/>
            <xsd:enumeration value="skipped"/>
        </xsd:restriction>
    </xsd:simpleType>

    <xsd:complexType name="transferSetType">
        <xsd:sequence>
            <xsd:element name="item"              type="itemType"
                        maxOccurs="unbounded"   minOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="size" type="xsd:int" use="required" />
        <xsd:attribute name="priority" type="priorityType" use="optional" />
    </xsd:complexType>

    <xsd:complexType name="itemType">
        <xsd:sequence>
            <xsd:element name="source"      type="fileSourceType"
                        maxOccurs="1"       minOccurs="1" />
            <xsd:element name="destination" type="fileDestinationType"
                        maxOccurs="1"       minOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="mode" type="modeType" use="required" />
        <xsd:attribute name="checksumMethod" type="checkSumMethod" use="required" />
    </xsd:complexType>

</xsd:schema>
```

Understanding the schedule log message

The elements and attributes used in the schedule log message are described:

`<schedulelog>`
 Group element that describes a single submitted scheduled file transfer.

Attribute	Description
version	Specifies the version of this element as detailed by WebSphere MQ Managed File Transfer.
ID	The unique identifier for the submitted schedule file transfer.

`<originator>`
 Group element that contains the elements specifying the originator of the request.

`<hostName>`
 The host name of the system where the source file is located.

`<userID>`
 The user ID that originated the file transfer.

`<mqmdUserID>`
 The WebSphere MQ user ID that was supplied in the message descriptor (MQMD)

<action>

Specifies the action to take with the scheduled transfer matching the ID attribute of <schedulelog> element. This element must be one of the following values:

- submit - new scheduled transfer
- delete - cancel schedule transfer
- expire - schedule transfer entry about to be processed
- skipped - a transfer that was scheduled cannot be started because the agent is offline. This message is logged when the agent becomes available to indicate the transfer was skipped.

Attribute	Description
time	Specifies the date and time the log entry was published (in date time format).

<sourceAgent>

Specifies the name of the agent on the system where the source file is located.

Attribute	Description
agent	Specifies the name of the agent.
QMgr	The name of the agent queue manager.

<destinationAgent>

Specifies the name of the agent on the system you want to transfer the file to.

Attribute	Description
agent	Specifies the name of the agent.
QMgr	The name of the agent queue manager.

<status>

The result code and supplement messages.

<transferSet>

Specifies a group of file transfers you want to perform together. During transmission <transferSet> is a group element containing <item> elements.

Attribute	Description
size	Specifies the number of transfer items.
priority	Priority level of the transfer. Priority is a value in the range 0-9, where 0 is the lowest priority. The default priority level is 0 and by default the transfer uses the priority level of the source agent.

<item>

Group element that contains elements specifying the source and destination file names and locations.

Attribute	Description
mode	Specifies the transfer mode as being either binary or text.
checksumMethod	Specifies the type of hash algorithm that generates the message digest to create the digital signature. Permitted values are MD5 or none

<source>

Group element that contains the <file> and <checksum> elements for the file on the source system.

Attribute	Description
recursive	Specifies that files are transferred recursively in subdirectories when the <source> element is a directory or contains wildcard characters.
disposition	Specifies the action that is taken on the <source> element when <source> has successfully been transferred to its destination. The valid options are as follows: • leave - the source files are left unchanged. • delete - the source files are deleted from the source system after the source file is successfully transferred.

<destination>

> Group element that contains the <file> and <checksum> elements for the file on the destination system.

Attribute	Description
type	The type of file or directory at the destination. The valid options are as follows: • file - specifies a file as the destination • directory - specifies a directory as the destination
exist	Specifies the action that is taken if a destination file exists on the destination system. The valid options are as follows: • error - reports an error and the file is not transferred. • overwrite - overwrites the existing destination file.

<file>

> Specifies the name of the file to transfer. Use the fully qualified path in the format consistent with your operating system, for example `C:/from/here.txt`. Do not use file URIs.

Attribute	Description
encoding	The encoding for a text file transfer.
EOL	Specifies the end of line marker. Permitted values are: • LF - line feed character only • CRLF - carriage return and line feed character sequence

<job>

> Group element that contains an element specifying job details. <job> is a user-defined job name identifier that is added to the log message when the transfer has started. This <job> element is the same as the <job> element that is included in the transfer request message, which is described in the following topic: "File transfer request message format" on page 803.
>
> **<name>**
> > The value of name can be any string.

Examples

Examples of XML messages that conform to this schema are provided for each of the following scheduled transfer actions:

• A scheduled transfer is created
• A scheduled transfer is canceled

- A schedule transfer expires

Transfers that are started by a schedule are logged in the same way as a standard transfer. For examples of log messages for transfers started by a schedule, see "Scheduled transfer log message examples" on page 642.

Schedule log examples:

Examples of the messages that are published to the SYSTEM.FTE topic with a topic string of Log/*agent_name*/*schedule_ID* when a scheduled transfer action occurs.

Scheduled transfer log message

Every time a request for a scheduled file transfer is processed by the agent, a schedule log message is published to the coordination queue manager (on its SYSTEM.FTE/Log/*agent name*/*schedule ID* topic). This message conforms to the ScheduleLog.xsd XML schema. For example:

```
<?xml version="1.0" encoding="UTF-8"?>
<schedulelog version="1.00" ID="5"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="ScheduleLog.xsd">
 <originator>
  <hostName>reportserver.com</hostName>
  <userID>USER1</userID>
 </originator>
 <action time="2008-11-23T21:32:01Z">submit</action>
 <schedule>
  <submit timebase="admin" timezone="Europe/London">2008-11-23T22:00</submit>
 </schedule>
 <sourceAgent agent="FTEAGENT" QMgr="QM1" />
 <destinationAgent agent="FTEAGENT" QMgr="QM1" />
 <status resultCode="0" />
 <transferSet size="1" priority="0">
  <item mode="binary" checksumMethod="MD5">
   <source recursive="false" disposition="leave">
    <file>c:\sourcefiles\source1.doc</file>
   </source>
   <destination type="file" exist="overwrite">
    <file>c:\destinationfiles\dest1.doc</file>
   </destination>
  </item>
 </transferSet>
</schedulelog>
```

This message is a log of the following information:
- Who originated the request
- When the request was submitted
- When the scheduled transfer starts
- The source and destination agent details
- The transfer specification

The ID attribute of the <schedulelog> element is a unique ID for this scheduled transfer (in the source agent). This ID is used to correlate schedule entries with the actual file transfers.

The <action> element value of submit confirms the request has been received.

Scheduled transfer cancel log message

When a request to cancel a pending scheduled file transfer is received by the agent, the following message is published to the SYSTEM.FTE/Log/*agent_name* topic:

```
<?xml version="1.0" encoding="UTF-8"?>
<schedulelog version="1.00" ID="5"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="ScheduleLog.xsd">
 <originator>
  <hostName>reportserver.com</hostName>
  <userID>USER1</userID>
 </originator>
 <action time="2008-11-23T21:56:27Z">delete</action>
 <status resultCode="0" />
</schedulelog>
```

The ID attribute value corresponds to the ID of the pending transfer request ID in the schedules message.

Scheduled transfer expire log message

When the current time matches the time of the earliest pending file transfer in the schedule list (as indicated by the value of the <next> element), a schedule log message is published to indicate that the scheduled transfer entry has expired:

```
<?xml version="1.0" encoding="UTF-8"?>
<schedulelog xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            version="4.00" ID="3"
            xsi:noNamespaceSchemaLocation="ScheduleLog.xsd">
    <originator>
        <hostName>example.com.</hostName>
        <userID>mqm</userID>
        <mqmdUserID>mqm</mqmdUserID>
    </originator>
    <action time="2011-01-26T13:03:26Z">expire</action>
    <sourceAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER"/>
    <status resultCode="0"/>
</schedulelog>
```

The <action> element value of "expire" confirms the schedule entry has now been removed from the schedule list and is being processed. A schedule message for the agent is published with the expired entry no longer present.

Related reference:

"Scheduled transfer log message formats" on page 650
Every time a request for a scheduled file transfer is processed by the agent, a schedule log message is published to the coordination queue manager (on its SYSTEM.FTE/Log/*agent name*/*schedule ID* topic). This message conforms to the `ScheduleLog.xsd` XML schema.

"Scheduled transfer log message examples" on page 642
When a transfer is in progress, messages are published to the SYSTEM.FTE topic with a topic string of Log/*agent_name*/*transfer_ID*. The XML examples show the log messages that are created when a file transfer occurs as a result of a schedule.

Monitor log message format

Monitor log messages are published to the SYSTEM.FTE topic with a topic string of Log/*agent_name*/*Monitors*/*monitor_name*/*monitor_ID*.

If you want to collect data or view monitor actions, set up a subscription to a wildcard topic tailored to the monitors that you are interested in. For example:

```
Log/#
```

or,

```
Log/agent_name/#
```

This subscription can be durable or non-durable. Durable subscriptions continue to exist when a subscribing application's connection to the queue manager is closed. Non-durable subscriptions exist only as long as a subscribing application's connection to the queue manager remains open.

The MonitorLog.xsd schema document is located in the *MQ_INSTALLATION_PATH*/mqft/samples/schema directory. The MonitorLog.xsd schema imports fteutils.xsd, which is in the same directory.

Schema

The following schema describes which elements are valid in a monitor log XML message.

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

    <xsd:include schemaLocation="fteutils.xsd"/>

    <xsd:element name="monitorLog">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="originator"        type="hostUserIDType"        maxOccurs="1" minOccurs="0"/>
                <xsd:element name="references"        type="referencesType"        maxOccurs="1" minOccurs="0"/>
                <xsd:element name="action"            type="monitorActionType"     maxOccurs="1" minOccurs="1"/>
                <xsd:element name="monitorAgent"      type="agentType"             maxOccurs="1" minOccurs="1"/>
                <xsd:element name="status"            type="statusType"            maxOccurs="1" minOccurs="1"/>
                <xsd:element name="monitorMetaData"   type="monitorMetaDataType"   maxOccurs="1" minOccurs="0"/>
                <xsd:element name="monitorExits"      type="exitGroupType"         maxOccurs="1" minOccurs="0"/>
                <xsd:element name="jobDetails"        type="jobType"               maxOccurs="1" minOccurs="0"/>
                <xsd:element name="taskXMLRequest"    type="taskXMLRequestType"    maxOccurs="1" minOccurs="0"/>
                <xsd:element name="monitorXMLRequest" type="monitorXMLRequestType"             maxOccurs="1" minOccurs="0"/>
            </xsd:sequence>
            <xsd:attribute name="version"     type="versionType" use="required"/>
            <xsd:attribute name="monitorName" type="xsd:string"  use="required"/>
            <xsd:attribute name="referenceId" type="xsd:string"  use="optional"/>
        </xsd:complexType>
    </xsd:element>

    <xsd:complexType name="monitorActionType">
        <xsd:simpleContent>
            <xsd:extension base="monitorActionEnumType">
                <xsd:attribute name="time" type="xsd:dateTime" use="required" />
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>

    <xsd:simpleType name="monitorActionEnumType">
        <xsd:restriction base="xsd:token">
            <xsd:enumeration value="create"/>
            <xsd:enumeration value="delete"/>
            <xsd:enumeration value="start"/>
            <xsd:enumeration value="stop"/>
            <xsd:enumeration value="triggerSatisfied"/>
            <xsd:enumeration value="triggerNotSatisfied"/>
            <xsd:enumeration value="triggerFail"/>
        </xsd:restriction>
    </xsd:simpleType>

    <xsd:complexType name="monitorMetaDataType">
        <xsd:sequence>
            <xsd:element name="originalMetaData" type="metaDataSetType"  maxOccurs="1" minOccurs="0"/>
            <xsd:element name="updatedMetaData"  type="metaDataSetType"  maxOccurs="unbounded" minOccurs="0"/>
        </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="taskXMLRequestType">
        <xsd:sequence>
            <xsd:element name="originalRequest" type="xsd:string"  maxOccurs="1" minOccurs="1"/>
            <xsd:element name="updatedRequest"  type="xsd:string"  maxOccurs="1" minOccurs="0"/>
        </xsd:sequence>
        <xsd:attribute name="taskId" type="xsd:string" use="required"/>
    </xsd:complexType>

    <xsd:complexType name="referencesType">
        <xsd:sequence>
            <xsd:element name="createRequest" type="xsd:string"  maxOccurs="1" minOccurs="0"/>
            <xsd:element name="taskRequest"   type="xsd:string"  maxOccurs="1" minOccurs="0"/>
        </xsd:sequence>
```

```
        </xsd:complexType>

        <xsd:complexType name="monitorXMLRequestType">
            <xsd:simpleContent>
                <xsd:extension base="xsd:string">
                    <xsd:attribute name="type" type="xmlContentEnumType" use="required" />
                </xsd:extension>
            </xsd:simpleContent>
        </xsd:complexType>

        <xsd:simpleType name="xmlContentEnumType">
            <xsd:restriction base="xsd:token">
                <xsd:enumeration value="escapedXML"/>
            </xsd:restriction>
        </xsd:simpleType>

</xsd:schema>
```

Understanding the monitor log message

The elements and attributes used in the monitor log messages are described in the following list:

`<monitorLog>`
> Group element containing the elements describe an action that has been performed by a monitor.

Attribute	Description
version	Required. The version of the monitor list message format.
monitorName	Required. The name of the monitor. Unique for the agent that the monitor is defined on.
referenceId	The ID of the monitor action.

`<originator>`
> Group element that contains the elements specifying the originator of the request.

`<hostName>`
> The host name of the system where the source file is located.

`<userID>`
> The user ID that originated the file transfer.

`<mqmdUserID>`
> Optional. The WebSphere MQ user ID that was supplied in the message descriptor (MQMD).

`<references>`
> References to the IDs of other messages associated with this monitor action.

`<createRequest>`
> The message ID of the XML request message that was used to create the monitor.

`<taskRequest>`
> The message ID of the XML request message that the monitor submits as a result of this action.

`<action>`
> The action that occurred, which this log message is associated with. The value inside the element can be one of the following: create, delete, start, stop, triggerSatisfied, triggerNotSatisfied, or triggerFail.

`<monitorAgent>`
> The agent that is monitoring the resource.

Attribute	Description
agent	Required. The name of the agent.

Attribute	Description
QMgr	Optional. The name of the queue manager that the agent connects to.
bridgeURL	Optional. If the agent is a protocol bridge agent, the URL of the protocol server.

<status>
> The status of the resource monitor action being logged.

Attribute	Description
resultCode	Required. The integer result code from the action.

<supplement>
> Additional information about the status of the resource monitor action being logged.

<monitorMetaData>
> Group element that contains the <originalMetaData> and <updatedMetaData> elements.

> **<originalMetaData>**
>> Element that contains one or more <metadata> elements that describe the metadata of the monitor before the action occurs.

> **<updatedMetaData>**
>> Element that contains one or more <metadata> elements that describe the metadata of the monitor after the action occurs.

> **<metadata>**
>> Defines a metadata key-value pair. The key is an attribute of the element; the value is the content of the element.

Attirbute	Description
key	The key of the metadata.

<monitorExits>
> Group element containing one or more <exit> elements.

<exits>
> Element describing an exit run by the resource monitor.

Attribute	Description
name	Required. The name of the resource monitor exit.

> **<status>**
>> The status of the resource monitor exit that is being logged.

Attribute	Description
resultCode	Required. The integer result code from the exit.

> **<supplement>**
>> Additional information about the status of the resource monitor exit that is being logged.

<jobDetails>
> Element containing a single <name> element.

> **<name>**
>> The name of the job..

\<taskXMLRequest\>
>   Group element that contains the \<originalRequest\> and \<updatedRequest\> elements.

Attribute	Description
taskId	The ID of the task request message.

>   **\<originalRequest\>**
>   >   Element that contains the escaped XML request message for the task that the monitor performs.
>
>   **\<updatedRequest\>**
>   >   Element that contains the updated escaped XML request message for the task that the monitor performs.

\<monitorXMLRequest\>
>   The monitor XML request.

Attribute	Description
type	Required. The format of the monitor XML request data inside of the \<monitorXMLRequest\> element. The only valid value is escapedXML.

Examples

Examples of XML messages that conform to this schema are provided for each of the following monitor actions:

- A monitor is created
- The condition of a monitor is satisfied when the monitor polls the resource
- The condition of a monitor is not satisfied when the monitor polls the resource
- A monitor is deleted

Related reference:

Examples of the messages that are published to the SYSTEM.FTE topic with a topic string of Log/*agent_name*/*monitor_ID* when a monitor action occurs.

Monitor log examples:

Examples of the messages that are published to the SYSTEM.FTE topic with a topic string of Log/*agent_name*/*monitor_ID* when a monitor action occurs.

Monitor created log message

```xml
<?xml version="1.0" encoding="UTF-8"?>
<monitorLog xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            version="4.00"
            monitorName="MONITORTWO"
            referenceId="414d51205553322e42494e44494e47538b0f404d04410020"
            xsi:noNamespaceSchemaLocation="MonitorLog.xsd">
    <originator>
        <hostName>example.com.</hostName>
        <userID>mqm</userID>
        <mqmdUserID>mqm</mqmdUserID>
    </originator>
    <references>
        <createRequest>414d51205553322e42494e44494e47538b0f404d04410020</createRequest>
    </references>
    <action time="2011-01-26T12:41:24Z">start</action>
    <monitorAgent agent="AGENT_JUPITER" QMgr="QM_JUPITER"/>
    <status resultCode="0"/>
</monitorLog>
```

Monitor condition satisfied log message

```
<?xml version="1.0" encoding="UTF-8"?>
    <monitorLog xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                version="4.00"
                monitorName="MONITORONE"
                referenceId="414d51205553322e42494e44494e47538b0f404d09430020"
                xsi:noNamespaceSchemaLocation="MonitorLog.xsd">
    <originator>
        <hostName>example.com.</hostName>
        <userID>mqm</userID>
        <mqmdUserID>mqm</mqmdUserID>
    </originator>
    <references>
        <createRequest>414d51205553322e42494e44494e47538b0f404d09430020</createRequest>
    </references>
    <action time="2011-01-26T12:56:46Z">triggerSatisfied</action>
    <monitorAgent agent="US2.BINDINGS.FILE" QMgr="US2.BINDINGS"/>
    <status resultCode="0"/>
    <monitorMetaData>
        <originalMetaData>
            <metaData key="AGENTNAME">AGENT_JUPITER</metaData>
            <metaData key="LASTMODIFIEDDATEUTC">2011-01-26</metaData>
            <metaData key="CURRENTTIMESTAMPUTC">20110126125646793</metaData>
            <metaData key="CURRENTTIMESTAMP">20110126125646793</metaData>
            <metaData key="LASTMODIFIEDDATE">2011-01-26</metaData>
            <metaData key="FILENAME">new.completed</metaData>
            <metaData key="LASTMODIFIEDTIMEUTC">12.56</metaData>
            <metaData key="LASTMODIFIEDTIME">12.56</metaData>
            <metaData key="FILESIZE">0</metaData>
            <metaData key="FILEPATH">/srv/nfs/incoming/new.completed</metaData>
        </originalMetaData>
        <updatedMetaData>
            <metaData key="AGENTNAME">AGENT_JUPITER</metaData>
            <metaData key="LASTMODIFIEDDATEUTC">2011-01-26</metaData>
            <metaData key="CURRENTTIMESTAMPUTC">20110126125646793</metaData>
            <metaData key="CURRENTTIMESTAMP">20110126125646793</metaData>
            <metaData key="LASTMODIFIEDDATE">2011-01-26</metaData>
            <metaData key="FILENAME">new.completed</metaData>
            <metaData key="LASTMODIFIEDTIMEUTC">12.56</metaData>
            <metaData key="LASTMODIFIEDTIME">12.56</metaData>
            <metaData key="FILESIZE">0</metaData>
            <metaData key="FILEPATH">/srv/nfs/incoming/new.completed</metaData>
        </updatedMetaData>
    </monitorMetaData>
    <taskXMLRequest taskId="null">
        <originalRequest>&lt;?xml version="1.0" encoding="UTF-8"?&gt;&lt;request
                    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.00"
                    xsi:noNamespaceSchemaLocation="FileTransfer.xsd"&gt;&lt;managedTransfer&gt;
                    &lt;originator&gt;&lt;hostName&gt;example.com.&lt;/hostName&gt;
                    &lt;userID&gt;mqm&lt;/userID&gt;&lt;/originator&gt;
                    &lt;sourceAgent QMgr="QM_JUPITER" agent="AGENT_JUPITER"/&gt;
                    &lt;destinationAgent QMgr="QM_JUPITER" agent="AGENT_SATURN"/&gt;
                    &lt;transferSet&gt;&lt;item checksumMethod="MD5" mode="binary"&gt;
                    &lt;source disposition="leave" recursive="false"&gt;
                    &lt;file&gt;/srv/nfs/incoming/*.txt&lt;/file&gt;&lt;/source&gt;
                    &lt;destination exist="error" type="directory"&gt;
                    &lt;file&gt;/srv/backup&lt;/file&gt;&lt;/destination&gt;
                    &lt;/item&gt;&lt;/transferSet&gt;&lt;/managedTransfer&gt;&lt;/request&gt;
        </originalRequest>
        <updatedRequest>&lt;?xml version="1.0" encoding="UTF-8"?&gt;&lt;request
                    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.00"
                    xsi:noNamespaceSchemaLocation="FileTransfer.xsd"&gt;&lt;managedTransfer&gt;
                    &lt;originator&gt;&lt;hostName&gt;example.com.&lt;/hostName&gt;
                    &lt;userID&gt;mqm&lt;/userID&gt;&lt;/originator&gt;
                    &lt;sourceAgent QMgr="QM_JUPITER" agent="AGENT_JUPITER"/&gt;
                    &lt;destinationAgent QMgr="QM_JUPITER" agent="AGENT_SATURN"/&gt;
                    &lt;transferSet&gt;&lt;item checksumMethod="MD5" mode="binary"&gt;
                    &lt;source disposition="leave" recursive="false"&gt;
                    &lt;file&gt;/srv/nfs/incoming/*.txt&lt;/file&gt;
                    &lt;/source&gt;&lt;destination exist="error" type="directory"&gt;
                    &lt;file&gt;/srv/backup&lt;/file&gt;&lt;/destination&gt;
                    &lt;/item&gt;&lt;/transferSet&gt;&lt;/managedTransfer&gt;&lt;/request&gt;
        </updatedRequest>
    </taskXMLRequest>
</monitorLog>
```

Monitor condition not satisfied log message

```
<?xml version="1.0" encoding="UTF-8"?>
<monitorLog xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            version="4.00"
            monitorName="MONITORONE"
            referenceId="414d51205553322e42494e44494e47538b0f404d09430020"
            xsi:noNamespaceSchemaLocation="MonitorLog.xsd">
    <originator>
        <hostName>example.com.</hostName>
        <userID>mqm</userID>
        <mqmdUserID>mqm</mqmdUserID>
    </originator>
    <references>
        <createRequest>414d51205553322e42494e44494e47538b0f404d09430020</createRequest>
    </references>
    <action time="2011-01-26T12:58:46Z">triggerNotSatisfied</action>
    <monitorAgent agent="US2.BINDINGS.FILE" QMgr="US2.BINDINGS"/>
    <status resultCode="0"/>
</monitorLog>
```

Monitor deleted log message

```
<?xml version="1.0" encoding="UTF-8"?>
<lst:monitorList xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                 xmlns:lst="http://www.ibm.com/xmlns/wmqfte/7.0.1/MonitorDefinition"
                 version="4.00"
                 agent="AGENT_JUPITER"
                 monitor="MONITORONE"
                 xsi:schemaLocation="http://www.ibm.com/xmlns/wmqfte/7.0.1/MonitorDefinition MonitorList.xsd">
    <status state="deleted"/>
    <configuration>
        <description/>
        <resources>
            <directory recursionLevel="0" id="">/srv/nfs/incoming</directory>
        </resources>
        <triggerMatch>
            <conditions>
                <condition>
                    <name/>
                    <resource id=""/>
                    <fileMatch>
                        <pattern>*.completed</pattern>
                    </fileMatch>
                </condition>
            </conditions>
        </triggerMatch>
        <tasks>
            <task>
                <name/>
                <description/>
                <taskXML>&lt;?xml version="1.0" encoding="UTF-8"?&gt;&lt;request
                        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="4.00"
                        xsi:noNamespaceSchemaLocation="FileTransfer.xsd"&gt;&lt;managedTransfer&gt;
                        &lt;originator&gt;&lt;hostName&gt;example.ibm.com.&lt;/hostName&gt;
                        &lt;userID&gt;mqm&lt;/userID&gt;&lt;/originator&gt;
                        &lt;sourceAgent QMgr="QM_JUPITER" agent="AGENT_JUPITER"/&gt;
                        &lt;destinationAgent QMgr="QM_JUPITER" agent="AGENT_SATURN"/&gt;
                        &lt;transferSet&gt;&lt;item checksumMethod="MD5" mode="binary"&gt;
                        &lt;source disposition="leave" recursive="false"&gt;
                        &lt;file&gt;/srv/nfs/incoming/*.txt&lt;/file&gt;&lt;/source&gt;
                        &lt;destination exist="error" type="directory"&gt;
                        &lt;file&gt;/srv/backup&lt;/file&gt;&lt;/destination&gt;
                        &lt;/item&gt;&lt;/transferSet&gt;&lt;/managedTransfer&gt;&lt;/request&gt;
                </taskXML>
            </task>
        </tasks>
    </configuration>
    <pollInterval units="minutes">1</pollInterval>
    <batch maxSize="1"/>
</lst:monitorList>
```

File transfer request message format

File transfers are initiated by XML messages arriving at an agent command queue, typically as a result of a user issuing a file transfer command or by using the WebSphere MQ Explorer plug-in. The transfer request XML must conform to the `FileTransfer.xsd` schema and have the <request> element as the root element. The `FileTransfer.xsd` schema document is located in the `MQ_INSTALLATION_PATH`/mqft/samples/ schema directory. The `FileTransfer.xsd` schema imports `fteutils.xsd`, which is in the same directory.

File transfer messages can have one of following three root elements:

* <request> - for new file transfer requests, managed call requests, or deleting scheduled transfers that are pending
* <cancel> - for canceling file transfers in progress
* <transferSpecifications> - for specifying multiple transfer file groups, used by the **fteCreateTransfer** command

For information about specifying multiple transfer groups by using the <transferSpecifications> element, see Using transfer definition files.

Schema

The following schema describes which elements are valid in a transfer request XML message.

V 7.5.0.1

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

<xsd:include schemaLocation="fteutils.xsd"/>

    <!--
        Defines the request of a managed transfer and version number
        <request version="1.00" ...
            <managedTransfer>
                ...
            </managedTransfer>
        </request>
     -->
    <xsd:element name="request">
        <xsd:complexType>
            <xsd:choice>
                <xsd:element name="managedTransfer" type="managedTransferType"/>
                <xsd:element name="deleteScheduledTransfer" type="deleteScheduledTransferType" />
                <xsd:element name="managedCall" type="managedCallType"/>
            </xsd:choice>
            <xsd:attribute name="version" type="versionType" use="required" />
        </xsd:complexType>
    </xsd:element>

    <!--
        Defines the cancel request of a managed transfer and version number
        <cancel version="1.00"
                xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
            <originator>
                <hostName>myMachine</hostName>
                <userID>myUserId</userID>
            </originator>      - Delete a scheduled transfer.

            <transfer>
                Transfer ID to Cancel
            </transfer>
        </cancel>
     -->
    <xsd:element name="cancel">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="originator" type="hostUserIDType" maxOccurs="1" minOccurs="1" />
                <xsd:choice>
                    <xsd:element name="transfer" type="IDType" maxOccurs="1" minOccurs="1" />
                    <xsd:element name="call"     type="IDType" maxOccurs="1" minOccurs="1" />
                </xsd:choice>
                <xsd:element name="reply"       type="replyType"       maxOccurs="1" minOccurs="0" />
```

```
            </xsd:sequence>
            <xsd:attribute name="version" type="versionType" use="required" />
        </xsd:complexType>
</xsd:element>

<!--
    Defines the transfer definition element structure.
        <transferSpecifications>
            <item ...
            <item ...
        </transferSpecifications>
 -->
<xsd:element name="transferSpecifications">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="item" type="itemType" minOccurs="1" maxOccurs="unbounded" />
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<!--
    Define a managed transfer of an instigator and request
    <managedTransfer>

        <originator>
            ...
        </originator>

        <schedule>
            <submit timebase="source"|"UTC">2008-12-07T16:07"</submit>
            <repeat>
                <frequency interval="hours">2</frequency>
                <expireTime>2008-12-0816:07</expireTime>
            </repeat>
        </schedule>

        <sourceAgent agent="here" QMgr="near" />
        <destinationAgent agent="there" QMgr="far" />

        <trigger>
        ...
        </trigger>

        <transferSet>
            ...
        </transferSet>
    </managedTransfer>
 -->

 <xsd:complexType name="managedTransferType">
    <xsd:sequence>
        <xsd:element name="originator"       type="origTransferRequestType"  maxOccurs="1" minOccurs="1"/>
        <xsd:element name="schedule"         type="scheduleType"     maxOccurs="1" minOccurs="0"/>
        <xsd:element name="sourceAgent"      type="agentType"        maxOccurs="1" minOccurs="1"/>
        <xsd:element name="destinationAgent" type="agentClientType"  maxOccurs="1" minOccurs="1" />
        <xsd:element name="trigger"          type="triggerType"      maxOccurs="1" minOccurs="0" />
        <xsd:element name="reply"            type="replyType"        maxOccurs="1" minOccurs="0" />
        <xsd:element name="transferSet"      type="transferSetType"  maxOccurs="1" minOccurs="1" />
        <xsd:element name="job"              type="jobType"          maxOccurs="1" minOccurs="0"/>
    </xsd:sequence>
</xsd:complexType>

<!--
    This is a modified form of origRequestType which is used on a managed transfer request.
    The hostName and userID are mandatory attributes in this case.
 -->
<xsd:complexType name="origTransferRequestType">
    <xsd:sequence>
        <xsd:element name="hostName" type="xsd:string" minOccurs="1" maxOccurs="1"/>
        <xsd:element name="userID" type="xsd:string" minOccurs="1" maxOccurs="1"/>
        <xsd:element name="mqmdUserID" type="xsd:string" minOccurs="0" maxOccurs="1"/>
        <xsd:element name="webBrowser" type="xsd:string" minOccurs="0" maxOccurs="1"/>
        <xsd:element name="webUserID" type="xsd:string" minOccurs="0" maxOccurs="1"/>
    </xsd:sequence>
</xsd:complexType>

<!--
    Defines the transferset of source and destination agent and one or more files
    <transferset priority="1">
```

```
        <metaDataSet>
            <metaData key="keyname">keyvalue</metaData>
            <metaData key="keyname">keyvalue</metaData>
        </metaDataSet>

        <item>
            ...
        </item>
    </transferset>
 -->
<xsd:complexType name="transferSetType">
    <xsd:sequence>
        <xsd:element name="metaDataSet"        type="metaDataSetType" maxOccurs="1" minOccurs="0" />
        <xsd:element name="preSourceCall"      type="commandActionType" maxOccurs="1" minOccurs="0" />
        <xsd:element name="postSourceCall"     type="commandActionType" maxOccurs="1" minOccurs="0" />
        <xsd:element name="preDestinationCall" type="commandActionType" maxOccurs="1" minOccurs="0" />
        <xsd:element name="postDestinationCall" type="commandActionType" maxOccurs="1" minOccurs="0" />
        <xsd:element name="item" type="itemType" maxOccurs="unbounded" minOccurs="0" />
    </xsd:sequence>
    <xsd:attribute name="priority" type="priorityType" use="optional" />
</xsd:complexType>


<!--
    Define a file pair with source and destination
    <item mode=[binary|text]>
        <source recursive="false" disposition="leave">
            <file>filename</file>
        </source>

        <destination type="file" exist="error">
            <file>filename</file>
        </destination>

    </item>
-->
<xsd:complexType name="itemType">
    <xsd:sequence>
        <xsd:element name="source"      type="fileSourceType"       maxOccurs="1" minOccurs="1" />
        <xsd:element name="destination" type="fileDestinationType"   maxOccurs="1" minOccurs="1" />
    </xsd:sequence>
    <xsd:attribute name="mode" type="modeType" use="required" />
    <xsd:attribute name="checksumMethod" type="checkSumMethod" use="required" />
</xsd:complexType>

<!--
    Defines the request to delete scheduled file transfer.
    <deleteScheduledTransfer>
        <originator>
            <delete>
                <hostName>myMachine</hostName>
                <userID>myUserId</userID>
            </delete>
        </originator>
        <ID>56</ID>
    </deleteScheduledTransfer>
 -->
<xsd:complexType name ="deleteScheduledTransferType">
    <xsd:sequence>
        <xsd:element name="originator" type="origDeleteType" maxOccurs="1" minOccurs="1" />
        <xsd:element name="ID"         type="idType"         maxOccurs="1" minOccurs="1" />
        <xsd:element name="reply"      type="replyType"      maxOccurs="1" minOccurs="0" />
    </xsd:sequence>
</xsd:complexType>


<xsd:complexType name="managedCallType">
    <xsd:sequence>
        <xsd:element name="originator"  type="origRequestType"  maxOccurs="1" minOccurs="1"/>
        <xsd:element name="agent"       type="agentType"        maxOccurs="1" minOccurs="1"/>
        <xsd:element name="reply"       type="replyType"        maxOccurs="1" minOccurs="0" />
        <xsd:element name="transferSet" type="callTransferSetType"  maxOccurs="1" minOccurs="1" />
        <xsd:element name="job"         type="jobType"          maxOccurs="1" minOccurs="0"/>
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="callTransferSetType">
    <xsd:sequence>
        <xsd:element name="metaDataSet" type="metaDataSetType" maxOccurs="1" minOccurs="0" />
```

```
            <xsd:element name="call" type="commandActionType" maxOccurs="1" minOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="priority" type="priorityType" use="optional" />
    </xsd:complexType>
</xsd:schema>
```

Understanding the transfer request message

The elements and attributes used in transfer request messages are described in the following list:

Element descriptions

`<request>`
Group element containing all the elements required to specify a file transfer request.

Attribute	Description
version	Specifies the version of this element as supplied by WebSphere MQ Managed File Transfer.

`<managedTransfer>`
Group element that contains all the elements required for a single file transfer or single group of file transfers.

`<deleteScheduledTransfer>`
Group element that contains originator and ID information to cancel a schedule transfer.

`<managedCall>`
Group element that contains all the elements required for a single managed call of a program or executable.

`<ID>`
Unique identifier that specifies the transfer request to delete from the list of pending scheduled transfers.

`<originator>`
Group element that contains the elements specifying the originator of the request.

`<hostName>`
The host name of the system where the source file is located.

`<userID>`
The user ID that originated the file transfer.

`<mqmdUserID>`
Optional. The WebSphere MQ user ID that was supplied in the message descriptor (MQMD).

`<schedule>`
Group element describing the scheduled time for the file transfer, the repeat behavior, and when the next occurrence is due.

`<submit>`
Specifies the date and time that the scheduled transfer is due to start.

Attribute	Description
timebase	Specifies which time zone to use. This attribute can have one of the following values: • source - use the time zone of the source agent • admin - use the time zone of the administrator issuing the command • UTC - use Coordinated Universal Time

Attribute	Description
timezone	The time zone description according to the timebase value

`<repeat>`

Group element that contains details about how often a scheduled transfer repeats, how many times a scheduled transfer repeats, and when a scheduled transfer stops repeating.

`<frequency>`

The time period that must elapse before the transfer repeats.

Attribute	Description
interval	The interval units, which must be one of the following values: • minutes • hours • days • weeks • months • years

`<expireTime>`

Optional element that specifies the date and time that a repeating scheduled transfer stops. This element and the <expireCount> element are mutually exclusive.

`<expireCount>`

Optional element that specifies the number of times the scheduled file transfer occurs before stopping. This element and the <expireTime> element are mutually exclusive.

`<sourceAgent>`

Specifies the name of the agent on the system where the source file is located.

Attribute	Description
agent	Specifies the name of the agent.
QMgr	The name of the agent queue manager.

`<destinationAgent>`

Specifies the name of the agent on the system you want to transfer the file to.

Attribute	Description
agent	Specifies the name of the agent.
QMgr	The name of the agent queue manager.
hostName	The host name or IP address of the agent queue manager.
portNumber	The port number used for client connections to the destination agent queue manager.
channel	The channel name used to connect to the destination agent queue manager.

`<trigger>`

Optional element that specifies a condition that must be true for the file transfer to take place.

Attribute	Description
log	A flag indicating whether trigger failures are logged. The valid values are as follows: • yes - log entries are created for failed triggered transfers • no - log entries are not created for failed triggered transfers

`<fileExist>`

Specifies a comma-separated list of file names located on the same system as the source agent. If a file in this name list satisfies the condition of the trigger, the transfer occurs. This element and the <fileSize> element are mutually exclusive.

Attribute	Description
comparison	Indicates how to evaluate source file names against the name list. The valid values are as follows: • = at least one file name in the name list must match • != a minimum of one of the files in the name list does not exist
value	Indicates the comparison type: • exist: file must exist

`<fileSize>`

Specifies a comma-separated list of file names located on the same system as the source agent. If a file in this name list satisfies the condition of the trigger, the transfer occurs. This element and the <fileExist> element are mutually exclusive.

Attribute	Description
comparison	Indicates how to evaluate source file names against the name list. The valid value is as follows: • >= one of the file names in the name list exists and has a minimum size as specified in the value attribute
value	File size specified as an integer value with units specified as one of the following: • B - bytes • KB - kilobytes • MB - megabytes • GB - gigabytes (the units value is not case-sensitive)

`<reply>`

Specifies the name of the temporary reply queue generated for synchronous file transfers (specified with the **-w** parameter on the command line). The name of the queue is defined by the key **dynamicQueuePrefix** in the command.properties configuration file or the default of WMQFTE.* if not specified.

Attribute	Description
QMGR	The name of the command queue manager on which the temporary dynamic queue is generated to receive replies.

Attribute	Description
persistent	Whether the message written to the reply queue is persistent. The valid values are as follows: • true - the message is persistent • false - the message is not persistent • qdef - the persistence of the message is defined by the properties of the reply queue The default value is false.

`<transferSet>`

Specifies a group of file transfers you want to perform together or a group of managed calls that you want to perform together. During transmission <transferSet> is a group element containing <item> elements.

Attribute	Description
priority	Priority level of the transfer. Priority is a value in the range 0-9, where 0 is the lowest priority. The default priority level is 0 and by default the transfer uses the priority level of the source agent.

`<metaDataSet>`

Optional group element containing one or more metadata items.

`<metaData>`

Specifies the user-defined metadata that is passed to the exit points called by the agent. The element contains the metadata value as a string.

Attribute	Description
key	Metadata name as a string

`<call>`

Group element that contains <command> elements specifying the program or executable to call.

`<command>`

Specifies the program or executable to call. The command must be located on the agent command path. For more information, see Table 34 on page 550. This element can contain optional <argument> elements.

Attribute	Description
name	The name of the command.
successRC	The successful return code that this command returns. Default is 0.
retryCount	The number of times that the command is to be retried if it fails.
retryWait	The time, in seconds, to wait between retries of the command.
type	The type of program to be called. The valid values are antscript, jcl, or executable.

`<argument>`

Specifies an argument to pass to the command.

`<item>`

Group element that contains elements specifying the source and destination file names and locations.

Attribute	Description
mode	Specifies the transfer mode as either binary or text.
checksumMethod	Specifies the type of hash algorithm that generates the message digest to create the digital signature. The valid values are MD5 or none.

`<source>`

Group element that specifies files on the source system and whether they are removed after the transfer completes

Attribute	Description
recursive	Specifies that files are transferred recursively in subdirectories when the <source> element is a directory or contains wildcard characters.
disposition	Specifies the action that is taken on the <source> element when <source>has successfully been transferred to its destination. The valid values are as follows: • leave - the source files are left unchanged. • delete - the source files are deleted from the source system after the source file is successfully transferred.

`<file>`

Specifies the transfer source, which can be a file or a directory name. Use the fully qualified path in the format consistent with your operating system, for example `C:/from/here.txt`. Do not use file URIs.

Attribute	Description
alias	Specifies an alias for the source file. This alias is the name of the source file, excluding any directory path specified for the transfer.
EOL	Specifies the end of line marker for text transfers. Valid values are: • LF - line feed character only • CRLF - carriage return and line feed character sequence
encoding	The encoding of the source file for a text file transfer.
delimiterType	Specifies the type of delimiter that is included in the destination file after individual message data. The valid values is as follows: • binary - a hexadecimal delimiter This attribute is available only if you have enabled the V7.0.4.1 function.

Attribute	Description
delimiterPosition	Specifies the position to insert delimiters when writing record-oriented source file records to a normal file. The valid values are as follows: • prefix - the delimiter is inserted into the destination file before the data from each source record-oriented file record. • postfix - the delimiter is inserted into the destination file after the data from each source record-oriented file record.
includeDelimiterInFile	Specifies whether to include a delimiter between records in record-oriented source files.

\<queue>

When used with the \<source> element, specifies the name of the queue to transfer from, which must be located on the source agent queue manager. Use the format *QUEUE*. Do not include the queue manager name, the queue must be present on the source agent queue manager. You cannot use the \<queue> element inside the \<source> element, if you have used it inside of the \<destination> element.

Attribute	Description
useGroups	Specifies whether to transfer only the first complete group of messages from the source queue. The valid values are as follows: • true - transfer only the first complete group of messages • false - transfer all messages on the source queue
groupId	Specifies the group of messages to read from the source queue. This attribute is valid only when the value of the useGroups attribute is true.
delimiterType	Specifies the type of delimiter that is included in the destination file after individual message data. The valid values are as follows: • text - a text or Java literal delimiter • binary - a hexadecimal delimiter
delimiter	Specifies the delimiter that is included in the destination file between individual message data.
delimiterPosition	Specifies whether the delimiter is included in the destination file before or after individual message data. The valid values are as follows: • prefix - the delimiter is included before the data • postfix - the delimiter is included after the data
encoding	Specifies the source queue encoding.

Attribute	Description
waitTime	Specifies the time, in seconds, for the source agent to wait for either: • a message to appear on the source queue, if the queue is empty or has become empty • a complete group to appear on the source queue, if the useGroups attribute has been set to true For information about setting the waitTime value, see "Guidance for specifying a wait time on a message-to-file transfer" on page 706.

`<destination>`

Group element that specifies the destination and the behavior if files exist at the destination agent.

You can specify only one of <file> and <queue> as a child element of destination.

Attribute	Description
type	The type of destination. The valid values are as follows: • file - specifies a file as the destination • directory - specifies a directory as the destination • queue - specifies a WebSphere MQ queue as the destination • filespace- specifies a file space as the destination The options file and directory are valid only when the **<destination>** element has a child element of **<file>**. The option queue is valid only when the **<destination>** element has a child element of **<queue>**. The option filespace is valid only when the **<destination>** element has a child element of **<filespace>**.
exist	Specifies the action that is taken if a destination file exists on the destination system. The valid values are as follows: • error - reports an error and the file is not transferred. • overwrite - overwrites the existing destination file. This attribute is not valid if the **<destination>** element has a child element of **<queue>** or **<filespace>**.

`<file>`

Specifies the transfer destination, which can be a file or a directory name. Use the fully qualified path in the format consistent with your operating system, for example `C:/from/here.txt`. Do not use file URIs.

Attribute	Description
alias	Specifies an alias for the destination file. This alias is the name of the source file, excluding any directory path specified for the transfer.
encoding	The encoding of the destination file for a text file transfer.

Attribute	Description
EOL	Specifies the end of line marker for text transfers. Valid values are: • LF - line feed character only • CRLF - carriage return and line feed character sequence

<queue>

> When used with the <destination> element, specifies the name of the queue to transfer to, which can be located on any queue manager that is connected to the destination agent queue manager. Use the format *QUEUE@QM* where *QUEUE* is the name of the queue to put the messages on and *QM* is the queue manager where the queue is located. You cannot use the <queue> element inside the <destination> element, if you have used it inside of the <source> element.

Attribute	Description
delimiter	The delimiter to split the file into multiple messages.
delimiterType	Specifies the type of delimiter. The valid values are as follows: • text - a Java regular expression • binary - a sequence of hexadecimal bytes • size - a number of bytes, kibibytes, or mebibytes. For example, 1 B, 1 K, or 1 M.
delimiterPosition	Specifies whether the delimiter is expected before or after the data to include in individual messages. The valid options are as follows: • prefix - the delimiter is expected before the data • postfix - the delimiter is expected after the data
includeDelimiterInMessage	A boolean specifying whether to include the delimiters that were used to split the file into multiple messages at the end of the messages.
encoding	Specifies the destination queue encoding.
persistent	Specifies whether the messages are persistent. The valid values are as follows: • true - the messages are persistent • false - the messages are not persistent • qdef - the persistence value of the messages is defined by the settings on the destination queue
setMqProps	A boolean specifying whether WebSphere MQ message properties are set on the first message in a file, and any messages written to the queue when an error occurs.
unrecognisedCodePage	Specifies whether a text mode transfer fails or conversion is performed, if the code page of the data is not recognized by the destination queue manager. The valid values are as follows: • fail - the transfer reports a failure • binary - the data is converted to the destination code page and the WebSphere MQ message header describing the format of the data is set to MQFMT_NONE. The default behavior is fail.

<filespace>

Group element specifying the name of the file space to transfer to.

<name>

When used with the <filespace> element, the value of this element specifies the name of the file space.

<attributes>

Optional group element that contains one or more <attribute> elements to specify distribution attribute information if you are transferring files to a IBM 4690 store controller.

<attribute>

Optional element that specifies file distribution attributes. Specify either the symbolic or numeric value.

Table 53. Valid values for file distribution attributes in WebSphere MQ Managed File Transfer

Symbolic value	Numeric value	Description
DIST(LOCAL)	DIST(1)	Local file
DIST(MIRRORED,UPDATE)	DIST(2)	Mirrored file, distribute at update
DIST(MIRRORED, CLOSE)	DIST(3)	Mirrored file, distribute at close
DIST(COMPOUND,UPDATE)	DIST(4)	Compound file, distribute at update
DIST(COMPOUND,CLOSE	DIST(5)	Compound file, distribute at close

For more information about distribution attributes for WebSphere MQ Managed File Transfer on IBM 4690, see "File distribution attributes" on page 87.

<preSourceCall>

Group element specifying a command to call at the source of the transfer, before the transfer starts.

<postSourceCall>

Group element specifying a command to call at the source of the transfer, after the transfer completes.

<preDestinationCall>

Group element specifying a command to call at the destination of the transfer, before the transfer starts.

<postDestinationCall>

Group element specifying a command to call at the destination of the transfer, after the transfer completes.

<command>

When used with the <preSourceCall>, <postSourceCall>, <preDestinationCall>, or <postDestinationCall> element, this element specifies the command to be called. The command must be located on the agent command path. For more information, see Table 34 on page 550.

Attribute	Description
name	The name of the command to run.
successRC	The return code that is expected if the command runs successfully.

<argument>

When used with the <command> element, this element specifies an argument to be passed in to the command. You can have any number of <argument> elements inside a <command> element.

<job>

Optional group element containing job information for the entire transfer specification. <job> is a user-defined job name identifier that is added to the log message when the transfer has started. This

<job> element is the same as the <job> element that appears in the transfer log message, which is described in the following topic: "File transfer log message formats" on page 626.

<name>
When used with the <job> element, the value of this element specifies the name of the job.

<transferSpecifications>
Group element that contains <item> elements for multiple transfer groups. See Using transfer definition files for further details about how to use this element.

<cancel>
Group element containing all the elements required to cancel a file transfer in progress.

Attribute	Description
version	Specifies the version of this element as supplied by WebSphere MQ Managed File Transfer.

<transfer>
When used with the <cancel> element, the value of this element specifies the transfer request ID to be canceled.

<job>
Group element containing job information.

<jobName>
Specifies logical job identifier.

File transfer cancel message format

A file transfer request returns a 48-character ID that identifies the transfer for a specific agent. This ID is used to cancel transfers.

Understanding the transfer cancel message

The elements and attributes used in transfer cancel messages are described:

<cancel>
Group element containing all the elements required to cancel a file transfer in progress.

Attribute	Description
version	Specifies the version of this element as supplied by WebSphere MQ Managed File Transfer.

<originator>
Group element that contains the elements specifying the originator of the request.

<hostName>
The host name of the system where the source file is located.

<userID>
The user ID that originated the file transfer.

<mqmdUserID>
Optional. The WebSphere MQ user ID that was supplied in the message descriptor (MQMD).

<transfer>
When used with the <cancel> element, the value of this element specifies the transfer request ID to be canceled.

<job>
Optional. Group element containing job information.

\<jobName\>
> Specifies logical job identifier.

Examples

Examples of XML messages that conform to this schema are provided for each of the following requests:

- Create a file transfer
- Create an asynchronous file transfer request
- Cancel a file transfer
- Create a scheduled transfer
- Delete a scheduled transfer
- Create a managed call
- Create a file transfer that includes managed calls

Transfer request examples:

Examples of the messages that you can put on the agent command queue to request that the agent create or cancel a transfer.

Create transfer request

```
<?xml version="1.0" encoding="UTF-8"?>
<request xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
version="4.00"
xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
    <managedTransfer>
        <originator>
            <hostName>example.com.</hostName>
            <userID>mqm</userID>
        </originator>
        <sourceAgent QMgr="QM_JUPITER" agent="AGENT_JUPITER"/>
        <destinationAgent QMgr="QM_JUPITER" agent="AGENT_JUPITER"/>
        <transferSet>
            <item checksumMethod="MD5" mode="binary">
            <source disposition="leave" recursive="false">
                <file>/etc/passwd</file>
            </source>
            <destination exist="overwrite" type="directory">
                <file>/tmp</file>
            </destination>
        </item>
        </transferSet>
    </managedTransfer>
</request>
```

Create transfer request - transfer to IBM 4690

In this example XML, the file xyz.txt is set to mirrored on close when transferred to the directory c:\adx_test on a IBM 4690 store controller.

```
<?xml version="1.0" encoding="UTF-8"?>
<request version="5.00"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
  <managedTransfer>
    <originator>
      <hostName>example.com.</hostName><userID>bob</userID>
    </originator>
    <sourceAgent agent="AGENT_A" QMgr="qm_a"/>
    <destinationAgent agent="AGENT_B" QMgr="qm_b"/>
    <transferSet>
      <item mode="binary" checksumMethod="MD5">
        <source recursive="false" disposition="leave">
          <file>xyz.txt</file>
        </source>
        <destination type="directory" exist="error">
          <file>c:\adx_test</file>
```

```
            <attributes>
              <attribute>DIST(MIRRORED,CLOSE)</attribute>
            </attributes>
          </destination>
        </item>
      </transferSet>
    </managedTransfer>
</request>
```

For more information about distribution type attributes for WebSphere MQ Managed File Transfer on IBM 4690, see "File distribution attributes" on page 87.

Create transfer request - asynchronous

When a user requests a blocking synchronous request, that is, they wait for the transfer to complete and receive status messages, the message placed on the command queue contains a reply element that specifies the queue that a reply message is sent to. The following example shows the message placed on the command queue used by FTEAGENT:

```
<?xml version="1.0" encoding="UTF-8"?>
<request version="4.00"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
  <managedTransfer>
    <originator>
        <hostName>reportserver.com</hostName>
        <userID>USER1</userID>
    </originator>
    <sourceAgent agent="FTEAGENT"
                 QMgr="QM1"/>
    <destinationAgent agent="AGENT2"
                      QMgr="QM2"/>
    <reply QMGR="QM1">WMQFTE.492D0D5502770020</reply>
    <transferSet>
      <item mode="binary" checksumMethod="MD5">
        <source recursive="false" disposition="leave">
          <file>c:\sourcefiles\source1.doc</file>
        </source>
        <destination type="file" exist="overwrite">
          <file>c:\destinationfiles\dest1.doc</file>
        </destination>
      </item>
    </transferSet>
  </managedTransfer>
</request>
```

The <reply> element is populated with the name of the command queue manager where a temporary dynamic queue has been created to receive reply about the successful (or otherwise) completion of the transfer. The name of the temporary dynamic queue is composed of two parts:
- The prefix as defined by the key **dynamicQueuePrefix** in the `command.properties` configuration file (it is WMQFTE. by default)
- The ID of the queue as generated by IBM WebSphere MQ

Cancel transfer request

```
<?xml version="1.0" encoding="UTF-8"?>
<cancel xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        version="4.00"
        xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
    <originator>
        <hostName>example.com.</hostName>
        <userID>mqm</userID>
    </originator>
    <transfer>414D51205553322E42494E44494E47538B0F404D032C0020</transfer>
    <reply QMGR="QM_JUPITER">WMQFTE.4D400F8B20002007</reply>
</cancel>
```

Related reference:

"File transfer request message format" on page 803
File transfers are initiated by XML messages arriving at an agent command queue, typically as a result of a user issuing a file transfer command or by using the WebSphere MQ Explorer plug-in. The transfer request XML must conform to the FileTransfer.xsd schema and have the <request> element as the root element. The FileTransfer.xsd schema document is located in the *MQ_INSTALLATION_PATH*/mqft/samples/ schema directory. The FileTransfer.xsd schema imports fteutils.xsd, which is in the same directory.

Scheduled transfer message examples:

Examples of the messages that you can put on the agent command queue to request that the agent create or delete a schedule.

Create scheduled transfer

```
<?xml version="1.0" encoding="UTF-8"?>
<request xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        version="4.00"
        xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
    <managedTransfer>
        <originator>
            <hostName>example.com.</hostName>
            <userID>mqm</userID>
        </originator>
        <schedule>
            <submit timebase="admin" timezone="Europe/London">2010-01-01T21:00</submit>
        </schedule>
        <sourceAgent QMgr="US2.BINDINGS" agent="US2.BINDINGS.FILE"/>
        <destinationAgent QMgr="US2.BINDINGS" agent="US2.BINDINGS.FILE"/>
        <transferSet>
            <item checksumMethod="MD5" mode="binary">
                <source disposition="leave" recursive="false">
                    <file>/etc/passwd</file>
                </source>
                <destination exist="overwrite" type="directory">
                    <file>/tmp</file>
                </destination>
            </item>
        </transferSet>
    </managedTransfer>
</request>
```

Delete scheduled transfer

```
<?xml version="1.0" encoding="UTF-8"?>
<request xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        version="4.00"
        xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
    <deleteScheduledTransfer>
        <originator>
            <delete>
                <hostName>example.com.</hostName>
                <userID>mqm</userID>
            </delete>
        </originator>
        <ID>1</ID>
        <reply QMGR="US2.BINDINGS">WMQFTE.4D400F8B20003902</reply>
    </deleteScheduledTransfer>
</request>
```

Related reference:

"File transfer request message format" on page 803
File transfers are initiated by XML messages arriving at an agent command queue, typically as a result of a user issuing a file transfer command or by using the WebSphere MQ Explorer plug-in. The transfer request XML must conform to the `FileTransfer.xsd` schema and have the <request> element as the root element. The `FileTransfer.xsd` schema document is located in the *MQ_INSTALLATION_PATH*/mqft/samples/ schema directory. The `FileTransfer.xsd` schema imports `fteutils.xsd`, which is in the same directory.

Call request message examples:

Examples of the messages that you can put on the agent command queue to request that the agent creates a managed call or creates a transfer that calls programs.

Managed call request example

```
<?xml version="1.0" encoding="UTF-8"?>
<request xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        version="1.00"
        xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
    <managedCall>
        <originator>
            <hostName>example.com.</hostName>
            <userID>mqm</userID>
        </originator>
        <agent agent="DNWE" QMgr="QM1"/>
        <transferSet>
            <call>
                <command name="echo" successRC="0">
                    <argument>call</argument>
                    <argument>test</argument>
                </command>
            </call>
        </transferSet>
        <job>
            <name>managedCallCalls.xml</name>
        </job>
    </managedCall>
</request>
```

Managed transfer request example with calls

```
<?xml version="1.0" encoding="UTF-8"?>
<request xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        version="1.00"
        xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
    <managedTransfer>
        <originator>
            <hostName>example.com.</hostName>
            <userID>mqm</userID>
        </originator>
        <sourceAgent agent="DNWE" QMgr="QM1"/>
        <destinationAgent agent="DNWE" QMgr="QM1"/>
        <transferSet>
            <preSourceCall>
                <command name="echo" successRC="0">
                    <argument>preSourceCall</argument>
                    <argument>test</argument>
                </command>
            </preSourceCall>
            <postSourceCall>
                <command name="echo" successRC="0">
                    <argument>postSourceCall</argument>
                    <argument>test</argument>
                </command>
            </postSourceCall>
```

```
            <preDestinationCall>
                <command name="echo" successRC="0">
                    <argument>preDestinationCall</argument>
                    <argument>test</argument>
                </command>
            </preDestinationCall>
            <postDestinationCall>
                <command name="echo" successRC="0">
                    <argument>postDestinationCall</argument>
                    <argument>test</argument>
                </command>
            </postDestinationCall>
        </transferSet>
        <job>
            <name>managedTransferCalls.xml</name>
        </job>
    </managedTransfer>
</request>
```

Related concepts:

"Specifying programs to run" on page 277
You can run programs on a system where a IBM WebSphere MQ Managed File Transfer agent is running.
As part of a file transfer request, you can specify a program to run either before a transfer starts, or after
it finishes. Additionally, you can start a program that is not part of a file transfer request by submitting a
managed call request.

Related reference:

"File transfer request message format" on page 803
File transfers are initiated by XML messages arriving at an agent command queue, typically as a result of
a user issuing a file transfer command or by using the WebSphere MQ Explorer plug-in. The transfer
request XML must conform to the `FileTransfer.xsd` schema and have the <request> element as the root
element. The `FileTransfer.xsd` schema document is located in the *MQ_INSTALLATION_PATH*/mqft/samples/
schema directory. The `FileTransfer.xsd` schema imports `fteutils.xsd`, which is in the same directory.

Monitor request message formats

Resource monitors are created when a suitable XML message arrives at an agent's command queue,
typically as a result of a user issuing the fteCreateMonitor command or using the WebSphere MQ
Explorer interface.

The monitor XML must conform to the `Monitor.xsd` schema using the <monitor> element as the root
element.

Monitor messages can have one of the following root elements:
* <monitor> - for creating and starting a new resource monitor
* <deleteMonitor> - for stopping and deleting an existing monitor

There is no command message for the fteListMonitors command because the command directly retrieves
matching monitor definitions from the SYSTEM.FTE topic.

Schema

The following schema describes which elements are valid in a monitor request XML message.

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        targetNamespace="http://www.ibm.com/xmlns/wmqfte/7.0.1/MonitorDefinition"
        xmlns="http://www.ibm.com/xmlns/wmqfte/7.0.1/MonitorDefinition">

<xsd:include schemaLocation="FileTransfer.xsd" />

    <xsd:element name="monitor">
        <xsd:complexType>
            <xsd:sequence>
```

```xml
                <xsd:element name="name"           type="monitorNameType"
                             minOccurs="1"         maxOccurs="1" />
                <xsd:element name="description"    type="xsd:string"
                             minOccurs="0"         maxOccurs="1" />
                <xsd:element name="pollInterval"   type="pollIntervalType"
                             minOccurs="1"         maxOccurs="1" default="10" />
                <xsd:element name="batch"          type="batchType"
                             minOccurs="0"         maxOccurs="1" />
                <xsd:element name="agent"          type="agentNameType"
                             minOccurs="1"         maxOccurs="1" />
                <xsd:element name="resources"      type="monitorResourcesType"
                             minOccurs="0"         maxOccurs="1" />
                <xsd:element name="triggerMatch"   type="triggerMatchType"
                             maxOccurs="1"         minOccurs="1" />
                <xsd:element name="reply"          type="replyType"
                             maxOccurs="1"         minOccurs="0" />
                <xsd:element name="tasks"          type="monitorTasksType"
                             maxOccurs="1"         minOccurs="1" />
                <xsd:element name="originator"     type="origRequestType"
                             maxOccurs="1"         minOccurs="1"/>
                <xsd:element name="job"            type="jobType"
                             maxOccurs="1"         minOccurs="0"/>
                <xsd:element name="defaultVariables" type="defaultVariablesType"
                             maxOccurs="1"         minOccurs="0"/>
            </xsd:sequence>
            <xsd:attribute name="version" type="versionType" use="required" />
        </xsd:complexType>
</xsd:element>

<xsd:element name="deleteMonitor">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="name"       type="monitorNameType"
                         minOccurs="1"     maxOccurs="1" />
            <xsd:element name="originator" type="origRequestType"
                         maxOccurs="1"     minOccurs="1"/>
            <xsd:element name="reply"      type="replyType"
                         maxOccurs="1"     minOccurs="0" />
        </xsd:sequence>
        <xsd:attribute name="version" type="versionType" use="required" />
    </xsd:complexType>
</xsd:element>

<xsd:complexType name="transferRequestType">
    <xsd:choice>
        <xsd:element name="managedTransfer" type="managedTransferType" />
        <xsd:element name="managedCall"     type="managedCallType" />
    </xsd:choice>
    <xsd:attribute name="version" type="versionType" />
</xsd:complexType>

<xsd:complexType name="monitorResourcesType">
    <xsd:choice>
        <xsd:sequence>
            <xsd:element name="directory" type="monitoredDirectoryType"
                         minOccurs="1"    maxOccurs="unbounded" />
        </xsd:sequence>
        <xsd:element name="queue" type="monitoredQueueType"/>
    </xsd:choice>
 </xsd:complexType>

<xsd:complexType name="monitoredDirectoryType">
    <xsd:simpleContent>
        <xsd:extension base="xsd:string">
            <xsd:attribute name="recursionLevel" type="xsd:nonNegativeInteger" />
            <xsd:attribute name="id" type="resourceIdAttrType" />
        </xsd:extension>
```

```
        </xsd:simpleContent>
    </xsd:complexType>


    <xsd:complexType name="monitoredQueueType">
        <xsd:simpleContent>
            <xsd:extension base="xsd:string">
                <xsd:attribute name="id" type="resourceIdAttrType" />
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>


    <xsd:complexType name="triggerMatchType">
        <xsd:sequence>
            <xsd:element name="conditions" type="conditionsType"
                        minOccurs="1"    maxOccurs="1" />
        </xsd:sequence>
    </xsd:complexType>


    <xsd:complexType name="conditionsType">
        <xsd:choice minOccurs="1">
            <xsd:element name="allOf"     type="listPredicateType"
                        minOccurs="1"    maxOccurs="1" />
            <xsd:element name="anyOf"     type="listPredicateType"
                        minOccurs="1"    maxOccurs="1" />
            <xsd:element name="condition" type="conditionType"
                        minOccurs="1"    maxOccurs="1" />
        </xsd:choice>
    </xsd:complexType>


    <xsd:complexType name="listPredicateType">
        <xsd:choice>
            <xsd:element name="condition" type="conditionType"
                        minOccurs="1"    maxOccurs="unbounded" />
        </xsd:choice>
    </xsd:complexType>


    <xsd:complexType name="conditionType">
        <xsd:sequence>
            <xsd:element name="name"     type="conditionNameType"
                        minOccurs="0"   maxOccurs="1" />
            <xsd:element name="resource" type="resourceIdType"
                        minOccurs="0"   maxOccurs="1" />
            <xsd:choice minOccurs="1">
                <xsd:element name="fileMatch"       type="fileMatchConditionType"
                            minOccurs="1"        maxOccurs="1" />
                <xsd:element name="fileNoMatch"     type="fileNoMatchConditionType"
                            minOccurs="1"        maxOccurs="1" />
                <xsd:element name="fileSize"        type="fileSizeConditionType"
                            minOccurs="1"        maxOccurs="1" />
                <xsd:element name="queueNotEmpty"   type="queueNotEmptyConditionType"
                            minOccurs="1"        maxOccurs="1" />
                <xsd:element name="completeGroups"  type="completeGroupsConditionType"
                            minOccurs="1"        maxOccurs="1" />
                <xsd:element name="fileSizeSame"    type="fileSizeSameType"
                            minOccurs="1"        maxOccurs="1"/>
            </xsd:choice>
        </xsd:sequence>
    </xsd:complexType>


    <xsd:complexType name="fileMatchConditionType">
        <xsd:sequence>
            <xsd:element name="pattern" type="conditionPatternType"
                        minOccurs="0"  default="*.*" />
            <xsd:element name="exclude" type="conditionPatternType"
                        minOccurs="0"  maxOccurs="1"/>
        </xsd:sequence>
    </xsd:complexType>
```

```
<xsd:complexType name="fileNoMatchConditionType">
    <xsd:sequence>
        <xsd:element name="pattern" type="conditionPatternType"
                     minOccurs="0"  default="*.*" />
        <xsd:element name="exclude" type="conditionPatternType"
                     minOccurs="0"  maxOccurs="1"/>
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="fileSizeConditionType">
    <xsd:sequence>
        <xsd:element name="compare" type="sizeCompareType"
                     minOccurs="1"  default="0" />
        <xsd:element name="pattern" type="conditionPatternType"
                     minOccurs="0"  default="*.*" />
        <xsd:element name="exclude" type="conditionPatternType"
                     minOccurs="0"  maxOccurs="1"/>
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="sizeCompareType">
    <xsd:simpleContent>
        <xsd:extension base="xsd:int">
            <xsd:attribute name="operator" type="sizeOperatorType"  use="required" />
            <xsd:attribute name="units"    type="fileSizeUnitsType" use="required" />
        </xsd:extension>
    </xsd:simpleContent>
</xsd:complexType>

<xsd:simpleType name="sizeOperatorType">
    <xsd:restriction base="xsd:string">
        <xsd:enumeration value=">=" />
    </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="fileSizeUnitsType">
    <xsd:restriction base="xsd:string">
        <xsd:pattern value="[bB]|[kK][bB]|[mM][bB]|[gG][bB]" />
    </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="conditionPatternType">
    <xsd:simpleContent>
        <xsd:extension base="xsd:string">
            <xsd:attribute name="type"     type="patternTypeAttributeType"
                           use="optional" default="wildcard"/>
        </xsd:extension>
    </xsd:simpleContent>
</xsd:complexType>

<xsd:simpleType name="patternTypeAttributeType">
    <xsd:restriction base="xsd:token">
        <xsd:enumeration value="regex" />
        <xsd:enumeration value="wildcard" />
    </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="conditionNameType">
    <xsd:simpleContent>
        <xsd:extension base="xsd:string" />
    </xsd:simpleContent>
</xsd:complexType>

<xsd:complexType name="queueNotEmptyConditionType"/>

<xsd:complexType name="completeGroupsConditionType"/>
```

```
<xsd:complexType name="fileSizeSameType">
    <xsd:sequence>
        <xsd:element name="pattern" type="conditionPatternType"
                    minOccurs="1"  maxOccurs="1"/>
        <xsd:element name="exclude" type="conditionPatternType"
                    minOccurs="0"  maxOccurs="1"/>
    </xsd:sequence>
    <xsd:attribute name="polls" type="positiveIntegerType" use="required" />
</xsd:complexType>

<xsd:complexType name="pollIntervalType">
    <xsd:simpleContent>
        <xsd:extension base="xsd:int">
            <xsd:attribute name="units"    type="timeUnitsType"
                            use="optional"  default="minutes" />
        </xsd:extension>
    </xsd:simpleContent>
</xsd:complexType>

<xsd:complexType name="batchType">
    <xsd:attribute name="maxSize" type="positiveIntegerType" use="required"/>
</xsd:complexType>

<xsd:simpleType name="timeUnitsType">
    <xsd:restriction base="xsd:token">
        <xsd:enumeration value="seconds" />
        <xsd:enumeration value="minutes" />
        <xsd:enumeration value="hours" />
        <xsd:enumeration value="days" />
        <xsd:enumeration value="weeks" />
        <xsd:enumeration value="months" />
        <xsd:enumeration value="years" />
    </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="monitorTasksType">
    <xsd:sequence>
        <xsd:element name="task"    type="monitorTaskType"
                    minOccurs="1" maxOccurs="1" />
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="monitorTaskType">
    <xsd:sequence>
        <xsd:element name="name"        type="monitorTaskNameType"
                    minOccurs="1"      maxOccurs="1" />
        <xsd:element name="description" type="xsd:string"
                    minOccurs="0"      maxOccurs="1" />
        <xsd:element name="transfer"    type="transferTaskType"
                    minOccurs="0"      maxOccurs="1" />
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="transferTaskType">
    <xsd:sequence>
        <xsd:element name="request"  type="transferRequestType"
                    minOccurs="1"   maxOccurs="1" />
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="resourceIdType">
    <xsd:attribute name="id" type="xsd:string" use="optional" />
</xsd:complexType>

<xsd:simpleType name="resourceIdAttrType">
    <xsd:restriction base="xsd:string"></xsd:restriction>
```

```
        </xsd:simpleType>

        <xsd:simpleType name="monitorNameType">
            <xsd:restriction base="xsd:string">
                <xsd:pattern value="[^%\*]+" />
            </xsd:restriction>
        </xsd:simpleType>

        <xsd:simpleType name="agentNameType">
            <xsd:restriction base="xsd:string">
                <xsd:pattern value="[.%_0-9A-Z]*" />
            </xsd:restriction>
        </xsd:simpleType>

        <xsd:simpleType name="monitorTaskNameType">
            <xsd:restriction base="xsd:string">
                <xsd:pattern value=".*" />
            </xsd:restriction>
        </xsd:simpleType>

        <xsd:complexType name="defaultVariablesType">
            <xsd:sequence>
                <xsd:element name="variable"        type="variableType"
                        maxOccurs="unbounded" minOccurs="1" />
            </xsd:sequence>
        </xsd:complexType>

        <xsd:complexType name="variableType">
            <xsd:simpleContent>
                <xsd:extension base="xsd:string">
                    <xsd:attribute name="key" type="xsd:string" use="required" />
                </xsd:extension>
            </xsd:simpleContent>
        </xsd:complexType>

</xsd:schema>
```

Understanding the create monitor message

The elements and attributes used in create monitor messages are described:

Element descriptions

`<monitor>`

Group element containing all the elements required to cancel a file transfer in progress.

Attribute	Description
version	Specifies the version of this element as supplied by WebSphere MQ Managed File Transfer.

`<name>`

The name of the monitor, unique within the monitor's agent.

`<description>`

Description of the monitor (not currently used).

`<pollInterval>`

The time interval between each check of the resource against the trigger condition.

Attribute	Description
units	Specifies the time units for the poll interval. Valid values are: • seconds • minutes • hours • days • weeks • months • years

`<agent>`
    Name of the agent the monitor is associated with.

`<resources>`
    Group element that contains the elements specifying the resources to monitor.

`<directory>`
    Fully qualified path specifying the directory on the monitor's agent machine to monitor.

Attribute	Description
recursionLevel	The number of subdirectories to monitor in addition to the specified directory.
id	Unique identifier for the resource.

`<queue>`
    Queue name specifying the queue to monitor on the monitoring agent's queue manager.

`<triggerMatch>`
    Group element that contains the elements specifying the trigger conditions to compare with the monitored resource.

`<conditions>`
    Group element that contains the elements specifying the type of condition to compare with the monitored resource.

`<allOf>`
    Predicate that specifies that all contained conditions must be satisfied.

`<anyOf>`
    Predicate that specifies that any contained conditions must be satisfied.

`<condition>`
    Defines a comparison condition that will contribute to the overall monitor trigger condition.

`<name>`
    Name of the condition.

`<resource>`
    Identifies the resource definition to compare the condition against.

Attribute	Description
id	Unique identifier for the resource.

If the resource that is being monitored is a directory, one of the following three elements must be specified in the condition:

- fileMatch
- fileNoMatch
- fileSize

If the resource that is being monitored is a queue, one of the following two elements must be specified in the condition:
- queueNotEmpty
- completeGroups

\<fileMatch\>
Group element for a file name match condition.

\<pattern\>
Specifies a file name match pattern. Files on the resource must match the pattern in order to satisfy the condition. The default pattern is * (any file will match).

\<fileNoMatch\>
Group element for an inverse file name match condition.

\<pattern\>
Specifies an inverse file name match pattern. If no files on the monitored resource match, the condition is satisfied. The default pattern is * (the absence of any file will match).

\<fileSize\>
Group element for a file size comparison.

\<compare\>
Specifies a file size comparison. The value must be a non-negative integer.

Attribute	Description
operator	Comparison operator to use. Only >=' is supported.
units	Specifies file size units, which can be one of: • B - bytes • KB - kilobytes • MB - megabytes • GB - gigabytes The units value is case insensitive, so mb' works as well as MB'.

\<pattern\>
File name pattern to match. Default is * (any file will match).

\<queueNotEmpty\>
This can only be specified if the resource is a queue. Specifies that there must be a message on the queue for the monitor to be triggered.

\<completeGroups\>
This can only be specified if the resource is a queue. Specifies that there must be a complete group of messages present on the queue for the monitor to be triggered. A single transfer task is executed for each complete group on the queue.

\<reply\>
Optional element that is used to specify reply queue for asynchronous requests.

Attribute	Description
QMGR	Queue manager name.

`<tasks>`
　　Group element to contain elements which specify the tasks to invoke when the monitor trigger conditions are satisfied.

`<task>`
　　Group element which defines an individual task that the monitor will invoke when the trigger conditions are satisfied. Currently only one task can be specified.

`<name>`
　　Name of the task. Accepts any alphanumeric characters.

`<description>`
　　Description of the task. Any text value is allowed.

`<transfer>`
　　Group element that defines a transfer task.

`<request>`
　　Group element that defines the type of task. This must contain one of the following elements which are inherited from the `FileTransfer.xsd` schema definition:

- managedTransfer
- managedCall

Attribute	Description
version	Version of the request as provided by WebSphere MQ Managed File Transfer. This is in the form n.mm where n is the major release version and mm is the minor version. For example 1.00.

`<originator>`
　　Group element that contains the elements specifying the originator of the request.

`<hostName>`
　　The host name of the system where the source file is located.

`<userID>`
　　The user ID that originated the file transfer.

`<mqmdUserID>`
　　Optional. The WebSphere MQ user ID that was supplied in the message descriptor (MQMD).

`<job>`
　　Group element containing job information.

`<jobName>`
　　Specifies logical job identifier.

`<defaultVariables>`
　　Group element containing one or more variable elements. These variables are used in variable substitution when monitoring a queue. For more information about variable substitution, see "Customizing tasks with variable substitution" on page 225.

`<variable>`
　　Element containing the value associated with the key given by the key attribute.

Attribute	Description
key	The name of the default variable.

Understanding the delete monitor message

The elements and attributes used in delete monitor messages are described:

Element descriptions

<deleteMonitor>
> Group element containing all the elements required to stop and delete a monitor.

Attribute	Description
version	Specifies the version of this element as supplied by WebSphere MQ Managed File Transfer.

<name>
> Name of monitor to delete.

<originator>
> Group element that contains the elements specifying the originator of the request.

<hostName>
> The host name of the system where the source file is located.

<userID>
> The user ID that originated the file transfer.

<mqmdUserID>
> Optional. The WebSphere MQ user ID that was supplied in the message descriptor (MQMD).

<reply>
> Specifies the name of the temporary reply queue generated for the request. The name of the queue is as defined by the key `dynamicQueuePrefix` in the `command.properties` configuration file. If this is not specified, the queue name has a default value of WMQFTE.

Attribute	Description
QMGR	The name of the command queue manager on which the temporary dynamic queue is generated to receive replies.

Examples

Examples of XML messages that conform to this schema are provided for each of the following monitor requests:
- Create a monitor
- Delete a monitor

Related concepts:

"Resource monitoring" on page 215
You can monitor WebSphere MQ Managed File Transfer resources; for example, a queue or a directory. When a condition on this resource is satisfied, the resource monitor starts a task, such as a file transfer. You can create a resource monitor by using the **fteCreateMonitor** command or the Monitors view in the WebSphere MQ Managed File Transfer plug-in for WebSphere MQ Explorer.

Monitor request message examples:

Examples of the messages that you can put on the agent command queue to request that the agent create or delete a monitor.

Create monitor request

```xml
<?xml version="1.0" encoding="UTF-8"?>
<monitor:monitor xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                 xmlns:monitor="http://www.ibm.com/xmlns/wmqfte/7.0.1/MonitorDefinition"
                 version="4.00"
                 xsi:schemaLocation="http://www.ibm.com/xmlns/wmqfte/7.0.1/MonitorDefinition ./Monitor.xsd">
    <name>EXAMPLEMONITOR</name>
    <pollInterval>1</pollInterval>
    <agent>US2.BINDINGS.FILE</agent>
    <resources>
        <directory recursionLevel="0">/srv/nfs/incoming</directory>
    </resources>
    <triggerMatch>
        <conditions>
            <allOf>
                <condition>
                    <fileMatch>
                        <pattern>*.completed</pattern>
                    </fileMatch>
                </condition>
            </allOf>
        </conditions>
    </triggerMatch>
    <reply QMGR="US2.BINDINGS">WMQFTE.4D400F8B20003702</reply>
    <tasks>
        <task>
            <name/>
            <transfer>
                <request xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                         version="4.00"
                         xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
                    <managedTransfer>
                        <originator>
                            <hostName>example.com.</hostName>
                            <userID>mqm</userID>
                        </originator>
                        <sourceAgent QMgr="US2.BINDINGS" agent="US2.BINDINGS.FILE"/>
                        <destinationAgent QMgr="US2.BINDINGS" agent="US2.BINDINGS.FILE"/>
                        <transferSet>
                            <item checksumMethod="MD5" mode="binary">
                                <source disposition="leave" recursive="false">
                                    <file>/srv/nfs/incoming/*.txt</file>
                                </source>
                                <destination exist="error" type="directory">
                                    <file>/srv/backup</file>
                                </destination>
                            </item>
                        </transferSet>
                    </managedTransfer>
                </request>
            </transfer>
        </task>
    </tasks>
    <originator>
        <hostName>example.com.</hostName>
        <userID>mqm</userID>
    </originator>
</monitor:monitor>
```

Delete monitor request

```xml
<?xml version="1.0" encoding="UTF-8"?>
<monitor:deleteMonitor xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                       xmlns:monitor="http://www.ibm.com/xmlns/wmqfte/7.0.1/MonitorDefinition"
                       version="4.00"
                       xsi:schemaLocation="http://www.ibm.com/xmlns/wmqfte/7.0.1/MonitorDefinition ./Monitor.xsd">
    <name>EXAMPLEMONITOR</name>
    <originator>
        <hostName>example.com.</hostName>
        <userID>mqm</userID>
    </originator>
    <reply QMGR="QM_JUPITER">WMQFTE.4D400F8B20003705</reply>
</monitor:deleteMonitor>
```

Resource monitors are created when a suitable XML message arrives at an agent's command queue, typically as a result of a user issuing the fteCreateMonitor command or using the WebSphere MQ Explorer interface.

Ping agent request message format

You can ping an agent by issuing an **ftePingAgent** command or by putting an XML message on the agent command queue. The ping agent request XML must conform to the PingAgent.xsd schema. After you have installed WebSphere MQ Managed File Transfer, you can find the PingAgent.xsd schema file in the following directory: *MQ_INSTALLATION_PATH*/mqft/samples/schema. The PingAgent.xsd schema imports fteutils.xsd, which is in the same directory.

When the agent receives a ping agent request message on its command queue, if the agent is active, it returns an XML response message to the command or application that put the ping agent request message on the command queue. The response message from the agent is in the format defined by Reply.xsd. For more information about this format, see "Reply message format" on page 832.

Schema

The following schema describes which elements are valid in an ping agent request XML message.

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
           xmlns="http://www.ibm.com/xmlns/wmqfte/7.0.1/PingAgent"
           targetNamespace="http://www.ibm.com/xmlns/wmqfte/7.0.1/PingAgent">

   <xsd:include schemaLocation="fteutils.xsd"/>

   <xsd:element name="pingAgent">
       <xsd:complexType>
           <xsd:sequence>
               <xsd:element name="originator" type="origRequestType" maxOccurs="1" minOccurs="1"/>
               <xsd:element name="agent"      type="agentType"       maxOccurs="1" minOccurs="1"/>
               <xsd:element name="reply"      type="replyType"       maxOccurs="1" minOccurs="0" />
           </xsd:sequence>
           <xsd:attribute name="version" type="versionType" use="required" />
       </xsd:complexType>
   </xsd:element>

</xsd:schema>
```

Understanding the ping agent request message

The elements and attributes used in the ping agent request messages are described in the following list:

<pingAgent>
  Group element containing all the elements required to specify a ping agent request.

<originator>
  Group element containing all the elements required to specify the originator of the ping request.

  <hostName>
    The host name of the machine where the request originated.

  <userID>
    The user name of the originator of the request.

  <mqmdUserID>
    The MQMD user name of the originator of the request.

<agent>
  The agent to ping.

Attribute	Description
agent	Required. The name of the agent.
QMgr	Optional. The queue manager that the agent connects to.

<reply>

The name of the queue for the agent to send the reply message to.

Attribute	Description
QMGR	Required. The name of the queue manager where the reply queue is located.

Example

This example shows a ping agent message sent to the agent AGENT_JUPITER. If AGENT_JUPITER is active and able to process agent requests, it sends a response message to the queue WMQFTE.4D400F8B20003708 on QM_JUPITER.

```
<?xml version="1.0" encoding="UTF-8"?>
<ping:pingAgent xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                xmlns:ping="http://www.ibm.com/xmlns/wmqfte/7.0.1/PingAgent"
                version="4.00">
    <originator>
        <hostName>example.com.</hostName>
        <userID>mqm</userID>
    </originator>
    <agent agent="AGENT_JUPITER" QMgr="QM_JUPITER"/>
    <reply QMGR="QM_JUPITER">WMQFTE.4D400F8B20003708</reply>
</ping:pingAgent>
```

Reply message format

When an agent receives an XML message on its agent command queue, if a response is required, the agent will send an XML reply message to the reply queue defined in the original message. The reply XML conforms to the Reply.xsd schema. The Reply.xsd schema document is located in the *MQ_INSTALLATION_PATH*/mqft/samples/schema directory. The Reply.xsd schema imports fteutils.xsd, which is in the same directory.

Schema

The following schema describes which elements are valid in a reply XML message.

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

<xsd:include schemaLocation="TransferLog.xsd"/>

    <xsd:element name="reply">
     <xsd:complexType>
      <xsd:sequence>
         <xsd:element name="transferSet" type="transferSetType" minOccurs="0" maxOccurs="1" />
       <xsd:element name="status" type="statusType" minOccurs="1" maxOccurs="1" />
      </xsd:sequence>
         <xsd:attribute name="version"   type="versionType"      use="required"/>
         <xsd:attribute name="ID"        type="IDType"           use="required"/>

     </xsd:complexType>
    </xsd:element>

</xsd:schema>
```

Understanding the reply message

The elements and attributes used in the reply messages are described in the following list:

`<reply>`
>   Element containing the elements that specify the reply information.

Attribute	Description
ID	The ID of the reply.
version	The version of the reply message format.

`<status>`
>   The status of the action that the agent was requested to perform.

Attribute	Description
resultCode	The result code returned from the action that the agent performed.

`<supplement>`
>   Additional response information about the action that the agent was requested to perform.

Example

In the following section is an example reply message:

```
<reply version="1.00"              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                                   xsi:noNamespaceSchemaLocation="Reply.xsd"
                                   ID="010202030000000000000000000000000000000000000000">
    <status resultCode="65">
        <supplement>Additional reply information</supplement>
    </status>
</reply>
```

Message formats for security

This topic describes the messages published to the coordination queue manager relevant to security.

Not authorized log message

If user authority checking is enabled the agent can publish not authorized messages to the coordination queue manager. "User authorities on WebSphere MQ Managed File Transfer actions" on page 409 describes how to enable user authority checking.

Every time a user submits a request to perform a restricted action to the agent, either by using a WebSphere MQ Managed File Transfern command or by using the WebSphere MQ Explorer plug-in, the agent checks that the user has the authority to perform the action. If the user fails that authority check, a not authorized log message is published to the coordination queue manager on its SYSTEM.FTE/Log/ *agent_name*/NotAuthorized topic.

This message conforms to the `TransferLog.xsd` XML schema. For example:

```
<?xml version="1.0" encoding="UTF-8"?>
<notAuthorized version="3.00"
            ID="414d5120716d312020202020202020204da5924a2010ce03"
            agentRole="sourceAgent"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:noNamespaceSchemaLocation="TransferLog.xsd"
            xmlns="">
 <action time="2009-08-28T12:31:15.781Z">not_authorized</action>
  <originator>
  <mqmdUserID>test1</mqmdUserID>
  </originator>
```

```
  <authority>administration</authority>
  <status resultCode="53">
  <supplement>BFGCH0083E: The user (test1) does not have the authority (ADMINISTRATION) required to shut down agent 'AGENT'.</supplement>
  <supplement>
&lt;?xml version=&quot;1.0&quot; encoding=&quot;UTF-8&quot;?&gt;
&lt;internal:request version=&quot3.00&quot; xmlns:xsi=&quot;http://www.w3.org/2001/XMLSchema-instance&quot;
 xmlns:internal=&quot;http://wmqfte.ibm.com/internal&quot;&gt;
&lt;internal:shutdown agent=&quot;SYSTEM.FTE.COMMAND.AGENT&quot; hostname= &quot;qm1&quot; mode=&quot;controlled&quot;/&gt;
&lt;reply QMGR=&quot;qm1&quot;&gt;WMQFTE.4A92A54D02CE1020&lt;/reply&gt;
&lt;/internal:request&gt;
    </supplement>
 </status>
</notAuthorized>
```

This message is a log of the following information:

- Who originated the request
- The level of WebSphere MQ Managed File Transfer access authority required to perform the request
- The status of the request
- The request specification

Understanding the not authorized log message

The elements and attributes used in the not authorized message are described:

`<notAuthorized>`
   Group element that describes a single failed user authorization check.

Attribute	Description
version	Specifies the version of this element as detailed by WebSphere MQ Managed File Transfer.
ID	The unique identifier for the request that was not authorized.

`<originator>`
   Group element that contains the elements specifying the originator of the request.

`<authority>`
   Specifies the level of WebSphere MQ Managed File Transfer access authority that the user required to perform the requested action.

`<mqmdUserID>`
   The WebSphere MQ user ID that was supplied in the message descriptor (MQMD)

`<action>`
   Specifies the authorization status of the request matching the ID attribute of <notAuthorized> element.

Attribute	Description
time	Specifies the date and time the log entry was published (in date time format).

`<status>`
   The result code and supplement messages.

MQMFT credentials file format

The `MQMFTCredentials.xml` file is new for WebSphere MQ Managed File Transfer V7.5. It contains sensitive information previously held in separate properties files. The security of credentials files is the responsibility of the user.

The `MQMFTCredentials.xml` file must conform to the `MQMFTCredentials.xsd` schema. The `MQMFTCredentials.xml` schema document is located in the *MQ_INSTALLATION_PATH*/mqft/samples/schema directory of the WebSphere MQ Managed File Transfer installation.

Schema

The following schema describes which elements are valid in the `MQMFTCredentials.xml` file.

```
<?xml version="1.0" encoding="UTF-8"?>
  <!--
    @start_non_restricted_prolog@
    Version: %Z% %I% %W% %E% %U% [%H% %T%]

    Licensed Materials - Property of IBM

    5724-H72

    Copyright IBM Corp. 2012 All Rights Reserved.

    US Government Users Restricted Rights - Use, duplication or
    disclosure restricted by GSA ADP Schedule Contract with
   IBM Corp.
    @end_non_restricted_prolog@
  -->

  <!--
    This schema defines the format of an MQMFTCredentials file. Files of this type
    store credential information for agent and logger processes. They can contain
    user names and passwords either in clear text or which have been obfuscated
    using the fteObfuscate command.
  -->

  <!-- Example MQMFTCredentials.xml file:

  <?xml version="1.0" encoding="UTF-8"?>
    <tns:mqmftCredentials xmlns:tns="http://wmqfte.ibm.com/MQMFTCredentials"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://wmqfte.ibm.com/MQMFTCredentials MQMFTCredentials.xsd">

      <tns:logger name="LOG1"   user="user1"          password="passw0rd"/>
      <tns:logger name="ORACLE" userCipher="kj2h3dfkgf" passwordCipher="la3n67eaer"/>
      <tns:file   path="home/emma/trust.jks"          password="passw0rd"/>
      <tns:file   path="/var/tmp/keystore.jks"        passwordCipher="e71vKCg2pf"/>

      <tns:qmgr   name="QM_COORD" user="tim" mqUserId="user1"  mqPassword="passw0rd"/>
      <tns:qmgr   name="QM_COORD" user="tom" mqUserId="user1"  mqPasswordCipher="e71vKCg2pf"/>>

    </tns:mqmftCredentials>
  -->

<schema targetNamespace="http://wmqfte.ibm.com/MQMFTCredentials"
  elementFormDefault="qualified"
  xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:tns="http://wmqfte.ibm.com/MQMFTCredentials">

  <element name="mqmftCredentials" type="tns:mqmftCredentialsType"/>

  <complexType name="mqmftCredentialsType">
    <sequence>
```

```
        <choice minOccurs="0" maxOccurs="unbounded">
          <element name="logger" type="tns:loggerType"/>
          <element name="file"   type="tns:fileType"/>

      </choice>
    </sequence>
  </complexType>

  <complexType name="loggerType">
    <attribute name="name"            type="string" use="required"/>
    <attribute name="user"            type="string" use="optional"/>
    <attribute name="userCipher"      type="string" use="optional"/>
    <attribute name="password"        type="string" use="optional"/>
    <attribute name="passwordCipher"  type="string" use="optional"/>
  </complexType>

  <complexType name="fileType">
    <attribute name="path"            type="string" use="required"/>
    <attribute name="password"        type="string" use="optional"/>
    <attribute name="passwordCipher"  type="string" use="optional"/>
  </complexType>

  <!-- Example XML:

  <tns:qmgr   name="QM_COORD" user="tim" mqUserId="user1"  mqPassword="passw0rd"/>
  <tns:qmgr   name="QM_COORD" user="tom" mqUserIdCipher="xh5U7812x"  mqPasswordCipher="e71vKCg2pf"/>
  <tns:qmgr   name="QM_COORD" mqUserId="defaultUser"  mqPassword="passw0rd"/>
-->

  <complexType name="mqUserPassType">
    <attribute name="name"            type="string" use="required"/>
    <attribute name="user"            type="string" use="optional"/>
    <attribute name="mqUserId"        type="string" use="optional"/>
    <attribute name="mqUserIdCipher"  type="string" use="optional"/>
    <attribute name="mqPassword"      type="string" use="optional"/>
    <attribute name="mqPasswordCipher" type="string" use="optional"/>
  </complexType>
</schema>
```

Understanding the `MQMFTCredentials.xml` file

The elements and attributes used in the `MQMFTCredentials.xml` file are described in the following list.

`<mqmftCredentials>`
The root element of the XML document.

`<file>`
The file in the transfer.

Attribute	Description
path	Path to the key or trust store file being accessed.
password	Password to access the file.

`<logger>`
The logger responsible for logging activity.

Attribute	Description
name	The name of the logger.
user	The user name the logger will use to connect to its database.

Attribute	Description
password	The password the logger will use to connect to its database.

Note: The MQMFTCredentials.xml file can contain sensitive information, so when it is created ensure that the file permissions are reviewed. When using a sandbox, set to it be excluded. For more information on sandboxes, see "Working with agent sandboxes" on page 104.

Related reference:

"**fteObfuscate** (encrypt sensitive data)" on page 511

The **fteObfuscate** command encrypts sensitive data in credentials files. This stops the contents of credentials files being read by someone who gains access to the file.

Protocol bridge credentials file format

The ProtocolBridgeCredentials.xml file in the agent configuration directory defines the user names and credential information that the protocol bridge agent uses to authorize itself with the protocol server.

The ProtocolBridgeCredentials.xml file must conform to the ProtocolBridgeCredentials.xsd schema. The ProtocolBridgeCredentials.xsd schema document is located in the *MQ_INSTALLATION_PATH*/mqft/samples/schema directory of the MQMFT installation. Users are responsible for manually creating the ProtocolBridgeCredentials.xml file, it is no longer created by the **fteCreateBridgeAgent** command. Sample files are available in the *MQ_INSTALLATION_PATH*/mqft/samples directory of the MQMFT installation.

V7.5 introduces a new <agent> element that contains the <server> or <serverHost> element for the named agent.

The ProtocolBridgeCredentials.xml file is periodically reloaded by the agent and any valid changes to the file will affect the behavior of the agent. The default reload interval is 30 seconds. This interval can be changed by specifying the agent property xmlConfigReloadInterval in the agent.properties file.

Schema - V7.5

The following schema describes which elements are valid in the ProtocolBridgeCredentials.xml file for V7.5

```
<schema targetNamespace="http://wmqfte.ibm.com/ProtocolBridgeCredentials" elementFormDefault="qualified"
          xmlns="http://www.w3.org/2001/XMLSchema" xmlns:tns="http://wmqfte.ibm.com/ProtocolBridgeCredentials">

 <!--
     <?xml version="1.0" encoding="UTF-8"?>
       <tns:credentials xmlns:tns="http://wmqfte.ibm.com/ProtocolBridgeCredentials"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:schemaLocation="http://wmqfte.ibm.com/ProtocolBridgeCredentials
     ProtocolBridgeCredentials.xsd ">
         <tns:agent name="agent1">
           <tns:serverHost name="myserver">
             <tns:user name="fred" serverPassword="pwd" serverUserId="bill"/>
             <tns:user name="jane" serverUserId="june" hostKey="1F:2e:f3">
               <tns:privateKey associationName="test" keyPassword="pwd2">
                 .... private key ...
               </tns:privateKey>
             </tns:user>
           </tns:serverHost>
         </tns:agent>

         <tns:agent name="agent2">
           <tns:server name="server*" pattern="wildcard">
             <tns:user name="fred" serverPassword="pwd" serverUserId="bill"/>
             <tns:user name="jane" serverUserId="june" hostKey="1F:2e:f3">
               <tns:privateKey associationName="test" keyPassword="pwd2">
                 .... private key ...
               </tns:privateKey>
             </tns:user>
```

```
            </tns:server>
        </tns:agent>

        <tns:agent name="agent3">
          <tns:serverHost name="ftpsServer"
                          keyStorePassword="keypass"
                          trustStorePassword="trustpass">
            <tns:user name="fred" serverPassword="pwd" serverUserId="bill"/>
          </tns:serverHost>
        </tns:agent>

    </tns:credentials>
-->

  <element name="credentials" type="tns:credentialsType"/>

  <complexType name="credentialsType">
   <sequence>
    <element name="agent" type="tns:agentType" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
  </complexType>

  <complexType name="agentType">
   <choice minOccurs="0" maxOccurs="1">
 <element name="serverHost" type="tns:serverHostType" minOccurs="0" maxOccurs="unbounded"/>
 <element name="server" type="tns:serverType" minOccurs="0" maxOccurs="unbounded"/>
 </choice>
 <attribute name="name" type="string" use="required"/>
  </complexType>

  <complexType name="serverHostType">
   <sequence>
    <element ref="tns:user" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="name" type="string" use="required"/>
   <attribute name="keyStorePassword" type="string" use="optional"/>
   <attribute name="keyStorePasswordCipher" type="string" use="optional"/>
   <attribute name="trustStorePassword" type="string" use="optional"/>
   <attribute name="trustStorePasswordCipher" type="string" use="optional"/>
  </complexType>

  <complexType name="serverType">
   <sequence>
    <element ref="tns:user" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="name" type="string" use="required"/>
   <attribute name="pattern" type="tns:patternType" use="optional" />
   <attribute name="keyStorePassword" type="string" use="optional"/>
   <attribute name="keyStorePasswordCipher" type="string" use="optional"/>
   <attribute name="trustStorePassword" type="string" use="optional"/>
   <attribute name="trustStorePasswordCipher" type="string" use="optional"/>
  </complexType>

  <element name="user" type="tns:userType"/>

  <complexType name="userType">
   <sequence>
    <element ref="tns:privateKey" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="name" type="string" use="required"/>
   <attribute name="serverUserId" type="string" use="optional"/>
   <attribute name="serverUserIdCipher" type="string" use="optional"/>
   <attribute name="serverPassword" type="string" use="optional"/>
   <attribute name="serverPasswordCipher" type="string" use="optional"/>
   <attribute name="hostKey" use="optional">
       <simpleType>
     <restriction base="string">
      <pattern
      value="([a-fA-F0-9]){2}(:([a-fA-F0-9]){2})*">
      </pattern>
     </restriction>
    </simpleType>
   </attribute>
  </complexType>

  <element name="privateKey" type="tns:privateKeyType"/>

  <complexType name="privateKeyType">
   <simpleContent>
```

```
        <extension base="string">
     <attribute name="keyPassword" type="string" use="optional"/>
     <attribute name="keyPasswordCipher" type="string" use="optional"/>
     <attribute name="associationName" type="string" use="required"/>
    </extension>
   </simpleContent>
  </complexType>

   <!--
 Determines the type of pattern matching to use.
 -->
 <simpleType name="patternType">
  <restriction base="string">
   <enumeration value="regex" />
   <enumeration value="wildcard" />
  </restriction>
 </simpleType>
</schema>
```

Schema - V7.0.4

The schema for the V7.0.4.1 and later, and the schema for the V7.0.4 and earlier, are available here: Protocol bridge credentials file format

Understanding the `ProtocolBridgeCredentials.xml` file

The elements and attributes used in the `ProtocolBridgeCredentials.xml` file are described in the following list.

`<credentials>`

Group element containing elements that describe the credentials used by a protocol bridge agent to connect to a protocol server.

`<agent>`

Element containing a <server> or <serverHost> definition for a named agent.

`<server>`

The protocol server that the protocol bridge connects to.

The <server> element is not supported for V7.0.4 or earlier.

Attribute	Description
name	The name of the protocol server.
pattern	If you have used wildcards or regular expressions to specify the pattern of a protocol server name, use either `wildcard` or `regex`.
trustStorePassword or trustStorePasswordCipher	Required when the **<server>** element refers to an FTPS server. The password used to access the truststore. If the **fteObfuscate** command has been used then the cipher version of the attribute must be used.
keyStorePassword or keyStorePasswordCipher	Optional. The password used to access the keystore. If the **fteObfuscate** command has been used then the cipher version of the attribute must be used.

`<serverHost>`

The host name of the protocol server that the protocol bridge connects to.

The `ProtocolBridgeCredentials.xml` file can either contain <serverHost> elements or <server> elements but you cannot use a mixture of the two different types. When you use <serverHost>, the name is matched against the protocol server's host name. When you use <server>, the name is matched against the protocol server's name (as defined in the `ProtocolBridgeProperties.xml` file).

Attribute	Description
name	The host name or IP address of the protocol server.
trustStorePassword or trustStorePasswordCipher	Required when the **<serverHost>** element refers to an FTPS server. The password used to access the truststore. If the **fteObfuscate** command has been used then the cipher version of the attribute must be used.
keyStorePassword or keyStorePasswordCipher	Optional. The password used to access the keystore. This property is optional unless you set the keyStore attribute, in which case it is required. If the **fteObfuscate** command has been used then the cipher version of the attribute must be used.

<user>

A user mapping from a WebSphere MQ Managed File Transfer user name to a protocol server user name.

Attribute	Description
name	The user name that is used with WebSphere MQ Managed File Transfer.
serverUserId or serverUserIdCipher	The user name that is used with the protocol server. If the **fteObfuscate** command has been used then the cipher version of the attribute must be used.
serverPassword or serverPasswordCipher	The password for the user name used on the protocol server. If the **fteObfuscate** command has been used then the cipher version of the attribute must be used.
hostKey	The server host SSH fingerprint.

<privateKey>

The private key of a user.

Attribute	Description
keyPassword or keyStorePasswordCipher	The password for the private key. If the **fteObfuscate** command has been used then the cipher version of the attribute must be used.
associationName	A name used for trace and logging.

Protocol bridge properties file format

The ProtocolBridgeProperties.xml file in the agent configuration directory defines properties for protocol file servers.

The ProtocolBridgeProperties.xml file must conform to the ProtocolBridgeProperties.xsd schema. The ProtocolBridgeProperties.xsd schema document is located in the *MQ_INSTALLATION_PATH*/mqft/samples/ schema directory of the MQMFT installation. A template file, ProtocolBridgeProperties.xml, is created by the **fteCreateBridgeAgent** command in the agent configuration directory.

The ProtocolBridgeProperties.xml file is periodically reloaded by the agent and any valid changes to the file will affect the behavior of the agent. The default reload interval is 30 seconds. This interval can be changed by specifying the agent property xmlConfigReloadInterval in the agent.properties file.

Schema

The following schema describes the ProtocolBridgeProperties.xml file.

Note: The maxReconnectRetry and reconnectWaitPeriod attributes are not supported on IBM WebSphere MQ V7.5 or on WebSphere MQ Managed File Transfer V7.0.2, or later.

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
@start_non_restricted_prolog@
Version: %Z% %I% %W% %E% %U% [%H% %T%]

Licensed Materials - Property of IBM

5724-H72

Copyright IBM Corp. 2011 2013 All Rights Reserved.

US Government Users Restricted Rights - Use, duplication or
disclosure restricted by GSA ADP Schedule Contract with
IBM Corp.
@end_non_restricted_prolog@
  -->

<!-- This schema defines the format of the ProtocolBridgeProperties.xml file that is located in the agent
 properties directory of a protocol bridge agent. This XML file
 is used to define properties for protocol servers.

 For more information, see the WebSphere MQ product documentation.
 -->

<schema targetNamespace="http://wmqfte.ibm.com/ProtocolBridgeProperties" elementFormDefault="qualified"
 xmlns="http://www.w3.org/2001/XMLSchema" xmlns:tns="http://wmqfte.ibm.com/ProtocolBridgeProperties">

 <!--
  Example: ProtocolBridgeProperties.xml

  <?xml version="1.0" encoding="UTF-8"?>
  <tns:serverProperties xmlns:tns="http://wmqfte.ibm.com/ProtocolBridgeProperties"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://wmqfte.ibm.com/ProtocolBridgeProperties
        ProtocolBridgeProperties.xsd">
    <tns:credentialsFile path="$HOME/ProtocolBridgeCredentials.xml" />
       <tns:defaultServer name="myserver" />
       <tns:ftpServer name="myserver" host="myhost.hursley.ibm.com" port="1234" platform="windows"
             timeZone="Europe/London" locale="en-GB" fileEncoding="UTF-8"
             listFormat="unix" limitedWrite="false" />
    <tns:sftpServer name="server1" host="myhost.hursley.ibm.com" platform="windows"
             fileEncoding="UTF-8" limitedWrite="false">
      <limits maxListFileNames="10" />
       </tns:sftpServer>
  </tns:serverProperties>
 -->

 <!-- Root element for the document -->
 <element name="serverProperties" type="tns:serverPropertiesType"></element>

 <!--
  A container for all protocol bridge server properties
 -->
 <complexType name="serverPropertiesType">
  <sequence>
   <element name="credentialsFile" type="tns:credentialsFileName" minOccurs="0" maxOccurs="1" />
   <element name="defaultServer" type="tns:serverName" minOccurs="0" maxOccurs="1" />
   <choice minOccurs="0" maxOccurs="unbounded">
    <element name="ftpServer" type="tns:ftpServerType" />
    <element name="sftpServer" type="tns:sftpServerType" />
    <element name="ftpsServer" type="tns:ftpsServerType" />
    <element name="ftpsfgServer" type="tns:ftpsfgServerType" />
    <element name="ftpssfgServer" type="tns:ftpssfgServerType" />
   </choice>
  </sequence>
 </complexType>

 <!--
  A container for a server name
 -->
 <complexType name="serverName">
  <attribute name="name" type="tns:serverNameType" use="required" />
 </complexType>

 <!--
```

```
       A container for a credentials file name
       -->
       <complexType name="credentialsFileName">
        <attribute name="path" type="string" use="required" />
       </complexType>

       <!--
        A container for all the information about an FTP server
        -->
       <complexType name="ftpServerType">
        <sequence>
         <element name="limits" type="tns:generalLimitsType" minOccurs="0" maxOccurs="1" />
        </sequence>
        <attributeGroup ref="tns:ftpServerAttributes"/>
             <attribute name="passiveMode" type="boolean" use="optional" />
       </complexType>

       <!--
        A container for all the information about an SFG FTP server
        -->
       <complexType name="ftpsfgServerType">
        <sequence>
         <element name="limits" type="tns:generalLimitsType" minOccurs="0" maxOccurs="1" />
        </sequence>
        <attributeGroup ref="tns:ftpServerAttributes"/>
       </complexType>

       <!--
        A container for all the information about an SFTP server
        -->
       <complexType name="sftpServerType">
        <sequence>
         <element name="limits" type="tns:sftpLimitsType" minOccurs="0" maxOccurs="1" />
        </sequence>
        <attributeGroup ref="tns:sftpServerAttributes"/>
       </complexType>

       <!--
        A container for all the information about a FTPS server
        -->
       <complexType name="ftpsServerType">
        <sequence>
         <element name="limits" type="tns:generalLimitsType" minOccurs="0" maxOccurs="1" />
        </sequence>
        <attributeGroup ref="tns:ftpsServerAttributes"/>
       </complexType>

       <!--
        A container for all the information about a SFG FTPS server
        -->
       <complexType name="ftpssfgServerType">
        <sequence>
         <element name="limits" type="tns:generalLimitsType" minOccurs="0" maxOccurs="1" />
        </sequence>
        <attributeGroup ref="tns:ftpsServerAttributes"/>
       </complexType>


       <!--
        Attributes common to all server types
        -->
       <attributeGroup name="generalServerAttributes">
        <attribute name="name" type="tns:serverNameType" use="required" />
        <attribute name="host" type="string" use="required" />
        <attribute name="port" type="nonNegativeInteger" use="optional" />
        <attribute name="platform" type="tns:platformType" use="required" />
        <attribute name="fileEncoding" type="string" use="required" />
        <attribute name="limitedWrite" type="boolean" use="optional" />
        <attribute name="controlEncoding" type="string" use="optional" />
       </attributeGroup>

       <!--
        Attributes common to ftp and ftps server types
        -->
       <attributeGroup name="ftpServerAttributes">
        <attributeGroup ref="tns:generalServerAttributes"/>
        <attribute name="timeZone" type="string" use="required" />
        <attribute name="locale" type="tns:localeType" use="required" />
        <attribute name="listFormat" type="tns:listFormatType" use="optional" />
```

```
│    <attribute name="listFileRecentDateFormat" type="tns:dateFormatType" use="optional" />
│    <attribute name="listFileOldDateFormat" type="tns:dateFormatType" use="optional" />
│    <attribute name="monthShortNames" type="tns:monthShortNamesType" use="optional" />
│   </attributeGroup>
│
│   <!--
│    Attributes common to ftps server types
│   -->
│   <attributeGroup name="ftpsServerAttributes">
│    <attributeGroup ref="tns:ftpServerAttributes"/>
│    <attribute name="ftpsType" type="tns:ftpsTypeType" use="optional" />
│    <attribute name="trustStore" type="string" use="required" />
│    <attribute name="trustStoreType" type="string" use="optional" />
│       <attribute name="keyStore" type="string" use="optional" />
│    <attribute name="keyStoreType" type="string" use="optional" />
│    <attribute name="ccc" type="boolean" use="optional" />
│    <attribute name="protFirst" type="boolean" use="optional" />
│    <attribute name="auth" type="string" use="optional" />
│    <attribute name="connectTimeout" type="nonNegativeInteger" use="optional"/>
│    <attribute name="cipherSuiteList" type="string" use="optional" />
│   </attributeGroup>
│
│
│   <!--
│    A container for limit-type attributes for a server. Limit parameters
│    are optional, and if not specified a system default will be used.
│   -->
│   <complexType name="generalLimitsType">
│    <attributeGroup ref="tns:generalLimitAttributes"/>
│   </complexType>
│
│   <complexType name="sftpLimitsType">
│    <attributeGroup ref="tns:generalLimitAttributes"/>
│    <attribute name="connectionTimeout" type="nonNegativeInteger" use="optional" />
│   </complexType>
│
│   <!--
│    Attributes for limits common to all server types
│   -->
│   <attributeGroup name="generalLimitAttributes">
│    <attribute name="maxListFileNames" type="positiveInteger" use="optional" />
│    <attribute name="maxListDirectoryLevels" type="nonNegativeInteger" use="optional" />
│    <attribute name="maxReconnectRetry" type="nonNegativeInteger" use="optional" />
│    <attribute name="reconnectWaitPeriod" type="nonNegativeInteger" use="optional" />
│    <attribute name="maxSessions" type="positiveInteger" use="optional" />
│    <attribute name="socketTimeout" type="nonNegativeInteger" use="optional" />
│   </attributeGroup>
│
│   <!--
│   The type for matching valid server names. Server names must be at least 2 characters in length and
│   are limited to alphanumeric characters and the following characters: ".", "_", "/" and "%".
│   -->
│   <simpleType name="serverNameType">
│         <restriction base="string">
│             <pattern value="[0-9a-zA-Z\._/%\-]{2,}"/>
│         </restriction>
│     </simpleType>
│
│   <!--
│    The types of platform supported.
│   -->
│   <simpleType name="platformType">
│    <restriction base="string">
│    </restriction>
│   </simpleType>
│
│   <!--
│    The type for matching a locale specification.
│   -->
│   <simpleType name="localeType">
│         <restriction base="string">
│             <pattern value="(..)[-_](..)"/>
│         </restriction>
│     </simpleType>
│
│   <!--
│    The types of list format supported (for FTP servers).
│   -->
│   <simpleType name="listFormatType">
```

```
      <restriction base="string">
      </restriction>
     </simpleType>

     <!--
      Date format for FTP client directory listing on an FTP server. This is
      the format to be passed to methods setDefaultDateFormatStr and
      setRecentDateFormatStr for Java class:
      org.apache.commons.net.ftp.FTPClientConfig
     -->
     <simpleType name="dateFormatType">
      <restriction base="string">
      </restriction>
     </simpleType>

     <!--
      A list of language-defined short month names can be specified. These are
      used for translating the directory listing received from the FTP server.
      The format is a string of three character month names separated by "|"
     -->
     <simpleType name="monthShortNamesType">
            <restriction base="string">
                <pattern value="(...\|){11}(...)"/>
            </restriction>
        </simpleType>

        <!--
      The enumerations of the allowed FTPS types: "implicit" & "explicit"
      If not specified the default is "explicit"
     -->
     <simpleType name="ftpsTypeType">
            <restriction base="string">
                <enumeration value="explicit"/>
                <enumeration value="implicit"/>
            </restriction>
        </simpleType>

        <!--
         Attribute Group for SFTP Servers
         -->
        <attributeGroup name="sftpServerAttributes">
      <attributeGroup ref="tns:generalServerAttributes"/>
      <attribute name="cipherList" type="string" use="optional" />
     </attributeGroup>
    </schema>
```

Understanding the `ProtocolBridgeProperties.xml` file

The elements and attributes that are used in the `ProtocolBridgeProperties.xml` file are described in the
following list:

`<serverProperties>`
> Root element of the XML document

> **`<credentialsFile>`**
>> Path to the file containing credentials. For WebSphere MQ V7.5, the value of this property can
>> contain environment variables. For more information, see "Environment variables in WebSphere
>> MQ Managed File Transfer properties" on page 537

> **`<defaultServer>`**
>> The protocol file server that acts as the default server for file transfers

> **`<ftpServer>`**
>> An FTP file server

> **`<sftpServer>`**
>> An SFTP file server

> **`<ftpsServer>`**
>> An FTPS file server

General server attributes that apply to all types of protocol file server:

Attribute	Description
name	Required. The name of the protocol file server. Protocol server names must be at least two characters in length, are not case-sensitive, and are limited to alphanumeric characters and the following characters: • period (.) • underscore (_) • forward slash (/) • percent sign (%)
host	Required. The host name or IP address of the protocol file server that you want to send files to or receive files from.
port	Optional. The port number of the protocol file server that you want to send files to or receive files from.
platform	Required. The platform of the protocol file server that you want to send files to or receive files from. Specify either UNIX or WINDOWS.Set this property according to how you enter paths on your FTP, FTPS, or SFTP server. For example, if you are running an FTP server on Windows but when you log in to the server, you must enter UNIX-style paths (that is, with forward slashes), set this value to UNIX and not WINDOWS. Servers running on Windows often present a UNIX-style file system.
fileEncoding	Required. Defines the character encoding that is used by the file server. This property is used when you transfer files in text mode so that the correct encoding sequences are changed when the files are moved between platforms. For example, UTF-8.
limitedWrite	Optional. The default mode when writing to a file server is to create a temporary file and then rename that file when the transfer has completed. For a file server that is configured as write only, the file is created directly with its final name. The value of this property can be true or false. The default is false.
controlEncoding	Optional. The control encoding value for control messages being sent to the protocol file server. This property affects the encoding of the file name that is used and must be compatible with the control encoding of the protocol file server. The default is UTF-8.

General attributes that apply to FTP and FTPS servers only:

Attribute	Description
timeZone	Required. The time zone of the protocol file server that you want to send files to or receive files from. For example: America/New_York or Asia/Tokyo.
locale	Required. The language that is used on the protocol file server that you want to send files to or receive files from. For example: en_US or ja_JP

Attribute	Description
listFormat	Optional. The listing format that defines the format of the file-listed information that is returned from the protocol file server. Use either Windows or UNIX. The default is UNIX.
listFileRecentDateFormat	Optional. The recent date format (less than a year) for FTP client directory listing on an FTP server. This attribute and the listFileOldDateFormat attribute allow you to redefine the expected date formats that are returned by the protocol file server. The default is as defined by the protocol file server.
listFileOldDateFormat	Optional. The old date format (more than a year) for FTP client directory listing on an FTP server. This attribute and the listFileRecentDateFormat attribute allow you to redefine the expected date formats that are returned by the protocol file server. The default is as defined by the protocol file server.
monthShortNames	Optional. A replacement list of month names that are used to decode date information returned from the protocol file server. This property consists of a list of 12 comma-separated names to override the default locale month values. The default is as defined by the protocol file server.

General attributes that apply to FTP servers only:

Attribute	Description
passiveMode	Optional. Controls whether the connection to the FTP server is passive or active. If you set the value of this property to false, the connection is active. If you set the value to true, the connection is passive. The default is false.

General attributes that apply to FTPS servers only:

Attribute	Description
ftpsType	Optional. Specifies whether the explicit or implicit form of the FTPS protocol is used. The default is explicit.
trustStore	Required. The location of the truststore that is used to determine whether the certificate presented by the FTPS server is trusted.
trustStoreType	Optional. The format of the truststore file. The default is JKS.
keyStore	Optional. The location of the keystore that is used to provide certificate information if challenged by the FTPS server. The default is for the protocol bridge to not be able to connect to FTPS servers that are configured to require the authentication of clients.
keyStoreType	Optional. The format of the keystore file. The default is JKS.

Attribute	Description
ccc	Optional. Selects whether a clear (unencrypted) command channel is used when authentication has completed. The default value is `false`, which means that the command channel remains encrypted for the entire duration of the FTPS session. This attribute is applicable only when the ftpsType is set to `explicit`.
protFirst	Optional. Specifies whether the **USER/PASS** commands are issued to the FTPS server before or after the **PBSZ/PROT** commands. The default value is `false`, which means **USER/PASS** commands are sent first followed by **PBSZ/PROT** commands. This attribute is applicable only when the ftpsType is set to `explicit`.
auth	Optional. Specifies the protocol that is specified as part of the **AUTH** command. A specified protocol will be tried first, then the default is to try TLS, SSL, TLS-C, or TLS-P until the FTPS server does not reject with a 504 reply code. This attribute is applicable only when the ftpsType is set to `explicit`.
cipherSuiteList	Specifies a comma-separated list of one or more cipher suite names. A cipher suite specifies the protocol, hash algorithm, and encryption algorithm that are used, and how many bits are used in the encryption key when data is exchanged between the agent and the FTPS server. The list supplied is used in the negotiation between the agent and the FTPS server. If you do not specify a value, the default set of ciphers enabled by Java is used in the negotiation between the agent and the FTPS server. If you specify a cipher suite that is not supported by the agent, transfers using the FTPS server will fail, even if there are supported cipher suites in the cipher suite list. For a list of valid cipher suite values, see Cipher suites in the IBM SDK and Runtime Environment Java Technology Edition Version 7 Information Center.

`<limits>`
    Container element for attributes that are common to all types of server and for attributes that are specific to a type of server:

General limit attributes that apply to all types of protocol file server:

Attribute	Description
maxListFileNames	Optional. The maximum number of names that are collected when scanning a directory on the protocol file server for file names. The default is 999999999.
maxListDirectoryLevels	Optional. The maximum number of directory levels on the protocol server to recursively scan for file names. The default is 1000.

Attribute	Description
maxReconnectRetry (This attribute is now deprecated.)	**Deprecated**. This attribute is not supported on IBM WebSphere MQ V7.5. or on WebSphere MQ Managed File Transfer V7.0.2, or later. Optional. The maximum number of times a protocol server tries to reconnect before the protocol bridge agent stops trying. The default is 2.
reconnectWaitPeriod (This attribute is now deprecated.)	**Deprecated**. This attribute is not supported on IBM WebSphere MQ V7.5. or on WebSphere MQ Managed File Transfer V7.0.2, or later. Optional. The time period, in seconds, to wait to before attempting to reconnect. The default is 10 seconds.
maxSessions	Optional. The maximum number of sessions for the protocol server. This number must be greater than or equal to the sum of the maximum number of source and destination transfers for the protocol bridge agent. The default is the sum of the values for the agent properties maxSourceTransfers, maxDestinationTransfers, and maxCommandHandlerThreads, plus 1. If these three properties are using their default values of 25, 25, and 5, the maxSessions default is then 56.
socketTimeout	Optional. The socket timeout in seconds. The value of this attribute is used during file streaming. The default is 30 seconds.

Limit attribute that applies to SFTP servers only:

Attribute	Description
connectionTimeout	Optional. The time, in seconds, to wait for a response from the protocol file server to a connection request. A timeout indicates that the protocol file server is not available. The default value is 30 seconds.
cipherList	Optional. Specifies a comma-separated list of ciphers that are used to communicate between the protocol bridge agent and the SFTP server. The ciphers are called in the order that they are specified in this list. The cipher must be available on the server and the client before it can be used. The default is `aes128-cbc,aes192-cbc,aes256-cbc`.

Connect:Direct credentials file format

The `ConnectDirectCredentials.xml` file in the agent configuration directory defines the user names and credential information that the Connect:Direct agent uses to authorize itself with a Connect:Direct node.

The `ConnectDirectCredentials.xml` file must conform to the `ConnectDirectCredentials.xsd` schema. The `ConnectDirectCredentials.xsd` schema document is located in the *MQ_INSTALLATION_PATH*/mqft/samples/schema directory of the MQMFT installation. A sample `ConnectDirectCredentials.xml` file is located in the *MQ_INSTALLATION_PATH*/mqft/samples/credentials directory of the MQMFT installation.

The file `ConnectDirectCredentials.xml` is periodically reloaded by the agent and any valid changes to the file will affect the behavior of the agent. The default reload interval is 30 seconds. This interval can be changed by specifying the agent property **xmlConfigReloadInterval** in the agent.properties file.

Schema

The following schema describes which elements are valid in the ConnectDirectCredentials.xml file.

```
<?xml version="1.0" encoding="UTF-8"?>

<!--
  This schema defines the format of the XML file that is located in the agent properties
  directory of a Connect:Direct bridge agent. The XML file ConnectDirectCredentials.xml
  is used by the default credential validation of the Connect:Direct bridge.
  For more information, see the WebSphere MQ InfoCenter
-->

<schema targetNamespace="http://wmqfte.ibm.com/ConnectDirectCredentials"
        elementFormDefault="qualified"
        xmlns="http://www.w3.org/2001/XMLSchema"
        xmlns:tns="http://wmqfte.ibm.com/ConnectDirectCredentials"

  <!--
    <?xml version="1.0" encoding="UTF-8"?>

    <tns:credentials xmlns:tns="http://wmqfte.ibm.com/ConnectDirectCredentials"
                     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                     xsi:schemaLocation="http://wmqfte.ibm.com/ConnectDirectCredentials
                     ConnectDirectCredentials.xsd">
      <tns:agent name="CDAGENT01">
        <tns:pnode name="cdnode*" pattern="wildcard">
          <tns:user name="MUSR_.*"
            ignorecase="true"
            pattern="regex"
            cdUserId="bob"
            cdPassword="passw0rd"
            pnodeUserId="bill"
            pnodePassword="alacazam">
          <tns:snode name="cdnode2" pattern="wildcard" userId="sue" password="foo"/>
          </tns:user>
        </tns:pnode>
      </tns:agent>
    </tns:credentials>

  -->

  <element name="credentials" type="tns:credentialsType"/>

  <complexType name="credentialsType">
    <sequence>
      <element name="agent" type="tns:agentType" minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
  </complexType>

  <complexType name="agentType">
    <sequence>
      <element name="pnode" type="tns:pnodeType" minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
    <attribute name="name" type="string" use="required"/>
  </complexType>


  <complexType name="pnodeType">
    <sequence>
      <element name="user" type="tns:userType" minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
    <attribute name="name" type="string" use="required"/>
    <attribute name="pattern" type="tns:patternType" use="optional"/>
  </complexType>

  <complexType name="userType">
```

```
    <sequence>
      <element name="snode" type="tns:snodeType" minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
    <attribute name="name" type="string" use="required"/>
    <attribute name="ignorecase" type="boolean" use="optional"/>
    <attribute name="pattern" type="tns:patternType" use="optional"/>
    <attribute name="cdUserId" type="string" use="optional"/>
    <attribute name="cdUserIdCipher" type="string" use="optional"/>
    <attribute name="cdPassword" type="string" use="optional"/>
    <attribute name="cdPasswordCipher" type="string" use="optional"/>
    <attribute name="pnodeUserId" type="string" use="optional"/>
    <attribute name="pnodeUserIdCipher" type="string" use="optional"/>
    <attribute name="pnodePassword" type="string" use="optional"/>
    <attribute name="pnodePasswordCipher" type="string" use="optional"/>
  </complexType>

  <complexType name="snodeType"
    <attribute name="name" type="string" use="required"/>
    <attribute name="pattern" type="tns:patternType" use="optional"/>
    <attribute name="userId" type="string" use="optional"/>
    <attribute name="userIdCipher" type="string" use="optional"/>
    <attribute name="password" type="string" use="optional"/>
    <attribute name="passwordCipher" type="string" use="optional"/>
  </complexType>

  <simpleType name="patternType">
    <restriction base="string">
      <enumeration value="regex"/>
      <enumeration value="wildcard"/>
    </restriction>
  </simpleType>
</schema>
```

Understanding the `ConnectDirectCredentials.xml` file

The elements and attributes used in the `ConnectDirectCredentials.xml` file are described in the following list.

`<credentials>`
Group element containing elements that describe the credentials used by a Connect:Direct bridge agent to connect to a Connect:Direct node.

`<agent>`
Group element containing elements for <pnode> definitions for a named agent.

`<pnode>`
The primary node (PNODE) in the Connect:Direct transfer. This node initiates the connection to the secondary node (SNODE).

Attribute	Description
name	The name of the Connect:Direct node. The value of this attribute can be a pattern that matches many node names.
pattern	Specifies the type of pattern that is used for the value of the name attribute. Valid values for the pattern attribute are • wildcard - wildcards are used • regex - Java regular expressions are used

`<user>`
The WebSphere MQ user that submits the transfer request.

Attribute	Description
name	The user name that is used with WebSphere MQ Managed File Transfer. The value of this attribute can be a pattern that matches many user names.
ignorecase	Specifies whether the case of the name is ignored. Valid values for the ignorecase attribute are • true - the name is not case sensitive • false - the name is case sensitive
pattern	Specifies the type of pattern that is used for the value of the name attribute. Valid values for the pattern attribute are • wildcard - wildcards are used • regex - Java regular expressions are used
cdUserId or cdUserIdCipher	The user name that is used by the Connect:Direct bridge to connect to its associated Connect:Direct node. If the **fteObfuscate** command has been used then the cipher version of the attribute must be used.
cdPassword or cdPasswordCipher	The password associated with the user name specified by the cdUserId attribute. If the **fteObfuscate** command has been used then the cipher version of the attribute must be used.
pnodeUserId or pnodeUserIdCipher	The user name that is used by the Connect:Direct primary node. If the **fteObfuscate** command has been used then the cipher version of the attribute must be used.
pnodePassword or pnodePasswordCipher	The password associated with the user name specified by the pnodeUserId attribute. If the **fteObfuscate** command has been used then the cipher version of the attribute must be used.

<snode>

The Connect:Direct node that performs the role of secondary node (SNODE) during the Connect:Direct file transfer.

Attribute	Description
name	The name of the Connect:Direct node. The value of this attribute can be a pattern that matches many node names.
pattern	Specifies the type of pattern that is used for the value of the name attribute. Valid values for the pattern attribute are • wildcard - wildcards are used • regex - Java regular expressions are used
userId or userIdCipher	The user name used to connect to this node during a file transfer. If the **fteObfuscate** command has been used then the cipher version of the attribute must be used.
password or passwordCipher	The password associated with the user name specified by the userId attribute. If the **fteObfuscate** command has been used then the cipher version of the attribute must be used.

Example

In this example, the Connect:Direct bridge agent connects to the Connect:Direct node pnode1. When a WebSphere MQ user with the user name beginning with the prefix `fteuser` followed by a single character, for example `fteuser2`, requests a transfer involving the Connect:Direct bridge, the Connect:Direct bridge agent will use the user name `cduser` and the password `passw0rd` to connect to the Connect:Direct node pnode1. When the Connect:Direct node pnode1 performs its part of the transfer it uses the user name `pnodeuser` and the password `passw0rd1`.

If the secondary node in the Connect:Direct transfer has a name that begins with the prefix FISH, the node pnode1 uses the user name `fishuser` and the password `passw0rd2` to connect to the secondary node. If the secondary node in the Connect:Direct transfer has a name that begins with the prefix CHIPS, the node pnode1 uses the user name `chipsuser` and the password `passw0rd3` to connect to the secondary node.

```
<?xml version="1.0" encoding="UTF-8"?>

<tns:credentials xmlns:tns="http://wmqfte.ibm.com/ConnectDirectCredentials"
                 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                 xsi:schemaLocation="http://wmqfte.ibm.com/ConnectDirectCredentials ConnectDirectCredentials.xsd">
  <tns:agent name="CDAGENT01">
    <tns:pnode name="pnode1" pattern="wildcard">
      <tns:user name="fteuser?" pattern="wildcard" ignorecase="true"
        cdUserId="cduser" cdPassword="passw0rd"
        pnodeUserId="pnodeuser" pnodePassword="passw0rd1">
      <tns:snode name="FISH*" pattern="wildcard"
        userId="fishuser" password="passw0rd2"/>
      <tns:snode name="CHIPS*" pattern="wildcard"
        userId="chipsuser" password="passw0rd3"/>
      </tns:user>
    </tns:pnode>
  </tns:agent>
</tns:credentials>
```

Connect:Direct process definitions file format

The `ConnectDirectProcessDefinitions.xml` file in the Connect:Direct bridge agent configuration directory specifies the user-defined Connect:Direct process to start as part of the file transfer.

The `ConnectDirectProcessDefinitions.xml` file must conform to the `ConnectDirectProcessDefinitions.xsd` schema. The `ConnectDirectProcessDefinitions.xsd` schema document is located in the *MQ_INSTALLATION_PATH*/mqft/samples/schema directory of the MQMFT installation. A template `ConnectDirectProcessDefinitions.xml` file is created by the **fteCreateCDAgent** command in the agent configuration directory.

The file `ConnectDirectProcessDefinitions.xml` is periodically reloaded by the agent and any valid changes to the file will affect the behavior of the agent. The default reload interval is 30 seconds. This interval can be changed by specifying the agent property xmlConfigReloadInterval in the agent.properties file.

Schema

The following schema describes which elements are valid in the `ConnectDirectProcessDefinitions.xml` file.

```
<schema targetNamespace="http://wmqfte.ibm.com/ConnectDirectProcessDefinitions"
        elementFormDefault="qualified"
        xmlns="http://www.w3.org/2001/XMLSchema"
        xmlns:tns="http://wmqfte.ibm.com/ConnectDirectProcessDefinitions">

    <element name="cdprocess" type="tns:cdprocessType"></element>

    <complexType name="cdprocessType">
        <sequence>
            <element name="processSet" type="tns:processSetType"
```

```
                    minOccurs="0"      maxOccurs="unbounded"></element>
        </sequence>
    </complexType>


    <complexType name="processSetType">
        <sequence>
            <element name="condition" type="tns:conditionType"
                    minOccurs="0"    maxOccurs="1" />
            <element name="process"   type="tns:processType"
                    minOccurs="1"    maxOccurs="1" />
        </sequence>
    </complexType>


    <complexType name="conditionType">
        <choice minOccurs="0" maxOccurs="unbounded">
            <element name="match" type="tns:matchType" />
            <element name="defined" type="tns:definedType" />
        </choice>
    </complexType>


    <complexType name="matchType">
        <attribute name="variable" type="string" use="required" />
        <attribute name="value" type="string" use="required" />
        <attribute name="pattern" type="tns:patternType" use="optional" />
    </complexType>


    <complexType name="definedType">
        <attribute name="variable" type="string" use="required" />
    </complexType>


    <complexType name="processType">
        <sequence>
            <element name="preTransfer"          type="tns:transferType"
                    minOccurs="0"                maxOccurs="1" />
            <element name="transfer"             type="tns:transferType"
                    minOccurs="0"                maxOccurs="1" />
            <element name="postTransferSuccess" type="tns:transferType"
                    minOccurs="0"                maxOccurs="1" />
            <element name="postTransferFailure" type="tns:transferType"
                    minOccurs="0"                maxOccurs="1" />
        </sequence>
    </complexType>


    <complexType name="transferType">
        <attribute name="process" type="string" use="required" />
    </complexType>


    <simpleType name="patternType">
        <restriction base="string">
            <enumeration value="regex" />
            <enumeration value="wildcard" />
        </restriction>
    </simpleType>

</schema>
```

Understanding the `ConnectDirectProcessDefinitions.xml` file

The elements and attributes used in the `ConnectDirectProcessDefinitions.xml` file are described in the following list.

cdProcess
The root element of the XML document.

processSet
Group element containing all the information about a set of user-defined processes.

condition
>   Group element containing the conditions that a transfer is tested against to determine whether the set
>   of processes contained in the processSet element are used.
>
>   **match**
>>   A condition that tests whether a the value of a variable matches a given value.

Attribute	Description
variable	Specifies a variable. The value of this variable is compared with the value of the value attribute. The variable is an intrinsic symbol. For more information, see "Substitution variables for use with user-defined Connect:Direct processes" on page 686.
value	Specifies a pattern to match against the value of the variable specified by the variable attribute.
pattern	Specifies the type of pattern that is used for the value of the value attribute. Valid values for the pattern attribute are • `wildcard` - wildcards are used • `regex` - Java regular expressions are used This attribute is optional and the default is `wildcard`.

>   **defined**
>>   A condition that tests whether a variable has been defined.

Attribute	Description
variable	Specifies a variable. If this variable exists, the match condition is satisfied. The variable is an intrinsic symbol. For more information, see "Substitution variables for use with user-defined Connect:Direct processes" on page 686.

process
>   Group element containing the information about where to locate the Connect:Direct processes to call
>   when a match is found.
>
>   **transfer**
>>   The Connect:Direct process to call during a transfer request.

Attribute	Description
process	Optional. Specifies the name of a file that contains a Connect:Direct process to call during a transfer request. The file path is relative to the Connect:Direct bridge agent configuration directory. This attribute is optional, the default is to use a process generated by MQMFT. For WebSphere MQ V7.5, the value of this property can contain environment variables. For more information, see "Environment variables in WebSphere MQ Managed File Transfer properties" on page 537

Example

In this example, there are three processSet elements.

The first processSet element specifies that if a transfer request has a **%FTESNODE** variable with a value that matches the pattern `Client*` and a **%FTESUSER** variable with a value of `Admin`, the Connect:Direct bridge agent submits the Connect:Direct process located in the *agent_configuration_directory*/`AdminClient.cdp` as part of the transfer.

The second processSet element specifies that if a transfer request has a **%FTESNODE** variable with a value that matches the pattern `Client*`, the Connect:Direct bridge agent submits the Connect:Direct process located in the *agent_configuration_directory*/`Client.cdp` as part of the transfer. The Connect:Direct bridge agent reads the processSet elements in the order that they are defined, and if it finds a match, it uses the first match and does not look for another match. For transfer requests that match the conditions of both the first and second processSet, the Connect:Direct bridge agent calls only the processes specified by the first processSet.

The third processSet element has no conditions and matches all transfers. If the transfer request does not match the conditions of the first or second processSet, the Connect:Direct bridge agent submits the Connect:Direct process specified by the third condition. This process is located in the *agent_configuration_directory*/`Default.cdp` as part of the transfer.

```
<?xml version="1.0" encoding="UTF-8"?>
<tns:cdprocess xmlns:tns="http://wmqfte.ibm.com/ConnectDirectProcessDefinitions"
               xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xsi:schemaLocation="http://wmqfte.ibm.com/ConnectDirectProcessDefinitions ConnectDirectProcessDefinitions.xsd">

    <tns:processSet>
        <tns:condition>
            <tns:match variable="%FTESNODE" value="Client*" pattern="wildcard" />
            <tns:match variable="%FTESUSER" value="Admin" pattern="wildcard" />
        </tns:condition>
        <tns:process>
            <tns:transfer process="AdminClient.cdp" />
        </tns:process>
    </tns:processSet>

    <tns:processSet>
        <tns:condition>
            <tns:match variable="%FTESNODE" value="Client*" pattern="wildcard" />
        </tns:condition>
        <tns:process>
            <tns:transfer process="Client.cdp" />
        </tns:process>
    </tns:processSet>

    <tns:processSet>
        <tns:process>
            <tns:transfer process="Default.cdp" />
        </tns:process>
    </tns:processSet>

</tns:cdprocess>
```

Connect:Direct node properties file format

The `ConnectDirectNodeProperties.xml` file in the Connect:Direct bridge agent configuration directory specifies information about remote Connect:Direct nodes that are involved in a file transfer.

The `ConnectDirectNodeProperties.xml` file must conform to the `ConnectDirectNodeProperties.xsd` schema. The `ConnectDirectNodeProperties.xsd` schema document is located in the *MQ_INSTALLATION_PATH*/mqft/samples/schema directory of the MQMFT installation. A template `ConnectDirectNodeProperties.xml` file is created by the **fteCreateCDAgent** command in the agent configuration directory.

The file `ConnectDirectNodeProperties.xml` is periodically reloaded by the agent and any valid changes to the file will affect the behavior of the agent. The default reload interval is 30 seconds. This interval can be changed by specifying the agent property xmlConfigReloadInterval in the agent.properties file.

Schema

The following schema describes which elements are valid in the `ConnectDirectNodeProperties.xml` file.

```
<?xml version="1.0" encoding="UTF-8"?>

<schema targetNamespace="http://wmqfte.ibm.com/ConnectDirectNodeProperties"
  elementFormDefault="qualified"
  xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:tns="http://wmqfte.ibm.com/ConnectDirectNodeProperties">

  <element name="nodeProperties" type="tns:nodePropertiesType"></element>

  <complexType name="nodePropertiesType">
    <sequence>
      <element name="credentialsFile" type="tns:credentialsFileName" minOccurs="0" maxOccurs="1" />
      <element name="node" type="tns:nodeType" minOccurs="0" maxOccurs="unbounded"></element>
    </sequence>
  </complexType>

  <complexType name="nodeType">
    <attribute name="name" type="string" use="required" />
    <attribute name="pattern" type="tns:patternType" use="optional" />
    <attribute name="type" type="string" use="required" />
  </complexType>

  <simpleType name="patternType">
    <restriction base="string">
      <enumeration value="regex" />
      <enumeration value="wildcard" />
    </restriction>
  </simpleType>

</schema>
```

Understanding the `ConnectDirectNodeProperties.xml` file

The elements and attributes used in the ConnectDirectNodeProperties.xml file are described in the following list.

nodeProperties
> Root element of the XML document.

credentialsFile
> Path to the credentials file where sensitive information is stored. For WebSphere MQ V7.5, the value of this property can contain environment variables. For more information, see "Environment variables in WebSphere MQ Managed File Transfer properties" on page 537

node
> Specifies one or more Connect:Direct nodes.

Attribute	Description
name	A pattern that identifies the names of Connect:Direct nodes that use the definitions specified by the node element. Pattern matching is not case sensitive.
pattern	Specifies the type of pattern that is used for the value of the name attribute. Valid values for the pattern attribute are: • wildcard - wildcards are used • regex - Java regular expressions are used For information about the types of regular expressions used by MQMFT, see "Regular expressions used by WebSphere MQ Managed File Transfer" on page 685.

Attribute	Description
type	Specifies the operating system type of the Connect:Direct node or nodes that match the pattern given by the name attribute. Valid values for the type attribute are: • `Windows` - the node runs on Windows • `UNIX` - the node runs on UNIX or Linux The value of this attribute is not case sensitive.

Example

In this example, the file specifies that all Connect:Direct nodes that have a name that begins with "cdnodew" run on a Windows platform and all Connect:Direct nodes that have a name that begins with "cdnodeu" run on a UNIX platform. The file specifies that all other Connect:Direct nodes run on a UNIX platform. The Connect:Direct bridge agent searches for matches from the start of the file to the end of the file and uses the first match that it finds. The Connect:Direct credentials file has been specified as `ConnectDirectCredentials.xml`.

```
<?xml version="1.0" encoding="UTF-8"?>
<tns:nodeProperties xmlns:tns="http://wmqfte.ibm.com/ConnectDirectNodeProperties"
                    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                    xsi:schemaLocation="http://wmqfte.ibm.com/ConnectDirectNodeProperties
                                                    ConnectDirectNodeProperties.xsd">

    <tns:credentialsFile path="ConnectDirectCredentials.xml" />
    <tns:node name="cdnodew*" pattern="wildcard" type="windows" />
    <tns:node name="cdnodeu.*" pattern="regex" type="unix" />
    <tns:node name="*" pattern="wildcard" type="unix" />

</tns:nodeProperties>
```

fteutils.xsd schema file

This schema defines elements and types used by many of the other WebSphere MQ Managed File Transfer schemas.

Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
@start_non_restricted_prolog@
Version: %Z% %I% %W% %E% %U% [%H% %T%]

Licensed Materials - Property of IBM

5724-H72

Copyright IBM Corp. 2008, 2012 All Rights Reserved.

US Government Users Restricted Rights - Use, duplication or
disclosure restricted by GSA ADP Schedule Contract with
IBM Corp.
@end_non_restricted_prolog@
-->

<!--
This schema defines elements and types used by many of the other MQMFT schemas.
For more information about MQMFT XML message formats, see
http://publib.boulder.ibm.com/infocenter/wmqfte/v7r0/index.jsp?topic=/com.ibm.wmqfte.doc/message_formats.htm
-->
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <!--
        Defines the version type 1.00 - 99.00
        <transaction version= 1.00
    -->
```

```
<xsd:simpleType name="versionType">
    <xsd:restriction base="xsd:string">
        <xsd:pattern value="[0-9]+\.[0-9][0-9]"/>
    </xsd:restriction>
</xsd:simpleType>
<!--
     Defines the transaction reference
     <transaction version= 1.00 ID="414d5120514d31202020202020202020205ecf0a4920011802"
 -->
<xsd:simpleType name="IDType">
    <xsd:restriction base="xsd:string">
        <xsd:pattern value="[0-9a-fA-F]{48}"/>
    </xsd:restriction>
</xsd:simpleType>
<!--
     This is an alias for hostUserIDType.
     Here to allow addition of attributes on originator elements
 -->
<xsd:complexType name="origRequestType">
    <xsd:complexContent>
        <xsd:extension base="hostUserIDType">
            <xsd:sequence>
                <xsd:element name="webBrowser" type="xsd:string" minOccurs="0" maxOccurs="1"/>
                <xsd:element name="webUserID" type="xsd:string" minOccurs="0" maxOccurs="1"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>
<!--
    Defines a Delete originator as a machine and user pair
    <hostName>myMachine</hostName>
    <userName>myUserId</userName>
 -->
<xsd:complexType name="origDeleteType">
    <xsd:sequence>
        <xsd:element name="delete" type="hostUserIDType" maxOccurs="1" minOccurs="0"/>
    </xsd:sequence>
</xsd:complexType>
<!--
    Defines a machine, user, MQMD userID triple
    <hostName>myMachine</hostName>
    <userID>myUserId</userID>
    <mqmdUserID>MQMDUSERID</mqmdUserID>
 -->
<xsd:complexType name="hostUserIDType">
    <xsd:sequence>
        <xsd:element name="hostName" type="xsd:string" minOccurs="0" maxOccurs="1"/>
        <xsd:element name="userID" type="xsd:string" minOccurs="0" maxOccurs="1"/>
        <xsd:element name="mqmdUserID" type="xsd:string" minOccurs="0" maxOccurs="1"/>
    </xsd:sequence>
</xsd:complexType>
<!--
    Define the destinationAgent with agent and queue manager name
    <destinationAgent agent="there" QMgr="far" agentType="BRIDGE" bridgeURL="ftp://server.example.net:21" bridgeNode="D
    optional agentType attribute expected to be one of STANDARD, BRIDGE, WEB_GATEWAY, EMBEDDED, CD_BRIDGE
-->
<xsd:complexType name="agentType">
    <xsd:attribute name="agent" type="xsd:string" use="required"/>
    <xsd:attribute name="agentType" type="xsd:string" use="optional"/>
    <xsd:attribute name="QMgr" type="xsd:string" use="optional"/>
    <xsd:attribute name="bridgeURL" type="xsd:string" use="optional"/>
    <xsd:attribute name="bridgeNode" type="xsd:string" use="optional"/>
    <xsd:attribute name="pnode" type="xsd:string" use="optional"/>
    <xsd:attribute name="snode" type="xsd:string" use="optional"/>
</xsd:complexType>
<!--
    Defines the status type; attr/resultCode and 0 or many supplements
```

```
    There may also be additional command specific data, either: transfer, ping or call data
    <status resultCode="8011">
        <supplement>Azionamento del USB</supplement>
        <supplement>morto come norweign azzurro</supplement>
    </status>
  -->
<xsd:complexType name="statusType">
    <xsd:sequence>
        <xsd:element name="supplement" type="xsd:string" maxOccurs="unbounded" minOccurs="0"/>
        <xsd:choice>
            <xsd:element name="filespace" type="fileSpaceReplyType" minOccurs="0" maxOccurs="1"/>
        </xsd:choice>
    </xsd:sequence>
    <xsd:attribute name="resultCode" type="resultCodeType" use="required"/>
</xsd:complexType>
<!--
    Defines the filespace type for use with communication between a web agent
    and a web gateway
    <filespace name="" location=""><Quota bytes=""/></filespace>
 -->
<xsd:complexType name="fileSpaceReplyType">
    <xsd:attribute name="name" use="required" type="xsd:string"/>
    <xsd:attribute name="location" use="required" type="xsd:string"/>
    <xsd:attribute name="quota" use="required" type="xsd:long"/>
</xsd:complexType>
<!--
    Defines the destinationAgent with agent and queue manager name, plus connection details.
    <destinationAgent agent="there" QMgr="far"/>
 -->
<xsd:complexType name="agentClientType">
    <xsd:attribute name="agent" type="xsd:string" use="required"/>
    <xsd:attribute name="QMgr" type="xsd:string" use="optional"/>
    <xsd:attribute name="hostName" type="xsd:string" use="optional"/>
    <xsd:attribute name="portNumber" type="xsd:nonNegativeInteger" use="optional"/>
    <xsd:attribute name="channel" type="xsd:string" use="optional"/>
</xsd:complexType>
<!--
    Defines the fileURI type as string
    <file encoding="UTF8" EOL="CR">C:/from/here.txt</file>
 -->
<xsd:complexType name="fileType">
    <xsd:simpleContent>
        <xsd:extension base="xsd:string">
            <xsd:attribute name="filespace" type="filespaceNameType" use="optional"/>
            <xsd:attribute name="alias" type="xsd:string" use="optional"/>
            <xsd:attribute name="encoding" type="encodingType" use="optional"/>
            <xsd:attribute name="EOL" type="EOLType" use="optional"/>
```

V 7.5.0.3 `<xsd:attribute name="size" type="xsd:long" use="optional"/> <xsd:attribute name="size" type="xsd:int" use="optional"/> <xsd:attribute name="last-modified" type="xsd:dateTime" use="optional"/> <xsd:attribute name="delimiter" type="xsd:string" use="optional"/> <xsd:attribute name="delimiterType" type="xsd:string" use="optional"/> <xsd:attribute name="delimiterPosition" type="delimiterPositionType" use="optional"/> <xsd:attribute name="includeDelimiterInFile" type="xsd:boolean" use="optional"/> <xsd:attribute name="keepTrailingSpaces" type="xsd:boolean" use="optional"/> </xsd:extension> </xsd:simpleContent> </xsd:complexType> <!-- Defines the filespace type as string <filespace> <name>tarquin</name> </filespace> --> <xsd:complexType name="filespaceType"> <xsd:sequence> <xsd:element name="name" type="filespaceNameType"/> </xsd:sequence> </xsd:complexType> <!-- Defines a name element <name>bob</name> --> <xsd:simpleType name="filespaceNameType"> <xsd:restriction base="xsd:string"/> </xsd:simpleType> <!-- Defines the accepted choices for the persistent attribute. --> <xsd:simpleType name="persistenceType"> <xsd:restriction base="xsd:token"> <xsd:enumeration value="true"/> <xsd:enumeration value="false"/> <xsd:enumeration value="qdef"/> </xsd:restriction> </xsd:simpleType> <!-- Defines the queueURI type as string with all supported attributes.`

<queue>QUEUE@QM</queue> --> <xsd:complexType name="queueType"> <xsd:simpleContent>
<xsd:extension base="xsd:string"> <xsd:attribute name="persistent" type="persistenceType"
use="optional"/> <xsd:attribute name="eofMarker" type="xsd:boolean" use="optional"/> <xsd:attribute
name="setMqProps" type="xsd:boolean" use="optional"/> <xsd:attribute name="split" type="xsd:boolean"
use="optional"/> <xsd:attribute name="useGroups" type="xsd:boolean" use="optional"/> <xsd:attribute
name="delimiter" type="xsd:string" use="optional"/> <xsd:attribute name="delimiterType"
type="xsd:string" use="optional"/> <xsd:attribute name="delimiterPosition" type="delimiterPositionType"
use="optional"/> <xsd:attribute name="includeDelimiterInMessage" type="xsd:boolean" use="optional"/>
<xsd:attribute name="groupId" type="groupIdType" use="optional"/> <xsd:attribute name="messageId"
type="messageIdType" use="optional"/> <xsd:attribute name="messageCount"
type="xsd:nonNegativeInteger" use="optional"/> <xsd:attribute name="messageLength"
type="xsd:nonNegativeInteger" use="optional"/> <xsd:attribute name="waitTime"
type="xsd:nonNegativeInteger" use="optional"/> <xsd:attribute name="encoding" type="encodingType"
use="optional"/> <xsd:attribute name="EOL" type="EOLType" use="optional"/> <xsd:attribute
name="unrecognisedCodePage" type="unrecognisedCodePageType" use="optional"/> </xsd:extension>
</xsd:simpleContent> </xsd:complexType> <!-- Defines the accepted values for the delimiterPosition
attribute. --> <xsd:simpleType name="delimiterPositionType"> <xsd:restriction base="xsd:token">
<xsd:enumeration value="postfix"/> <xsd:enumeration value="prefix"/> </xsd:restriction>
</xsd:simpleType> <!-- Defines the groupId type <queue
groupId="414d5120514d31202020202020202020205ecf0a4920011802"> Also allow a substitution variable of the
form ${variable} --> <xsd:simpleType name="groupIdType"> <xsd:restriction base="xsd:string">
<xsd:pattern value="[0-9a-fA-F]{48}|\$\{.*\}"/> </xsd:restriction> </xsd:simpleType> <!-- Defines the
messageId type <queue messageId="414d5120514d31202020202020202020205ecf0a4920011802"> Also allow a
substitution variable of the form ${variable} --> <xsd:simpleType name="messageIdType"> <xsd:restriction
base="xsd:string"> <xsd:pattern value="[0-9a-fA-F]{48}|\$\{.*\}"/> </xsd:restriction> </xsd:simpleType>
<!-- Defines the accepted values for the unrecognisedCodePage attribute. --> <xsd:simpleType
name="unrecognisedCodePageType"> <xsd:restriction base="xsd:token"> <xsd:enumeration value="fail"/>
<xsd:enumeration value="binary"/> </xsd:restriction> </xsd:simpleType> <!-- Defines a single source
file reference <source type="file" recursive="false" disposition="leave"> <file>filename</file> <source/>
--> <xsd:complexType name="fileSourceType"> <xsd:sequence> <xsd:choice> <xsd:element name="file"
type="fileType"/> <xsd:element name="queue" type="queueType"/> </xsd:choice> </xsd:sequence>
<xsd:attribute name="type" type="SourceType" use="optional"/> <xsd:attribute name="recursive"
type="xsd:boolean" use="optional"/> <xsd:attribute name="disposition" type="sourceDispositionType"
use="optional"/> <xsd:attribute name="correlationString1" type="xsd:string" use="optional"/>
<xsd:attribute name="correlationNum1" type="xsd:nonNegativeInteger" use="optional"/> <xsd:attribute
name="correlationBoolean1" type="xsd:boolean" use="optional"/> </xsd:complexType> <!-- Defines the
enumeration values for source type type="file|queue" --> <xsd:simpleType name="SourceType">
<xsd:restriction base="xsd:token"> <xsd:enumeration value="file"/> <xsd:enumeration value="queue"/>
</xsd:restriction> </xsd:simpleType> <!-- Defines the enumeration values for source disposition
disposition="leave|delete" --> <xsd:simpleType name="sourceDispositionType"> <xsd:restriction
base="xsd:token"> <xsd:enumeration value="leave"/> <xsd:enumeration value="delete"/>
</xsd:restriction> </xsd:simpleType> <!-- Defines a single destination file reference <destination
type="file" exist="overwrite"> <file>filename</file> <destination/> --> <xsd:complexType
name="fileDestinationType"> <xsd:sequence> <xsd:choice> <xsd:element name="file" type="fileType"/>
<xsd:element name="filespace" type="filespaceType"/> <xsd:element name="queue" type="queueType"/>
</xsd:choice> <xsd:element name="attributes" type="attributeType" minOccurs="0" maxOccurs="1"/>
</xsd:sequence> <xsd:attribute name="type" type="DestinationType" use="optional"/> <xsd:attribute
name="exist" type="existType" use="optional"/> <xsd:attribute name="correlationString1"
type="xsd:string" use="optional"/> <xsd:attribute name="correlationNum1"
type="xsd:nonNegativeInteger" use="optional"/> <xsd:attribute name="correlationBoolean1"
type="xsd:boolean" use="optional"/> </xsd:complexType> <!-- Defines the enumeration values for
destination file type type="file|directory|queue|dataset|pds|filespace" 'dataset' and 'pds' only apply to
z/OS environments. --> <xsd:simpleType name="DestinationType"> <xsd:restriction base="xsd:token">
<xsd:enumeration value="file"/> <xsd:enumeration value="directory"/> <xsd:enumeration
value="queue"/> <xsd:enumeration value="dataset"/> <xsd:enumeration value="pds"/>
<xsd:enumeration value="filespace"/> </xsd:restriction> </xsd:simpleType> <!-- Defines the

enumerations values for file exists on destination behavior exist="error|overwrite" --> <xsd:simpleType name="existType"> <xsd:restriction base="xsd:token"> <xsd:enumeration value="error"/> <xsd:enumeration value="overwrite"/> </xsd:restriction> </xsd:simpleType> <!-- Defines one or more file attributes <destination encoding=? CFLF=?> <file>filename</file> <attributes> <attribute>DIST(MIRRORED,UPDATE)</attribute> </attributes> <destination/> --> <xsd:complexType name="attributeType"> <xsd:sequence> <xsd:element name="attribute" type="xsd:string" maxOccurs="unbounded" minOccurs="1"/> </xsd:sequence> </xsd:complexType> <!-- Defines a single file reference <source encodings=? CFLF=?> <file>filename</file> <checksum method="MD5">3445678</checksum> <source/> .. or .. <destination encoding=? CFLF=?> <file>filename</file> <checksum method="MD5">3445678</checksum> <destination/> --> <xsd:complexType name="fileCheckSumType"> <xsd:sequence> <xsd:element name="file" type="fileType"/> <xsd:element name="checksum" type="checksumType" maxOccurs="1" minOccurs="0"/> </xsd:sequence> </xsd:complexType> <!-- Defines the checksum type and method <checksum method="MD5|none">3445678</checksum> --> <xsd:complexType name="checksumType"> <xsd:simpleContent> <xsd:extension base="xsd:string"> <xsd:attribute name="method" type="checkSumMethod" use="required"/> </xsd:extension> </xsd:simpleContent> </xsd:complexType> <!-- Defines the enumeration values for checksumMethod <checksum method="MD5|none">3445678</checksum> Note: upper case is used since MD5 is acronym and normally written upper case. --> <xsd:simpleType name="checkSumMethod"> <xsd:restriction base="xsd:token"> <xsd:enumeration value="none"/> <xsd:enumeration value="MD5"/> </xsd:restriction> </xsd:simpleType> <!-- Defines the enumeration values for agentRole agentRole="sourceAgent|destinationAgent" --> <xsd:simpleType name="agentRoleType"> <xsd:restriction base="xsd:token"> <xsd:enumeration value="sourceAgent"/> <xsd:enumeration value="destinationAgent"/> <xsd:enumeration value="callAgent"/> </xsd:restriction> </xsd:simpleType> <!-- Defines the enumeration values for mode. text, binary or a substitution variable <item mode="binary|text||${variableName}"> --> <xsd:simpleType name="modeType"> <xsd:restriction base="xsd:string"> <xsd:pattern value="binary|text|$\{.*\}"/> </xsd:restriction> </xsd:simpleType> <!-- Defines the enumeration values for EOL <file EOL="LF|CRLF"> --> <xsd:simpleType name="EOLType"> <xsd:restriction base="xsd:token"> <xsd:enumeration value="LF"/> <xsd:enumeration value="CRLF"/> </xsd:restriction> </xsd:simpleType> <!-- Defines the encoding type as a string --> <xsd:simpleType name="encodingType"> <xsd:restriction base="xsd:string"/> </xsd:simpleType> <!-- <schedule> <submit timebase="source"|"admin">2008-12-07T16:07"</submit> <repeat> <frequency interval="hours">2</frequency> <expireTime>2008-12-0816:07</exipreTime> </repeat> </schedule> --> <xsd:complexType name="scheduleType"> <xsd:sequence> <xsd:element name="submit" type="submitType" maxOccurs="1" minOccurs="1"/> <xsd:element name="repeat" type="repeatType" maxOccurs="1" minOccurs="0"/> </xsd:sequence> </xsd:complexType> <!-- <submit timebase="source|admin|UTC">2008-12-07T16:07"</submit> --> <xsd:complexType name="submitType"> <xsd:simpleContent> <xsd:extension base="noZoneTimeType"> <xsd:attribute name="timebase" type="timebaseType" use="required"/> <xsd:attribute name="timezone" type="xsd:string" use="required"/> </xsd:extension> </xsd:simpleContent> </xsd:complexType> <!-- <repeat> <frequency interval="hours">2</frequency> ..optionally.. <expireTime>2008-12-0816:07</expireTime> ..or.. <expireCount>2</expireCount> </repeat> --> <xsd:complexType name="repeatType"> <xsd:sequence> <xsd:element name="frequency" type="freqType" maxOccurs="1" minOccurs="1"/> <xsd:choice minOccurs="0"> <xsd:element name="expireTime" type="noZoneTimeType"/> <xsd:element name="expireCount" type="positiveIntegerType"/> </xsd:choice> </xsd:sequence> </xsd:complexType> <!-- <frequency interval="hours">2</frequency> --> <xsd:complexType name="freqType"> <xsd:simpleContent> <xsd:extension base="positiveIntegerType"> <xsd:attribute name="interval" type="intervalType" use="required"/> </xsd:extension> </xsd:simpleContent> </xsd:complexType> <!-- Defines positive integer type i.e., 1+ --> <xsd:simpleType name="positiveIntegerType"> <xsd:restriction base="xsd:integer"> <xsd:minInclusive value="1"/> </xsd:restriction> </xsd:simpleType> <!-- Defines the interval enumeration values of "minutes","hours","days", "weeks", "months" or "years" --> <xsd:simpleType name="intervalType"> <xsd:restriction base="xsd:token"> <xsd:enumeration value="minutes"/> <xsd:enumeration value="hours"/> <xsd:enumeration value="days"/> <xsd:enumeration value="weeks"/> <xsd:enumeration value="months"/> <xsd:enumeration value="years"/> </xsd:restriction> </xsd:simpleType> <!-- Defines the interval of either "source","admin" or "UTC" source = use timezone of the source Agent. admin = use timezone of the administrator executing the command script. UTC = Timezone is UTC. --> <xsd:simpleType name="timebaseType">

```
<xsd:restriction base="xsd:token"> <xsd:enumeration value="source"/> <xsd:enumeration
value="admin"/> <xsd:enumeration value="UTC"/> </xsd:restriction> </xsd:simpleType> <!-- Defines a
date and time without a time zone (2008-12-08T16:07) --> <xsd:simpleType name="noZoneTimeType">
<xsd:restriction base="xsd:string"> <xsd:pattern value="[\n\r\t ]*\d{4}\-(0[1-9]|1[0-2])\-(0[1-9]|[1-2][0-
9]|3[0-1])T([0-1][0-9]|2[0-3]):[0-5][0-9]([\+\-]\d{4}|Z)?[\n\r\t ]*"/> </xsd:restriction>
</xsd:simpleType> <!-- Defines the ID element, e.g. 56 --> <xsd:simpleType name="idType">
<xsd:restriction base="xsd:string"/> </xsd:simpleType> <!-- Defines the resultCode type -2 - 9999 <status
resultCode="8011"> --> <xsd:simpleType name="resultCodeType"> <xsd:restriction base="xsd:int">
<xsd:minInclusive value="-2"/> <xsd:maxInclusive value="9999"/> </xsd:restriction> </xsd:simpleType>
<!-- Define the metaDataSet type comprising one or more key value pairs <metaDataSet> <metaData
key="name">value</metaData> <metaData key="name">value</metaData> </metaDataSet> -->
<xsd:complexType name="metaDataSetType"> <xsd:sequence> <xsd:element name="metaData"
type="metaDataType" maxOccurs="unbounded" minOccurs="1"/> </xsd:sequence> </xsd:complexType>
<!-- Define the metaData type which is made up of a key and a value <metaData key="name">value</
metaData> --> <xsd:complexType name="metaDataType"> <xsd:simpleContent> <xsd:extension
base="xsd:string"> <xsd:attribute name="key" type="xsd:string" use="required"/> </xsd:extension>
</xsd:simpleContent> </xsd:complexType> <!-- Defines containing element for triggers <trigger
log="yes"> <fileExist comparison="=" value="Exist">file1</fileExist> <fileSize comparison=">="
value="1GB">file1</fileSize> </trigger> --> <xsd:complexType name="triggerType"> <xsd:choice
minOccurs="0" maxOccurs="unbounded"> <xsd:element name="fileExist" type="fileExistTriggerType"
maxOccurs="unbounded" minOccurs="1"/> <xsd:element name="fileSize" type="fileSizeTriggerType"
maxOccurs="unbounded" minOccurs="1"/> </xsd:choice> <xsd:attribute name="log"
type="logEnabledType" use="required"/> </xsd:complexType> <!-- Defines the file exists trigger type
<fileExist comparison="=" value="Exist">file1</trigger> --> <xsd:complexType
name="fileExistTriggerType"> <xsd:simpleContent> <xsd:extension base="xsd:string"> <xsd:attribute
name="comparison" type="comparisonFileExistTriggerType" use="required"/> <xsd:attribute
name="value" type="valueFileExistTriggerType" use="required"/> </xsd:extension>
</xsd:simpleContent> </xsd:complexType> <!-- Defines file size trigger type <fileSize comparison="="
value="1GB">file1,file2,file3</trigger> --> <xsd:complexType name="fileSizeTriggerType">
<xsd:simpleContent> <xsd:extension base="xsd:string"> <xsd:attribute name="comparison"
type="comparisonFileSizeTriggerType" use="required"/> <xsd:attribute name="value"
type="valueFileSizeTriggerType" use="required"/> </xsd:extension> </xsd:simpleContent>
</xsd:complexType> <!-- Defines the enumeration values for file exists trigger conditions
valueFileExistTriggerType="exist|noexist" --> <xsd:simpleType name="valueFileExistTriggerType">
<xsd:restriction base="xsd:token"> <xsd:enumeration value="exist"/> <xsd:enumeration
value="noexist"/> </xsd:restriction> </xsd:simpleType> <!-- Defines the enumeration values for file
exists trigger comparison operator comparisonFileExistTriggerType="="|"!=" --> <xsd:simpleType
name="comparisonFileExistTriggerType"> <xsd:restriction base="xsd:token"> <xsd:enumeration
value="="/> <xsd:enumeration value="!="/> </xsd:restriction> </xsd:simpleType> <!-- Defines the
enumeration values for file size trigger comparison operator comparisonFileSizeTriggerType=">=" -->
<xsd:simpleType name="comparisonFileSizeTriggerType"> <xsd:restriction base="xsd:token">
<xsd:enumeration value="&amp;gt;="/> </xsd:restriction> </xsd:simpleType> <!-- Defines the file size
value pattern <fileSize comparison=">=" value="10|10B|10KB|10MB|10GB">file1</fileSize> -->
<xsd:simpleType name="valueFileSizeTriggerType"> <xsd:restriction base="xsd:string"> <xsd:pattern
value="[0123456789]+([bB]|[kK][bB]|[mM][bB]|[gG][bB]|)"/> </xsd:restriction> </xsd:simpleType> <!--
Defines the enumeration values for trigger logging enabled flag <trigger log="yes|no"> -->
<xsd:simpleType name="logEnabledType"> <xsd:restriction base="xsd:token"> <xsd:enumeration
value="yes"/> <xsd:enumeration value="no"/> </xsd:restriction> </xsd:simpleType> <!-- Defines the
reply type <reply QMGR="QMGR name" persistent="true">Queue Name</reply> --> <xsd:complexType
name="replyType"> <xsd:simpleContent> <xsd:extension base="xsd:string"> <xsd:attribute
name="QMGR" type="xsd:string" use="required"/> <xsd:attribute name="persistent"
type="persistenceType" use="optional"/> <xsd:attribute name="detailed" type="detailedType"
use="optional" /> </xsd:extension> </xsd:simpleContent> </xsd:complexType> <!-- Defines the
accepted choices for the detailed attribute. --> <xsd:simpleType name="detailedType"> <xsd:restriction
base="xsd:token"> <xsd:enumeration value="true"/> <xsd:enumeration value="false"/> </xsd:restriction>
</xsd:simpleType> <!-- Defines the priority type <transferset priority="1"> --> <xsd:simpleType
```

name="priorityType"> <xsd:restriction base="xsd:string"> <xsd:pattern value="[0123456789]"/>
</xsd:restriction> </xsd:simpleType> <!-- Define the job information element <job>
<name>JOBNAME</name> </job> --> <xsd:complexType name="jobType"> <xsd:sequence>
<xsd:element name="name" type="xsd:string"/> </xsd:sequence> </xsd:complexType> <!-- Defines an
action <action> <runCommand name="myCommand.sh" /> </action> --> <xsd:complexType
name="commandActionType"> <xsd:choice> <xsd:element name="command" type="commandType"
maxOccurs="1" minOccurs="0"/> </xsd:choice> </xsd:complexType> <!-- Defines a command
<command name="runme" successRC="0" maxReplyLength="1024"> <argument>firstArg</argument>
<argument>secondArg</argument> </command> --> <xsd:complexType name="commandType">
<xsd:sequence> <xsd:element name="argument" type="xsd:string" maxOccurs="unbounded"
minOccurs="0"/> <xsd:element name="target" type="xsd:string" maxOccurs="unbounded"
minOccurs="0"/> <xsd:element name="property" type="propertyType" maxOccurs="unbounded"
minOccurs="0"/> </xsd:sequence> <xsd:attribute name="name" type="xsd:string" use="required"/>
<xsd:attribute name="successRC" type="xsd:string" use="optional"/> <xsd:attribute name="retryCount"
type="nonNegativeIntegerType" use="optional"/> <xsd:attribute name="retryWait"
type="nonNegativeIntegerType" use="optional"/> <xsd:attribute name="type" type="callTypeType"
use="optional"/> <xsd:attribute name="priority" type="commandPriorityType" use="optional"/>
<xsd:attribute name="message" type="xsd:string" use="optional"/> </xsd:complexType> <!-- Defines the
enumeration values for the type of a command type="executable|antscript|jcl|os4690background" -->
<xsd:simpleType name="callTypeType"> <xsd:restriction base="xsd:token"> <xsd:enumeration
value="executable"/> <xsd:enumeration value="antscript"/> <xsd:enumeration value="jcl"/>
<xsd:enumeration value="os4690background"/> </xsd:restriction> </xsd:simpleType> <!-- Defines the
priority type for a command priority="5" --> <xsd:simpleType name="commandPriorityType">
<xsd:restriction base="xsd:string"> <xsd:pattern value="[123456789]"/> </xsd:restriction>
</xsd:simpleType> <!-- Defines the property type that is used as a child of commandType <property
name="xxx" value="yyy"/> --> <xsd:complexType name="propertyType"> <xsd:attribute name="name"
type="xsd:string" use="required"/> <xsd:attribute name="value" type="xsd:string" use="required"/>
</xsd:complexType> <!-- Defines a non-negative integer type --> <xsd:simpleType
name="nonNegativeIntegerType"> <xsd:restriction base="xsd:integer"> <xsd:minInclusive value="0"/>
</xsd:restriction> </xsd:simpleType> <!-- Defines the transfer command specific reply information, to be
included as part the general reply <transferReply> <preSourceData> <runCommandReply
resultCode="0"> <stdout> <line>the quick brown fox jumped over the lazy dog</line> </stdout>
<stderr></stderr> </runCommandReply> </preSourceData> </transferReply> --> <xsd:complexType
name="transferReplyType"> <xsd:sequence> <xsd:element name="preSourceData" type="actionReplyType"
minOccurs="0" maxOccurs="1"/> <xsd:element name="postSourceData" type="actionReplyType"
minOccurs="0" maxOccurs="1"/> <xsd:element name="preDestinationData" type="actionReplyType"
minOccurs="0" maxOccurs="1"/> <xsd:element name="postDestinationData" type="actionReplyType"
minOccurs="0" maxOccurs="1"/> </xsd:sequence> </xsd:complexType> <!-- Define the action reply type
information <actionReply> <runCommandReply resultCode="1"> <stdout></stdout> <stderr>
<line>permission denied</line> </stderr> </runCommandReply> </actionReply> -->
<xsd:complexType name="actionReplyType"> <xsd:choice> <xsd:element name="runCommandReply"
type="commandReplyType" maxOccurs="1" minOccurs="0"/> </xsd:choice> </xsd:complexType> <!--
Defines command specific reply information, to be included as part the general reply <commandReply
resultCode="0"> <stdout> <line>first line of output text</line> <line>second line of output text</line>
</stdout> <stderr> <line>line of error text</line> </stderr> </commandReply> --> <xsd:complexType
name="commandReplyType"> <xsd:sequence> <xsd:element name="stdout" type="textLinesType"
maxOccurs="1" minOccurs="1"/> <xsd:element name="stderr" type="textLinesType" maxOccurs="1"
minOccurs="1"/> </xsd:sequence> <xsd:attribute name="resultCode" type="xsd:int" use="required"/>
</xsd:complexType> <!-- Defines type for lines of text --> <xsd:complexType name="textLinesType">
<xsd:sequence> <xsd:element name="line" type="xsd:string" maxOccurs="unbounded" minOccurs="0"/>
</xsd:sequence> </xsd:complexType> <!-- Defines the ping agent command specific reply information,
to be included as part the general reply <pingAgentReply resultCode="0"> <agentVersion>Build level:
f000-20090408-1200</agentVersion> </pingAgentReply> --> <xsd:complexType
name="pingAgentReplyType"> <xsd:sequence> <xsd:element name="agentVersion" type="xsd:string"
maxOccurs="1" minOccurs="0"/> </xsd:sequence> </xsd:complexType> <!-- Defines sequence of exit
elements <exit ... <exit ... --> <xsd:complexType name="exitGroupType"> <xsd:sequence> <xsd:element

name="exit" type="exitType" minOccurs="1" maxOccurs="unbounded"/> </xsd:sequence>
</xsd:complexType> <!-- Defines the outcome of calling a command <command ... <callResult ... -->
<xsd:complexType name="callGroupType"> <xsd:sequence> <xsd:element name="command"
type="commandType" minOccurs="1" maxOccurs="1"/> <xsd:element name="callResult"
type="callResultType" minOccurs="0" maxOccurs="1"/> </xsd:sequence> </xsd:complexType> <!--
Defines eiter the successful call of a command, or a failed attempt to call a commad <callResultType
outcome="success|failure|error" retries="X"> <result ... /> </callResultType> --> <xsd:complexType
name="callResultType"> <xsd:sequence> <xsd:element name="result" type="resultType" minOccurs="1"
maxOccurs="unbounded"/> </xsd:sequence> <xsd:attribute name="returnCode" type="xsd:integer"
use="optional"/> <xsd:attribute name="retries" type="xsd:integer" use="optional"/> <xsd:attribute
name="outcome" type="outcomeType" use="required"/> </xsd:complexType> <!-- Defines the
information recorded for the successfull call of a command <result...> <stdout... <stderr... <error...
</result...> --> <xsd:complexType name="resultType"> <xsd:sequence> <xsd:element name="stdout"
type="outputType" minOccurs="0" maxOccurs="1"/> <xsd:element name="stderr" type="outputType"
minOccurs="0" maxOccurs="1"/> <xsd:element name="error" type="xsd:string" maxOccurs="1"
minOccurs="0"/> </xsd:sequence> <xsd:attribute name="returnCode" type="xsd:integer"
use="optional"/> <xsd:attribute name="outcome" type="outcomeType" use="required"/> <xsd:attribute
name="time" type="xsd:dateTime" use="required"/> </xsd:complexType> <!-- Enumeration of call
outcomes - success, failure or error --> <xsd:simpleType name="outcomeType"> <xsd:restriction
base="xsd:token"> <xsd:enumeration value="success"/> <xsd:enumeration value="failure"/>
<xsd:enumeration value="error"/> </xsd:restriction> </xsd:simpleType> <!-- Defines the information
recorded for each line of standard output / standard error generated by calling a program <line>line
1</line> <line>line 2</line> etc. --> <xsd:complexType name="outputType"> <xsd:sequence>
<xsd:element name="line" type="xsd:string" maxOccurs="unbounded" minOccurs="1"/> </xsd:sequence>
</xsd:complexType> <!-- Defines the information recorded for an unsuccessful program call. -->
<xsd:complexType name="callFailedType"> <xsd:simpleContent> <xsd:extension base="xsd:string"/>
</xsd:simpleContent> </xsd:complexType> <!-- Defines the exit type; records the transfer exit class
name and a status message <exit name="class com.example.exit.StartExit"> <status ... </exit> -->
<xsd:complexType name="exitType"> <xsd:sequence> <xsd:element name="status" type="exitStatusType"
minOccurs="1" maxOccurs="1"/> </xsd:sequence> <xsd:attribute name="name" type="xsd:string"
use="required"/> </xsd:complexType> <!-- Defines exit status to record whether exit voted to proceed or
cancel transfer. <status resultCode="proceed"> <supplement>go ahead</supplement> </status> -->
<xsd:complexType name="exitStatusType"> <xsd:sequence> <xsd:element name="supplement"
type="xsd:string" maxOccurs="unbounded" minOccurs="0"/> </xsd:sequence> <xsd:attribute
name="resultCode" type="exitResultEnumType" use="optional"/> </xsd:complexType> <!-- Defines the
enumeration for transfer exit result values. <status resultCode="proceed"> --> <xsd:simpleType
name="exitResultEnumType"> <xsd:restriction base="xsd:token"> <xsd:enumeration value="proceed"/>
<xsd:enumeration value="cancelTransfer"/> <xsd:enumeration value="cancelTask"/> </xsd:restriction>
</xsd:simpleType> </xsd:schema>

Related concepts:

"Message formats for WebSphere MQ Managed File Transfer" on page 750
WebSphere MQ Managed File Transfer uses messages in XML format for a number of purposes: to
interact with the Web Gateway; to command an agent; to log information about the monitors, schedules,
and transfers; and to define information used for configuration. The logical structure of the XML formats
used for these purposes described by XML schema.

Developing applications

Running programs before or after a transfer

Examples of using fteCreateTransfer to start programs
You can use the **fteCreateTransfer** command to specify programs to run before or after a transfer.

In addition to using **fteCreateTransfer**, there are other ways to invoke a program before or after a
transfer. For more information, see "Specifying programs to run" on page 277.

All these examples use the following syntax to specify a program:

`[`*type*`:]`*commandspec*`[,[`*retrycount*`][,[`*retrywait*`][,`*successrc*`]]]`

For more information about this syntax, see "**fteCreateTransfer** (create new file transfer)" on page 461.

Running an executable program

The following example specifies an executable program called `mycommand` and passes two arguments, a and b, to the program.

`mycommand(a,b)`

To run this program at the source agent AGENT1 before the transfer starts, use the following command:

```
fteCreateTransfer -sa AGENT1 -da AGENT2 -presrc mycommand(a,b)
destinationSpecification sourceSpecification
```

Running, and retrying, an executable program

The following example specifies an executable program called `simple`, which does not take any arguments. A value of 1 is specified for `retrycount` and a value of 5 is specified for `retrywait`. These values mean that the program will be retried once if it does not return a successful return code, after a wait of five seconds. No value is specified for `successrc`, so the only successful return code is the default value of 0.

`executable:simple,1,5`

To run this program at the source agent AGENT1 after the transfer has completed, use the following command:

```
fteCreateTransfer -sa AGENT1 -da AGENT2 -postsrc executable:simple,1,5
destinationSpecification sourceSpecification
```

Running an IBM 4690 executable program in the background

The following example specifies an executable program called `mycmd.bat`. A value of 9 is specified for `priority` and a value of `message123` is specified for `message`. These values mean that the program will be given the highest priority, and `message123` is displayed on the IBM 4690 system background control screen for the command. No value is specified for `successrc`, so the only successful return code is the default value of 0.

`os4690background:mycmd.bat(arg1,arg2),,,9,message123`

To run this program at the source agent AGENT1 before the transfer starts, use the following command:

`fteCreateTransfer -sa AGENT1 —presrc os4690background:mycmd.bat(arg1,arg2),,,9,message123`

Running an Ant script and specifying successful return codes

The following example specifies an Ant script called `myscript` and passes two properties to the script. The script is run using the **fteAnt** command. The value for `successrc` is specified as >2&<7&!5|0|14, which specifies that return codes of 0, 3, 4, 6, and 14 indicate success.

`antscript:myscript(prop1=fred,prop2=bob),,,>2&<7&!5|0|14`

To run this program at the destination agent AGENT2 before the transfer has started, use the following command:

```
fteCreateTransfer -sa AGENT1 -da AGENT2 -predst
"antscript:myscript(prop1=fred,prop2=bob),,,>2&<7&!5|0|14" destinationSpecification sourceSpecification
```

Running an Ant script and specifying targets to call

The following example specifies an Ant script called `script2` and two targets, `target1` and `target2`, to call. The property `prop1` is also passed in, with a value of `recmfm(F,B)`. The comma (,) and parentheses in this value are escaped using a backslash character (\).

```
antscript:script2(target1,target2,prop1=recmfm\(F\,B\)),,,>2&<7&!5|0|14
```

To run this program at the destination agent AGENT2 after the transfer has completed, use the following command:

```
fteCreateTransfer -sa AGENT1 -da AGENT2
-postdst "antscript:script2(target1,target2,prop1=recmfm\(F\,B\)),,,>2&<7&!5|0|14"
destinationSpecification sourceSpecification
```

Running a JCL script

The following example specifies a JCL script called `zosbatch`. A value of 3 is specified for `retrycount`, a value of 30 is specified for `retrywait` and a value of 0 is specified for `successrc`. These values mean that the script will be retried three times if it does not return a successful return code of 0, with a wait of thirty seconds between each attempt.

```
jcl:zosbatch,3,30,0
```

To run this program at the source agent AGENT1 after the transfer has completed, use the following command:

```
fteCreateTransfer -sa AGENT1 -da AGENT2 -postsrc jcl:zosbatch,3,30,0
destinationSpecification sourceSpecification
```

Related concepts:

"Specifying programs to run" on page 277
You can run programs on a system where a IBM WebSphere MQ Managed File Transfer agent is running. As part of a file transfer request, you can specify a program to run either before a transfer starts, or after it finishes. Additionally, you can start a program that is not part of a file transfer request by submitting a managed call request.

Related reference:

"**fteCreateTransfer** (create new file transfer)" on page 461
The **fteCreateTransfer** command creates and starts a new file transfer from the command line. This command can start a file transfer immediately, schedule a file transfer for a future time and date, repeat a scheduled transfer one or more times, and trigger a file transfer based on certain conditions.

Working with the Web Gateway

Web Gateway API reference
The WebSphere MQ Managed File Transfer Web Gateway defines a RESTful HTTP application programming interface (API) for creating transfers, downloading files from file spaces and viewing the status of submitted transfers using HTTP requests and responses.

This reference topic describes the API for user actions. For administrative actions such as creating and deleting file spaces, see "Web Gateway administration API reference" on page 885.

Resource types

The following WebSphere MQ Managed File Transfer object types are supported by this specification:

File   A file transferred to or from a WebSphere MQ Managed File Transfer agent.

Filespace
> A logical area containing files that have been sent to the user or group associated with that file space.

Transfer
An instance of a WebSphere MQ Managed File Transfer transfer.

HTTP verbs

The HTTP verbs in the following table are supported by this specification.

HTTP verb	WebSphere MQ Managed File Transfer operations
POST	• Upload a file or files to a destination agent
GET	• Retrieve the status of a previous transfer • Retrieve a list of files in a file space • Download a file from a file space
DELETE	• Delete, and optionally download, a file from a file space

HTTP headers and HTML form fields for using the Web Gateway:

You can customize a request to create or retrieve a resource by using HTTP headers or HTML form fields. Each parameter maps to a property or function of WebSphere MQ Managed File Transfer.

The HTTP convention is to preface custom headers with `x-` followed by a product-specific identifier. WebSphere MQ Managed File Transfer uses the product identifier `fte-`.

For the possible values of each header or form field listed in the following table, see the topic that describes the equivalent WebSphere MQ Managed File Transfer command and parameter. For example, the set of possible values for the `x-fte-action` header is the set of possible values for the fteCreateTransfer command when used with the `-de` parameter.

Header name	Form field name	Supported object types and verbs	Command line command and parameter	Description	Example usage of header	Example usage of form field
x-fte-action	action	File (POST)	fteCreateTransfer, -de	Specifies the action to take if the destination file exists. Valid options are: • overwrite • error - This is the default.	x-fte-action:overwrite	`<input type="HIDDEN" name="action" value="overwrite"/>`
x-fte-priority	priority	File (POST)	fteCreateTransfer, -pr	Specifies the priority of the transfer.	x-fte-priority:5	`<input type="HIDDEN" name="priority" value="5"/>`
x-fte-type	type	File (POST)	fteCreateTransfer, -t	Specifies whether the transfer is in binary or text mode. Valid options are: • text • binary - This is the default.	x-fte-type:binary	`<input type="HIDDEN" name="type" value="binary"/>`
x-fte-checksum	checksum	File (POST)	fteCreateTransfer, -cs	Specifies the checksum algorithm to use to check that the transfer sent the data correctly. Valid options are: • none • MD5 - This is the default.	x-fte-checksum:MD5	`<input type="HIDDEN" name="checksum" value="MD5"/>`
x-fte-metadata	metadata	File (POST)	fteCreateTransfer, -md	Specifies metadata to associate with the transfer. The metadata header or form field can be specified multiple times within a single request.	x-fte-metadata:a=b,c=d,e=f	`<input type="HIDDEN" name="metadata" value="a=b,c=d,e=f"/>`

Header name	Form field name	Supp orted object types and verbs	Command line command and parameter	Description	Example usage of header	Example usage of form field
x-fte-jobname	jobname	File (POST)	fteCreateTransfer, -jn	Specifies to job name to associate with the transfer.	x-fte-jobname:BatchOrder_1	`<input type="HIDDEN" name="jobname" value="BatchOrder_1"/>`
x-fte-postdest	postdest	File (POST)	fteAnt, where the ant script that is called contains a nested postdst element	Specifies a command to execute at the destination agent when the file transfer has completed. Supports all attributes available to a program invocation. For more details of these attributes, see the topic "Program invocation nested elements" on page 917	x-fte-postdest: [command=virus_scan.sh, type=executable, successrc=0]	`<input type="HIDDEN" name="postdest" value="[command= virus_scan.sh, type=executable, successrc=0]"/>`
x-fte-postdest-args	postdest-args	File (POST)	fteAnt, where the Ant script that is called specifies **arg** elements on a postdst element	Specifies one or more arguments to pass to a command if a command has been specified using the x-fte-postdest header. Only valid if the type attribute specified in the x-fte-postdest header is executable. Supports all attributes available to a program invocation. For more details of these attributes, see the topic "Program invocation nested elements" on page 917.	x-fte-postdest-args:[argument1, argument2]	`<input type="HIDDEN" name="postdest-args" value="[argument1, argument2]"/>`
x-fte-postdest-properties	postdest-properties	File (POST)	fteAnt, where the Ant script that is called specifies **property** elements on a postdst element	Specifies one or more properties to pass to an Ant script if an Ant script has been specified using the x-fte-postdest header. Only valid if the type attribute specified in the x-fte-postdest header is antscript. Supports all attributes available to a program invocation. For more details of these attributes, see the topic "Program invocation nested elements" on page 917.	x-fte-postdest-properties :[scanspeed=fast, resultoutput=scan.log]	`<input type="HIDDEN" name= "postdest-properties" value="[scanspeed=fast, resultoutput= scan.log]"/>`
x-fte-postdest-targets	postdest-targets	File (POST)	fteAnt, where the Ant script that is called specifies **target** elements on a postdst element	Specifies one or more targets to run from an Ant script if an Ant script has been specified using the x-fte-postdest header. Only valid if the type attribute specified in the x-fte-postdest header is antscript. For more details, see the topic "Program invocation nested elements" on page 917.	x-fte-postdest-targets:[scanfile]	`<input type="HIDDEN" name="postdest-targets" value="[scanfile]"/>`
x-fte-include-file-in-response	None	File (DELETE)	None	Specifies whether the deleted file is included in the HTTP response. The default is false.	x-fte-include-file-in-response:true Specifies whether the deleted file is included in the HTTP response. The default is false.	None

Header name	Form field name	Supported object types and verbs	Command line command and parameter	Description	Example usage of header	Example usage of form field
x-fte-check-integrity	None	File space (GET)	None	On a request to view the contents of a file space, specifies that an integrity check should be carried out on the file space files. On a request to list all file spaces, specifies that an integrity check should be carried out on the file space root. The default is false.	x-fte-check-integrity:true	None
V7.5.0.1 x-fte-csrf-token	csrf-token	File (POST, DELETE) File space (POST, DELETE)	None	Specifies the current JSESSIONID that the Web Gateway should compare against its own JSESSIONID on a request to create or delete a file, or to create, delete or modify a file space. This provides protection against cross-site request forgery (CSRF) attacks if CSRF protection is enabled on the Web Gateway.	x-fte-csrf-token: E324D3C64E6B4662 0843C88A25B4C32D	`<input type="HIDDEN" name="csrf-token" value="E324D3C64E6B4 6620843C88A25B4C32D"/>`

Uniform Resource Identifier syntax for using the Web Gateway:

A WebSphere MQ Managed File Transfer Uniform Resource Identifier (URI) is distinguished from other WebSphere MQ URIs by the context root specified at deploy time. The recommended context root is `/wmqfte`.

WebSphere MQ Managed File Transfer resources are distinguished from each other by their types. A resource is addressed by its resource type and an identifying token.

WMQFTE Uniform Resource Identifier syntax diagram
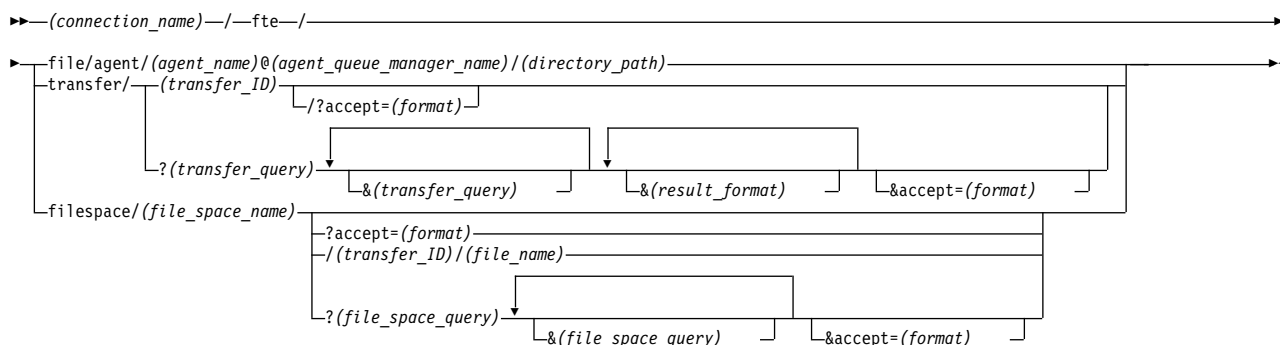
Parameters

(`connection_name`)
Required. The host name and, optionally, the port of the server hosting the WebSphere MQ Managed File Transfer Web Gateway. Not case-sensitive.

`fte`
Required. Prefix indicating that the URI is addressed to the WebSphere MQ Managed File Transfer Web Gateway. Case-sensitive.

`file`
Optional. Indicates that you are addressing a file resource. Case-sensitive.

agent
> Optional. Indicates that the type of destination is an agent. Case-sensitive.

(*agent_name*)
> Optional. The name of the agent to send the file to. Not case-sensitive, agent names are converted to uppercase.

(*agent_queue_manager_name*)
> Required. The name of the queue manager used by the agent to send the file to. Case-sensitive.

(*directory_path*)
> Optional. The path of the directory on the destination agent file system that you are addressing. The directory path must contain only unreserved or escaped characters. Case-sensitive.
>
> If the *directory_path* part of the URI begins with a forward slash (/) character, in addition to the forward slash character used as a path separator, the *directory_path* is resolved as an absolute path. If you want to upload a file to an absolute path, you must encode the forward slash as the string %2F so that it is not removed. If you do not want Web Gateway uploads to be able to write to an absolute path on the destination agent's file system, you must configure user or agent sandboxing on the destination agent.
>
> If the *directory_path* does not begin with an additional forward slash character, the directory path is resolved relative to the transfer root directory of the destination agent.

transfer
> Optional. Indicates that you are addressing a transfer resource. Case-sensitive.

(*transfer_ID*)
> Optional. The transfer ID is the unique 48 character hexadecimal string that identifies the transfer. Not case-sensitive.

accept=(*format*)
> Optional. Specifies the format of the response that the Web Gateway returns. The value of *format* is one of the following values:
> - **JSON** - Specifies that the response is in JavaScript Object Notation.
> - **XML** - Specifies that the response is in XML format. This is the default.
>
> Not case-sensitive. You can also set the format of the response using the `Accept:` header in the request. The format that is set using the URI takes priority over the format set using the `Accept:` header.

(*transfer_query*)
> Optional. Requests information about all transfers that match the query, from the WebSphere MQ Managed File Transfer Web Gateway. You can specify multiple queries, separated by the ampersand character (&), but only one of each type of query.
>
> The query can be one of the following types:
> - **srcagent**=(*agent_name*)
> - **destagent**=(*agent_name*)
> - **agent**=(*agent_name*)
> - **status**=(*status_value*)
> - **metadata**=(*metadata_info*)
> - **endafter**=(*date*)
> - **endbefore**=(*date*)
> - **startafter**=(*date*)
> - **startbefore**=(*date*)
> - **srcfile**=(*file_path*)
> - **destfile**=(*file_path*)

- **jobname**=(*job_name*)
- **returncode**=(*return_code*)

For more information about these queries, see "Query parameters."

filespace
Optional. Indicates that you are addressing a file space resource. Case-sensitive.

(*file_space_name***)**
Optional. The name of the file space you are addressing. This is the name of the user associated with the file space. Case-sensitive.

(*file_name***)**
Optional. The name of the file to download. If a file name has a space character in the name this character must be represented by the string %20 in the URI. Case-sensitive.

(*file_space_query***)**
Optional. Requests information about all files in the file space that match the query, from the WebSphere MQ Managed File Transfer Web Gateway. You can specify multiple queries, separated by the ampersand character (&), but only one of each type of query.

The query can be one of the following types:
- **endafter**=(*date*)
- **endbefore**=(*date*)
- **startafter**=(*date*)
- **startbefore**=(*date*)

For more information about these queries, see "Query parameters."

(*result_format***)**
- **sortby**=(*sort_by_values*)
- **sort**=(*sort_values*)
- **start**=(*start_value*)
- **count**=(*count_value*)

For more information about these result formats, see "Result format parameters" on page 874.

Query parameters

srcagent=**(***agent_name***)**

Requests information about transfers that have *agent_name* as the source agent. The value of *agent_name* is not case-sensitive, agent names are converted to uppercase.

If you use the **srcagent** query you cannot use the **agent** query.

destagent=**(***agent_name***)**

Requests information about transfers that have *agent_name* as the destination agent. The value of *agent_name* is not case-sensitive, agent names are converted to uppercase.

If you use the **destagent** query you cannot use the **agent** query.

agent=**(***agent_name***)**

Requests information about transfers that have *agent_name* as either the source agent, the destination agent or both. The value of *agent_name* is not case-sensitive, agent names are converted to uppercase.

If you use the **agent** query you cannot use the **srcagent** or **destagent** query.

status=**(***status_value***)**

Requests information about transfers that have *status_value* as their transfer status. The value of *status_value* is case-sensitive and is a comma-separated list enclosed in brackets ([ ]). The comma-separated list contains one or many of the following values:

- **submitted**
- **started**
- **success**
- **partial success**
- **cancelled**
- **failure**

metadata=**(***metadata_info***)**

Requests information about transfers that have *metadata_info* as part of their metadata.

The value of *metadata_info* is in one of the following formats:

name   The name part of a metadata name-value pair. If the transfer has metadata with this name and any value the transfer matches the query.

name=value
    A metadata name-value pair. If the transfer has metadata with this name and this value the transfer matches the query.

endafter=**(***date***)**

Requests information about transfers that completed after the date given by the *date* value. The value of *date* is in one of the following formats:

yyyy-MM-ddTHH:mm:ss
    The date and time. For example, `2010-08-26T12:25:40`.

yyyy-MM-ddTHH:mm
    The date and time, without seconds. For example, `2010-08-26T12:25`, which is evaluated as `2010-08-26T12:25:00`.

yyyy-MM-ddTHH
    The date and time, without seconds and minutes. For example, `2010-08-26T12`, which is evaluated as `2010-08-26T12:00:00`.

yyyy-MM-dd
    The date. For example, `2010-08-26`, which is evaluated as `2010-08-26T00:00:00`.

yyyy-MM
    The date without days. For example, `2010-08`, which is evaluated as `2010-07-31T23:59:59`.

yyyy   The year. For example, `2010`, which is evaluated as `2009-12-31T23:59:59`.

The date and time are in Coordinated Universal Time (UTC).

You can specify a date in a different timezone by adding a four-digit number, prefaced by a plus (+) sign or minus (-) sign, to the end of the date to indicate the difference in time between UTC and the timezone you are using. For example, to specify 7pm on the 26th August 2010 in the timezone for San Francisco, Pacific Daylight Time, which is 7 hours behind UTC, use the following value: `2010-08-26T19:00-0700`.

endbefore=**(***date***)**

Requests information about transfers that completed before the date given by the *date* value. The value of *date* is in one of the following formats:

yyyy-MM-ddTHH:mm:ss
    The date and time. For example, `2010-08-26T12:25:40`.

yyyy-MM-ddTHH:mm

The date and time, without seconds. For example, `2010-08-26T12:25`, which is evaluated as `2010-08-26T12:25:00`.

yyyy-MM-ddTHH

The date and time, without seconds and minutes. For example, `2010-08-26T12`, which is evaluated as `2010-08-26T12:00:00`.

yyyy-MM-dd

The date. For example, `2010-08-26`, which is evaluated as `2010-08-26T00:00:00`.

yyyy-MM

The date without days. For example, `2010-08`, which is evaluated as `2010-07-31T23:59:59`.

yyyy   The year. For example, `2010`, which is evaluated as `2009-12-31T23:59:59`.

The date and time are in Coordinated Universal Time (UTC).

You can specify a date in a different timezone by adding a four-digit number, prefaced by a plus (+) sign or minus (-) sign, to the end of the date to indicate the difference in time between UTC and the timezone you are using. For example, to specify 7pm on the 26th August 2010 in the timezone for San Francisco, Pacific Daylight Time, which is 7 hours behind UTC, use the following value: `2010-08-26T19:00-0700`.

startafter=(*date*)

Requests information about transfers that started after the date given by the *date* value. The value of *date* is in one of the following formats:

yyyy-MM-ddTHH:mm:ss

The date and time. For example, `2010-08-26T12:25:40`.

yyyy-MM-ddTHH:mm

The date and time, without seconds. For example, `2010-08-26T12:25`, which is evaluated as `2010-08-26T12:25:00`.

yyyy-MM-ddTHH

The date and time, without seconds and minutes. For example, `2010-08-26T12`, which is evaluated as `2010-08-26T12:00:00`.

yyyy-MM-dd

The date. For example, `2010-08-26`, which is evaluated as `2010-08-26T00:00:00`.

yyyy-MM

The date without days. For example, `2010-08`, which is evaluated as `2010-07-31T23:59:59`.

yyyy   The year. For example, `2010`, which is evaluated as `2009-12-31T23:59:59`.

The date and time are in Coordinated Universal Time (UTC).

You can specify a date in a different timezone by adding a four-digit number, prefaced by a plus (+) sign or minus (-) sign, to the end of the date to indicate the difference in time between UTC and the timezone you are using. For example, to specify 7pm on the 26th August 2010 in the timezone for San Francisco, Pacific Daylight Time, which is 7 hours behind UTC, use the following value: `2010-08-26T19:00-0700`.

startbefore=(*date*)

Requests information about transfers that started before the date given by the *date* value. The value of *date* is in one of the following formats:

yyyy-MM-ddTHH:mm:ss

The date and time. For example, `2010-08-26T12:25:40`.

yyyy-MM-ddTHH:mm

The date and time, without seconds. For example, `2010-08-26T12:25`, which is evaluated as `2010-08-26T12:25:00`.

yyyy-MM-ddTHH

The date and time, without seconds and minutes. For example, `2010-08-26T12`, which is evaluated as `2010-08-26T12:00:00`.

yyyy-MM-dd

The date. For example, `2010-08-26`, which is evaluated as `2010-08-26T00:00:00`.

yyyy-MM

The date without days. For example, `2010-08`, which is evaluated as `2010-07-31T23:59:59`.

yyyy   The year. For example, `2010`, which is evaluated as `2009-12-31T23:59:59`.

The date and time are in Coordinated Universal Time (UTC).

You can specify a date in a different timezone by adding *Z* to the end of the date in any of the listed formats. The value of *Z* is a four-digit number indicating the difference in time between UTC and the timezone you are using. For example, to specify 7pm on the 26th August 2010 in the timezone for San Francisco, Pacific Daylight Time, which is 7 hours behind UTC, use the following value: `2010-08-26T19:00-0700`.

srcfile=(*file_path***)**

Requests information about transfers that have *file_path* as the full source file path. Case-sensitive.

If a file path contains a space character, this character must be represented by the string %20 in the query.

destfile=(*file_path***)**

Requests information about transfers that have *file_path* as the full destination file path. Case-sensitive.

If a file path contains a space character, this character must be represented by the string %20 in the query.

jobname=(*job_name***)**
Requests information about transfers that have *job_name* as their job name. Job name is case-sensitive.

returncode=(*return_code***)**
Requests information about transfers that have *return_code* as their return code. The return code of a transfer is a positive integer. For a list of possible return codes, see "Return codes for WebSphere MQ Managed File Transfer" on page 373.

transferid=(*transfer_ID***)**
Optional. The transfer ID is the unique 48 character hexadecimal string that identifies the transfer that transferred the file to the file space. Not case-sensitive.

Result format parameters

sortby=(*sort_by_values***)**

Specifies which value to sort the results by. For a transfer query the value of *sort_by_value* is one of the following values:
- **srcagent**
- **destagent**
- **status**
- **startdate**
- **enddate**

- **jobname**

By default the results are sorted by **startdate**.

sort=**(***sort_value***)**

Specifies whether the results that are returned are sorted in ascending or descending order of the value specified for **sortby** query. The value of *sort_value* is one of the following values:

- **ascending**
- **descending**

You can only specify the **sort** query if you have specified the **sortby** query.

start=**(***start_value***)**

Specifies the index of the first result to return. The value of *start_value* is 0 or a positive integer. The first result found by the Web Gateway has an index of 0.

count=**(***count_value***)**

Specifies the number of results to return. The value of *count_value* is a positive integer that is less than 100. You can only return 100 results at a time.

Examples

For example, to use a POST request to transfer a file resource to a destination agent called ACCOUNTS, which uses an agent queue manager called DEPT1, use the following URI:

```
http://example.org/wmqfte/file/agent/ACCOUNTS@DEPT1/
```

In this example:
- `http://example.org` is the host system.
- `/wmqfte` indicates the URI is a WebSphere MQ Managed File Transfer URI.
- `/file` indicates that the resource being addressed is a file resource.
- `/agent/ACCOUNTS@DEPT1/` is the identifying token. This identifying token is a combination of the destination type, in this case `agent`, a destination agent name, in this case `ACCOUNTS`, and the destination agent queue manager name prefixed by an @ sign, in this case `@DEPT1`.

For example, to address a transfer resource:

```
http://example.org/wmqfte/transfer/414d5120514d5f4c4d343336303920201159c54820027102
```

In this example:
- `http://example.org` is the host system.
- `/wmqfte` indicates the URI is a WebSphere MQ Managed File Transfer URI.
- `/transfer` indicates that the resource being addressed is a transfer resource.
- `/414d5120514d5f4c4d343336303920201159c54820027102` is the identifying token, which in this case is the hexadecimal transfer ID.

Content types for using the Web Gateway:

File transfer requests that you submit to the WebSphere MQ Managed File Transfer Web Gateway SupportPac must correspond to certain media types. Responses from the Web Gateway have a media type of `application/xml` or `application/json`.

Request

Content transferred to WebSphere MQ Managed File Transfer using HTTP must be in one of the formats in the following table.

Table 54. The MQMFT resources and HTTP verbs that accept different media-types

Media-type	Valid MQMFT resources	Allowed verbs
multipart/form-data	File (transfers of multiple files or transfers with metadata)	POST, GET, DELETE
application/xml	Transfer, Filespace	POST, GET, DELETE

When you POST a file as part of a multipart request any media type can be used in each multipart boundary. The media type of the file determines whether the file transfer is in binary or text mode, unless the mode is overridden with the `x-fte-type` header.

Table 55. The transfer mode used by default for different media-types

Media-type	Transfer mode used
text/*	text
application/xml	binary
Any other media-type	binary

Response body

The Web Gateway can return a response with a media type of `application/xml` or `application/json` in response to both file upload requests (POST of a FILE resource) and transfer status requests (GET of a TRANSFER resource). For more information about JSON and XML response formats, see "Response formats: XML and JSON." The Web Gateway can return a response with any media type in response to a file download request (GET of a FILESPACE resource).

Response formats: XML and JSON:

The WebSphere MQ Managed File Transfer Web Gateway returns responses to queries in one of two formats: XML or JavaScript Object Notation (JSON).

You can specify the format of the response from the Web Gateway by including the `Accept: return-type` header in the request or by including the query `accept=return-type` in the URI. You can use a web application to parse the content of the XML or JSON response and display it in an appropriate format to a web user.

The default format is XML. If you specify the format using both the `Accept:` header and the query `accept=` in the URI, the Web Gateway returns a response in the format specified by the query in the URI.

Transfer query response formats:

When you request the status of a transfer or multiple transfers from the WebSphere MQ Managed File Transfer Web Gateway the response is returned in either JSON or XML format.

XML

The following example shows the format of a simple transfer query XML response.

```
<transfers xsi:noNamespaceSchemaLocation="WebTransferStatus.xsd">
  <transfer end-time="2010-08-26T12:00:00.260Z"
            start-time="2010-08-26T11:55:00.076Z"
            status="Success"
            id="414d51205745422e4654452020202020c1a1a34b03720120">
    <source>
      <agent qmgr="QM_JUPITER" name="AGENT_CALLISTO"/>
      <metadata>
        <key value="FIRST_JOB" name="com.ibm.wmqfte.JobName"/>
        <key value="AGENT_CALLISTO" name="com.ibm.wmqfte.SourceAgent"/>
        <key value="AGENT_EUROPA" name="com.ibm.wmqfte.DestinationAgent"/>
        <key value="serenity.example.com."
             name="com.ibm.wmqfte.OriginatingHost"/>
        <key value="user1" name="com.ibm.wmqfte.MqmdUser"/>
        <key value="414d51205745422e4654452020202020c1a1a34b03720120"
             name="com.ibm.wmqfte.TransferId"/>
        <key value="user1" name="com.ibm.wmqfte.OriginatingUser"/>
      </metadata>
    </source>
    <destination>
      <agent qmgr="QM_JUPITER" name="AGENT_EUROPA"/>
      <metadata>
        <key value="FIRST_JOB" name="com.ibm.wmqfte.JobName"/>
        <key value="AGENT_CALLISTO" name="com.ibm.wmqfte.SourceAgent"/>
        <key value="AGENT_EUROPA" name="com.ibm.wmqfte.DestinationAgent"/>
        <key value="user1" name="com.ibm.wmqfte.MqmdUser"/>
        <key value="serenity.example.com."
             name="com.ibm.wmqfte.OriginatingHost"/>
        <key value="user1" name="com.ibm.wmqfte.OriginatingUser"/>
        <key value="414d51205745422e4654452020202020c1a1a34b03720120"
             name="com.ibm.wmqfte.TransferId"/>
      </metadata>
    </destination>
    <stats retry-count="0" file-warnings="0" file-failures="0"
           bytes-transferred="259354303"/>
    <result text="BFGRP0032I: The file transfer request has successfully completed."
            code="0"/>
    <transfer-set>
      <file result-code="0" mode="text">
        <source-file name="/home/user1/output.zip">
          <attribute-values last-modified="2010-08-19T14:16:57.000Z"
                            file-size="259354303" disposition="leave"
                            checksum-value="98611a272a27d373f92d73a08cf0d4f4"
                            checksum-method="MD5"/>
        </source-file>
        <destination-file name="/tmp/output.zip">
          <attribute-values last-modified="2010-08-26T12:00:00.000Z"
                            file-size="259354303" exists-action="error"
                            checksum-value="98611a272a27d373f92d73a08cf0d4f4"
                            checksum-method="MD5"/>
        </destination-file>
      </file>
    </transfer-set>
  </transfer>
</transfers>
```

JSON

The following example shows the format of a simple transfer query JSON response.

```
{

    "transfers" : {
        "transfer" : {
            "end-time" : "2010-08-26T12:00:00.260Z",
            "status" : "Success",
```

```
         "start-time" : "2010-08-26T11:55:00.076Z",
         "id" : "414d51205745422e4654452020202020c1a1a34b03720120",
         "result" : {
            "code" : "0",
            "text" : "BFGRP0032I: The file transfer request has successfully completed."
         }
         ,
         "destination" : {
            "metadata" : {
               "key" : [
                  {
                     "name" : "com.ibm.wmqfte.JobName",
                     "value" : "FIRST_JOB"
                  }
                  ,
                  {
                     "name" : "com.ibm.wmqfte.SourceAgent",
                     "value" : "AGENT_CALLISTO"
                  }
                  ,
                  {
                     "name" : "com.ibm.wmqfte.DestinationAgent",
                     "value" : "AGENT_EUROPA"
                  }
                  ,
                  {
                     "name" : "com.ibm.wmqfte.MqmdUser",
                     "value" : "user1"
                  }
                  ,
                  {
                     "name" : "com.ibm.wmqfte.OriginatingHost",
                     "value" : "serenity.example.com."
                  }
                  ,
                  {
                     "name" : "com.ibm.wmqfte.OriginatingUser",
                     "value" : "user1"
                  }
                  ,
                  {
                     "name" : "com.ibm.wmqfte.TransferId",
                     "value" : "414d51205745422e4654452020202020c1a1a34b03720120"
                  }
               ]
            }
            ,
            "agent" : {
               "name" : "AGENT_EUROPA",
               "qmgr" : "QM_JUPITER"
            }
         }
      }
      ,
      "stats" : {
         "bytes-transferred" : "259354303",
         "retry-count" : "0",
         "file-warnings" : "0",
         "file-failures" : "0"
      }
      ,
      "transfer-set" : {
         "file" : {
            "result-code" : "0",
            "mode" : "text",
            "source-file" : {
               "name" : "\/home\/user1\/output.zip",
               "attribute-values" : {
                  "last-modified" : "2010-08-19T14:16:57.000Z",
                  "checksum-value" : "98611a272a27d373f92d73a08cf0d4f4",
                  "checksum-method" : "MD5",
                  "file-size" : "259354303",
                  "disposition" : "leave"
               }
            }
            ,
            "destination-file" : {
               "name" : "\/tmp\/output.zip",
               "attribute-values" : {
                  "exists-action" : "error",
```

```
                    "last-modified" : "2010-08-26T12:00:00.000Z",
                    "checksum-value" : "98611a272a27d373f92d73a08cf0d4f4",
                    "checksum-method" : "MD5",
                    "file-size" : "259354303"
                }
            }
        }
    }
    ,
    "source" : {
        "metadata" : {
            "key" : [
                {
                    "name" : "com.ibm.wmqfte.JobName",
                    "value" : "FIRST_JOB"
                }
                ,
                {
                    "name" : "com.ibm.wmqfte.SourceAgent",
                    "value" : "AGENT_CALLISTO"
                }
                ,
                {
                    "name" : "com.ibm.wmqfte.DestinationAgent",
                    "value" : "AGENT_EUROPA"
                }
                ,
                {
                    "name" : "com.ibm.wmqfte.OriginatingHost",
                    "value" : "serenity,example.com."
                }
                ,
                {
                    "name" : "com.ibm.wmqfte.MqmdUser",
                    "value" : "user1"
                }
                ,
                {
                    "name" : "com.ibm.wmqfte.TransferId",
                    "value" : "414d51205745422e4654452020202020c1a1a34b03720120"
                }
                ,
                {
                    "name" : "com.ibm.wmqfte.OriginatingUser",
                    "value" : "user1"
                }
            ]
        }
    }
    ,
    "agent" : {
        "name" : "AGENT_CALLISTO",
        "qmgr" : "QM_JUPITER"
    }
        }
    }
    }
}
```

Understanding the transfer query response

The names of the elements and attributes in the XML response format and the names of the objects in the JSON response format are the same. These elements, attributes, and objects are described in the following list:

transfers

> Group containing transfer information for all of the transfers that match the query.

transfer

> Group containing the information for a single transfer.

Attribute or object	Description
end-time	The time that the transfer finished in Coordinated Universal Time.

Attribute or object	Description
start-time	The time that the transfer started in Coordinated Universal Time.
status	The status of the transfer.
id	The unique hexadecimal ID of the transfer.

source Group containing information about the source of the transfer.

destination
Group containing information about the destination of the transfer.

agent Specifies the name and queue manager of an agent. When used within the element or object **source** this element or object specifies the source agent information; when used within the element or object **destination** this element or object specifies the destination agent information.

Attribute or object	Description
name	The name of the agent.
qmgr	The queue manager that the agent connects to.

metadata
Group containing transfer information in name-value pairs.

key Specifies a name-value pair.

Attribute or object	Description
name	The identifier of a piece of metadata.
value	The value of a piece of metadata.

stats Specifies information about the whole transfer.

Attribute or object	Description
retry-count	The number of times that the transfer went into recovery and was tried again by the agent.
file-warnings	The number of files in the transfer set that generated warnings while being transferred, but otherwise transferred successfully.
file-failures	The number of files in the transfer set that failed to transfer successfully.
bytes-transferred	The number of bytes transferred in this transfer.

result Specifies the return code and supplementary information of the transfer.

Attribute or object	Description
code	The return code of the transfer. For more information, see "Return codes for WebSphere MQ Managed File Transfer" on page 373.
text	The supplementary information of a transfer.

transfer-set
Group containing information about the files that were transferred.

file Group containing information about one file in the transfer.

Attribute or object	Description
result-code	The return code of the transfer of the individual file. For more information, see "Return codes for files in a transfer" on page 378.
mode	The transfer mode. Valid values are: • text • binary

source-file
>Specifies the name of the source file.

Attribute or object	Description
name	The name of the file on the source system.

destination-file
>Specifies the name of the destination file.

Attribute or object	Description
name	The name of the file on the destination system.

attribute-values
>Specifies additional information about the file being transferred. When used within the element or object **source-file** this element or object specifies information about the file on the source system; when used within the element or object **destination-file** this element or object specifies information about the file on the destination system.

Attribute or object	Description
file-size	The size of the file.
exists-action	Specifies what to do if the destination file already exists. Valid values are: • error • overwrite This attribute is valid only when the attribute-values element or object is used within the destination-file element or object.
disposition	Specifies what to do with the source file after the transfer is complete. Valid values are: • delete • leave This attribute is valid only when the attribute-values element or object is used within the source-file element or object.
checksum-method	The method used to produce a checksum value of this file.
checksum-value	The checksum value of the file.
last-modified	The time when the file was last modified, in Coordinated Universal Time.

File space query response formats:

When you request a list of some or all of the files in a file space from the WebSphere MQ Managed File Transfer Service Web Gateway the response is returned in either JSON or XML format, depending on what you have specified using the `Accept:` header.

XML

The following example shows the format of a simple file space query XML response.

```
<fileSpaces xsi:noNamespaceSchemaLocation="WebFileSpaceList.xsd">
  <fileSpace size="1" name="james">
    <file fileLink="/wmqfte/filespace/james/414d51205745422e4654452020202020c1a1a34b03720120/file.zip"
          transferLink="/wmqfte/transfer/414d51205745422e4654452020202020c1a1a34b03720120"
          transferID="414d51205745422e4654452020202020c1a1a34b03720120"
          name="/tmp/ae55bc7">
      <attribute-values mode="text" time="2010-08-26T19:00:02.000Z"
                        file-size="259354303"
                        checksum-value="98611a272a27d373f92d73a08cf0d4f4"
                        checksum-method="none"/>
    </file>
  </fileSpace>
</fileSpaces>
```

The XML response conforms to the schema `WebFileSpaceList.xsd`, which is located in the *MQ_INSTALLATION_PATH*/mqft/samples/schema directory of your MQMFT installation.

JSON

The following example shows the format of a simple file space query JSON response.

```
{
    "fileSpaces" : {
        "fileSpace" : {
            "name" : "james",
            "size" : "1",
            "file" : {
                "transferLink" : "\/wmqfte\/transfer\/414d51205745422e4654452020202020c1a1a34b03720120",
                "fileLink" : "\/wmqfte\/filespace\/1234\/414d51205745422e4654452020202020c1a1a34b03720120\/file.zip",
                "name" : "\/tmp\/ae55bc7",
                "transferID" : "414d51205745422e4654452020202020c1a1a34b03720120",
                "attribute-values" : {
                    "checksum-value" : "98611a272a27d373f92d73a08cf0d4f4",
                    "checksum-method" : "none",
                    "time" : "2010-08-26T19:00:02.000Z",
                    "file-size" : "259354303",
                    "mode" : "text"
                }
            }
        }
    }
}
```

Understanding the file space query response

The names of the elements and attributes in the XML response format and the names of the objects in the JSON response format are the same. These elements, attributes, and objects are described in the following list:

filespaces
  Group containing file space information.

filespace
  Group containing the information for a single file space.

Attribute or object	Description
size	The number of files in the file space returned by the query.

Attribute or object	Description
name	The name of the file space.

file Group containing the file information.

Attribute or object	Description
fileLink	Part of the URI used to download the file from the file space. The full URI for downloading the file is `host-name/fileLink`
transferLink	Part of the URI used to view the transfer information of the transfer that put the file in the file space. The full URI for viewing the transfer information is `host-name/transferLink`
transferID	The unique hexadecimal ID of the transfer that put the file in the file space.
name	The file path of the file on the system that hosts the file space.

attribute-values
Specifies additional information about the file being transferred.

Attribute or object	Description
file-size	The size of the file.
mode	The mode of the transfer. Valid values are: • text • binary
checksum-method	The method used to produce a checksum value of this file.
checksum-value	The checksum value of the file.
time	The time when the file was transferred to the file space, in Coordinated Universal Time.
integrity-check-result	The result of an integrity check on the file. Valid values are: • OK • MISSING_FILESYSTEM • MISSING_DATABASEENTRY

HTTP response codes:

Status codes are returned in HTTP responses to requests made to the .

The header of a response returned by the Web Gateway contains an HTTP response code. The HTTP header in the following example contains the HTTP response code 200 OK:

```
HTTP/1.1 200 OK
Server: WAS/6.0
Content-length: 0
```

The following table describes the possible values for the HTTP response code and an example of an associated WebSphere MQ Managed File Transfer error code that can be returned. For more information

about the WebSphere MQ Managed File Transfer error codes, see Diagnostic messages.

Table 56. HTTP response codes

HTTP response code	Example WebSphere MQ Managed File Transfer error code	Example description
200 OK	None	A valid request has been handled correctly and optionally a response has been provided to the user.
202 Accepted	None	A valid request has been handled correctly but WebSphere MQ Managed File Transfer does not guarantee that the requested action has completed. For example, a file upload transfer request has been handled and submitted to a WebSphere MQ Managed File Transfer agent but the transfer has not yet taken place.
400 Bad Request	BFGWI0001	The URI is not valid because it is missing a resource type.
403 Forbidden	BFGWI0056	There is no WebSphere MQ Message Descriptor (MQMD) user identifier defined for the user.
404 Not Found	BFGWI0015	The requested resource cannot be found.
405 Method Not Allowed	BFGWI0016	The requested resource does not support the HTTP verb that has been used in the request. For example, a GET has been used against a resource that only allows POST or DELETE.
410 Resource Gone	BFGWI0031	The requested resource is no longer available. For example, the requested file has been deleted from the file space.
413 Request Entity Too Large	BFGWI0026	The request contains a file that is too large to be handled by the server.
415 Unsupported Media Type	BFGWI0017	A request has been received with a media type, specified by the `Content-type` HTTP header, that is not supported.
500 Internal Server Error	BFGWI0018	An internal error has been encountered when handling the request. An FFDC or ABEND file has been produced.
502 Bad Gateway	BFGWI0019	The request cannot be completed because an error occurred outside WebSphere MQ Managed File Transfer. For example, a WebSphere MQ queue manager is not available.
503 Service Unavailable	BFGWI0020	The destination is temporarily unavailable. For example, a WebSphere MQ queue is full.

Table 56. HTTP response codes (continued)

HTTP response code	Example WebSphere MQ Managed File Transfer error code	Example description
504 Gateway Timeout	BFGWI0021	An attempt to complete the request has timed out because of time limits imposed by WebSphere MQ Managed File Transfer, or because of time limits imposed by the HTTP client.

Web Gateway administration API reference

The WebSphere MQ Managed File Transfer Web Gateway defines a RESTful HTTP application programming interface (API) for administration of file transfer artifacts.

This reference information describes the API for administering Web Gateway objects such as file spaces. For information on the API for non-administrative tasks, see "Web Gateway API reference" on page 866.

Resource types

The following WebSphere MQ Managed File Transfer object types are supported by this specification:

Filespace
> A logical area containing files that have been sent to the user or group associated with that file space.

User A set of mappings between web user ID and WebSphere MQ Message Descriptor (MQMD) user ID. These mappings control the MQMD user ID that is used for a file transfer request.

HTTP verbs

The HTTP verbs in the following table are supported by this specification.

HTTP verb	WebSphere MQ Managed File Transfer operations
POST	• Create an instance of a file space. • Modify the configuration of an existing file space. • Create a set of mappings between web user ID and MQMD user ID. • Modify or add to the set of mappings between web user ID and MQMD user ID.
GET	• View the current configuration of a file space. This configuration includes the file space name, the maximum size of the file space and the list of people who are authorized to write to the file space. • List all the file spaces that currently exist. • View the current set of mappings between web user ID and MQMD user ID.
DELETE	• Delete an instance of a file space. • Delete the set of mappings, or a subset of the mappings, between web user ID and MQMD user ID.

HTTP headers for administering the Web Gateway:

You can customize a request to create or retrieve a resource using HTTP headers. There are no headers defined for use with the administration API for the WebSphere MQ Managed File Transfer Web Gateway.

The HTTP convention is to preface custom headers with x- followed by a product-specific identifier. WebSphere MQ Managed File Transfer uses the product identifier fte-. For details of the headers that are supported by the Web Gateway API, see "HTTP headers and HTML form fields for using the Web Gateway" on page 867. There are no additional headers defined for administration purposes.

Uniform Resource Identifier syntax for administering the Web Gateway:

A WebSphere MQ Managed File Transfer Uniform Resource Identifier (URI) is distinguished from other WebSphere MQ URIs by the context root specified at deploy time. The recommended context root is /wmqfte. The URI used for administration tasks is distinguished from existing WebSphere MQ Managed File Transfer URIs by the term /admin.

WebSphere MQ Managed File Transfer resources are distinguished from each other by their types. A resource is addressed by its resource type and an identifying token.

WebSphere MQ Managed File Transfer Administration Uniform Resource Identifier syntax diagram
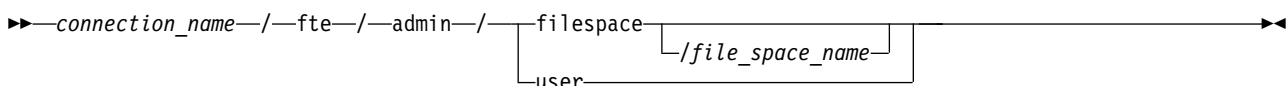
```
►►──connection_name──/──fte──/──admin──/──┬──filespace──────────────────┬──────────────►◄
                                          │              └─/file_space_name─┘│
                                          └──user─────────────────────────┘
```

Parameters

connection_name
   Required. The host name and, optionally, the port of the server hosting the Web Gateway. Not case-sensitive.

fte
   Required. Indicates that the URI is addressed to the Web Gateway. Case-sensitive.

admin
   Required. Indicates that you are using the administrative functions of the Web Gateway. Case-sensitive.

filespace
   Indicates that you are addressing a file space resource. For more information about file spaces, see "File spaces" on page 313. Case-sensitive.

   One of the parameters **filespace** or **user** is required.

file_space_name
   The name of the file space you are addressing. This is the name of the user associated with the file space. The value of *file_space_name* must be 255 characters or fewer in length. Case-sensitive.

   Only applicable if you specify **filespace**. Optional if you use the HTTP verb GET, required if you use POST or DELETE. If you use the HTTP verb GET and do not provide a value for *file_space_name*, the Web Gateway returns a list of all file spaces.

user
   Indicates that you are addressing the set of mappings between web user ID and MQMD user ID. For more information about the format of this set of mappings, see "XML format for mapping web user ID to an MQMD user ID" on page 892. Case-sensitive.

   One of the parameters **filespace** or **user** is required.

Examples

For example, to address a file space resource that is owned by the user `sarah`, use the following URI:

`http://example.org/wmqfte/admin/filespace/sarah/`

In this example:
- `http://example.org` is the host system.
- `/wmqfte` indicates that the URI is a WebSphere MQ Managed File Transfer URI.
- `/admin` indicates that you are accessing administrative functions of the Web Gateway.
- `/filespace` indicates that the resource being addressed is a file space resource.
- `/sarah/` is the identifying token. This token is the name of the file space, which is also the name of the user who owns the file space.

For example, to address the set of mappings between user ID and MQMD ID, use the following URI:

`http://example.org/wmqfte/admin/user`

Content types for administering the Web Gateway:

HTTP requests that you submit to the WebSphere MQ Managed File Transfer Web Gateway administration API must have a media type of `application/xml`. Responses from the Web Gateway also have a media type of `application/xml`.

Request

Content transferred to WebSphere MQ Managed File Transfer using HTTP, as part of a request to the administration API, must be in one of the formats in the following table.

Media-type	Valid WebSphere MQ Managed File Transfer resources	Allowed verbs
application/xml	Filespace	• POST (create an instance of a file space) • GET (view the current configuration of a file space) • DELETE (delete an instance of a file space)

Response body

If an HTTP request is successful, the Web Gateway returns a response with a media type of `application/xml`. For details of the XML schema for this response, see "File space information response format" on page 889.

HTTP response codes from the Web Gateway administration API:

Status codes are returned in HTTP responses to requests made to the WebSphere MQ Managed File Transfer Web Gateway administration API.

The header of a response returned by the Web Gateway contains an HTTP response code. The HTTP header in the following example contains the HTTP response code 200 OK:

```
HTTP/1.1 200 OK
Server: WAS/6.0
Content-length: 0
```

The following table describes the possible values for the HTTP response code and some of the additional WebSphere MQ Managed File Transfer error codes that can be returned by the administration API:

Table 57. HTTP response codes

HTTP response code	Example WebSphere MQ Managed File Transfer error codes	Example description
200 OK	None	A valid request has been handled correctly and optionally a response has been provided to the user.
202 Accepted	None	A valid request has been handled correctly but WebSphere MQ Managed File Transfer does not guarantee that the requested action has completed.
400 Bad Request	BFGWI0501	The URI is not valid because it is missing an administration resource type.
404 Not Found	BFGWI0515	The requested resource cannot be found. For example, the file space does not exist.
405 Method Not Allowed	BFGWI0516	The requested resource does not support the HTTP verb that has been used in the request. For example, a HEAD method is used on `/admin/filespace/` `file_space_name/` and the only valid methods are POST, GET or DELETE.
415 Unsupported Media Type	BFGWI0517	An administration request has been received with a media type, specified by the `Content-type` HTTP header, that is not supported. For more information on the media types supported by the administration API, see the topic "Content types for administering the Web Gateway" on page 887
500 Internal Server Error	BFGWI0018	An internal error has been encountered when handling the request. An FFDC or ABEND file has been produced.

Table 57. HTTP response codes (continued)

HTTP response code	Example WebSphere MQ Managed File Transfer error codes	Example description
502 Bad Gateway	BFGWI0019	The request cannot be completed because an error occurred outside WebSphere MQ Managed File Transfer. For example, a WebSphere MQ queue manager is not available.
503 Service Unavailable	BFGWI0020	The destination is temporarily unavailable. For example, a WebSphere MQ queue is full.
504 Gateway Timeout	BFGWI0021	An attempt to complete the request has timed out because of time limits imposed by WebSphere MQ Managed File Transfer, or because of time limits imposed by the HTTP client.

For information about additional WebSphere MQ Managed File Transfer error response codes that can be returned by the Web Gateway, see the topic Diagnostic messages.

Administration response and request formats:

The WebSphere MQ Managed File Transfer Web Gateway returns responses to queries in one of two formats: XML or JavaScript Object Notation (JSON). You can submit requests to create, modify, and delete file spaces, or map user names to MQMD user IDs to the Web Gateway only in XML format.

You can specify the format of the response from the Web Gateway by including the `Accept: return-type` header in the request or by including the query `accept=return-type` in the URI. You can use a web application to parse the content of the XML or JSON response and display it in an appropriate format to a web user.

The default format is XML. If you specify the format using both the `Accept:` header and the query `accept=` in the URI, the Web Gateway returns a response in the format specified by the query in the URI.

File space information response format:

When you request information about the definition and attributes of a file space from the WebSphere MQ Managed File Transfer Service Web Gateway the response is returned in XML format or in JSON format. The XML response conforms to the schema `FileSpaceInfo.xsd`, which is located in the `MQ_INSTALLATION_PATH`/mqft/samples/schema directory of your MQMFT installation.

XML

The following example shows the format of a simple file space information XML response.

```
<filespaces xsi:noNamespaceSchemaLocation="FileSpaceInfo.xsd">
  <filespace transfers="1" location="/tmp/filespace/daniel" name="daniel">
    <quota bytes="1048576"/>
    <writers>
      <authorized>
        <agent-user>daniel</agent-user>
        <agent-user>SYS.ADMIN.*</agent-user>
      </authorized>
      <unauthorized>
        <agent-user>dave</agent-user>
```

```
        </unauthorized>
      </writers>
    </filespace>
</filespaces>
```

JSON

The following example shows the format of a simple file space information JSON response.

```
{
  "filespaces":{
    "filespace":{
      "transfers":"1",
      "location":"/tmp/filespace/daniel",
      "name":"daniel",
      "writers":{
        "authorized":{
          "agent-user":"daniel",
          "agent-user":"SYS.ADMIN.*"
        },
        "unauthorized":{
          "agent-user":"dave"
        }
      },
      "quota":{
        "bytes":"1048576"
      }
    }
  }
}
```

Understanding the file space information response

The elements and attributes of the file space information response are described in the following list:

filespaces
> Group containing one or more <filespace> elements.

filespace
> Group containing the information for the file space.

Attribute	Description
transfers	The number of transfers that are in progress to the file space.
location	The location in the file system of the file space.
name	The name of the file space.
integrity-check-result	The result of an integrity check on the file space. Valid values are: • OK • MISSING_FILESYSTEM • MISSING_DATABASEENTRY

quota Element describing the amount of file system space that the file space can use.

Attribute	Description
bytes	The maximum number of bytes that the file space can use.

writers
> Group containing information about which users are authorized and not authorized to access the file space.

authorized

Group containing a list of users that are authorized to access the file space.

unauthorized

Group containing a list of users that are not authorized to access the file space. If a user name or a user wildcard match appears in both the authorized and the unauthorized lists, they are not authorized to access the file space.

agent-user

Element containing the name of the user that is authorized or unauthorized. This user name can include wildcard characters, to match multiple users.

File space create or alter request format:

You can request to create or alter a file space from the WebSphere MQ Managed File Transfer Web Gateway by including content in XML format in the HTTP request. The XML format conforms to the schema `FileSpaceInfo.xsd`, which is located in the `MQ_INSTALLATION_PATH`/mqft/samples/schema directory of your WMQMFT installation.

The following example shows the format of an XML request to create a file space.

```
<filespaces>
  <filespace>
    <quota bytes="1048576"/>
    <writers>
      <authorized>
        <agent-user>SYS.ADMIN.*</agent-user>
      </authorized>
      <unauthorized>
        <agent-user>dave</agent-user>
      </unauthorized>
    </writers>
  </filespace>
</filespaces>
```

The following example shows the format of an XML request to modify the configuration of an existing file space. You must use the `action=add`, `action=remove` and `action=overwrite` attributes to change the lists of authorized and unauthorized writers.

```
<filespaces>
  <filespace>
    <quota bytes="2097152"/>
    <writers>
      <authorized action="add">
        <agent-user>emily</agent-user>
      </authorized>
      <unauthorized action="remove">
        <agent-user>dave</agent-user>
      </unauthorized>
    </writers>
  </filespace>
</filespaces>
```

Understanding the file space creation or modification request

The elements and attributes of the request are described in the following list:

filespaces

Element containing a single <filespace> element.

filespace

Group element containing the information for a file space.

quota Element describing the amount of file system space that the file space can use. If a user submits a file transfer request that would cause the file space to exceed its quota, the transfer fails and an error is produced.

Attribute	Description
bytes	The maximum number of bytes that the file space can use.

writers

Group containing information about which users are authorized and not authorized to access the file space.

authorized

Group containing a list of users that are authorized to access the file space.

Attribute	Description
action	The action to take on the agent-user names specified in the child elements. Valid options are: • `add` - add new agent-user names to the authorized list • `remove` - remove agent-user names from the authorized list • `overwrite` - replace all of the existing authorized list with the list provided.

unauthorized

Group containing a list of users that are not authorized to access the file space. If a user is included in both the authorized and unauthorized lists, they are not authorized to access the file space.

Attribute	Description
action	The action to take on the agent-user names specified in the child elements. Valid options are: • `add` - add new agent-user names to the unauthorized list • `remove` - remove agent-user names from the unauthorized list • `overwrite` - replace all of the existing unauthorized list with the list provided.

agent-user

Element containing the name of the user that is authorized or unauthorized. This user name can include wildcard characters, to match multiple users.

XML format for mapping web user ID to an MQMD user ID:

You can create a set of mappings between web user ID and WebSphere MQ Message Descriptor (MQMD) user ID by submitting a request to the WebSphere MQ Managed File Transfer Web Gateway. The HTTP request must include content in the following XML format.

The following example shows the format of an XML request to create a set of mappings. To modify an existing set of mappings, use the same format.

```
<users>
  <user>
    <userID>mike</userID>
    <mqmdUserID>mqmike</mqmdUserID>
```

```
    </user>
    <user>
      <userID>lisa</userID>
      <mqmdUserID>mqlisa</mqmdUserID>
    </user>
</users>
```

If you attempt to start a file upload with a web user ID that is not mapped to an MQMD user ID, the value of the defaultMQMDUserID initialization parameter is used. The value of this parameter is set when you deploy the Web Gateway application to an application server environment. For more information, see "Deploying the Web Gateway with WebSphere Application Server Version 7.0" on page 184 and "Preparing to deploy the Web Gateway with WebSphere Application Server Community Edition" on page 167.

Understanding the request to create or change user ID mappings

The elements and attributes of the request are described in the following list:

users Group element containing <user> elements.

user Element containing the information for a user of the Web Gateway.

userID
> Element containing the web user ID for the user. This is the user ID that is defined in the application server environment that hosts the Web Gateway.

mqmdUserID
> Element containing the name of the MQMD user ID (the WebSphere MQ user ID that is supplied in the message descriptor) to use in file upload transfers initiated by the user.
>
> The mqmdUserID attribute has a maximum length of 12 characters.

File space administration logging format:

When a file space is created, altered, or deleted the changes to the file space are logged in the event log of the application server hosting the Web Gateway. This allows an administrator to view the changes that have been made to file spaces.

Log format
```
FTELOG: operation - status. Requested by user_ID at host_name.
Information: information
```

operation
> The operation that was requested to be performed on the file space. The values of operation are:
> - `create file space`
> - `modify file space`
> - `delete file space`

status Whether the requested operation was successful. The values of status are:
> - `successful`
> - `failed`, in this case a reason for the failure is also given

user_ID
> The user name of the user that requested the file space operation.

host_name
> The host name of the system that the user made the request from.

information
> Information about the request. For example:

```
        File space: fred, quota: 123456 bytes, added authorized writers: [tom dick harry],
        added unauthorized writers: [tarzan jane], removed unauthorized writers: [bob]
```

These log messages are written to the application server's event log. These files can be found in the following directories:

- For WebSphere Application Server version 7.0, *WAS7_install_location*/profiles/*profile_name*/logs/ *server_name*
- For WebSphere Application Server Community Edition, *WASCE_install_location*/var/log

fteCreateWebAgent (create a WebSphere MQ Managed File Transfer web agent)

The **fteCreateWebAgent** command creates an agent and its associated configuration for use with the Web Gateway. This command is provided with WebSphere MQ Managed File Transfer Server.

Purpose

Use the **fteCreateWebAgent** command to create a web agent. This command provides you with the MQSC commands that you must run on the queue manager that is used by the agent to create the following agent queues:

- SYSTEM.FTE.AUTHADM1.*agent_name*
- SYSTEM.FTE.AUTHAGT1.*agent_name*
- SYSTEM.FTE.AUTHMON1.*agent_name*
- SYSTEM.FTE.AUTHOPS1.*agent_name*
- SYSTEM.FTE.AUTHSCH1.*agent_name*
- SYSTEM.FTE.AUTHTRN1.*agent_name*
- SYSTEM.FTE.COMMAND.*agent_name*
- SYSTEM.FTE.DATA.*agent_name*
- SYSTEM.FTE.EVENT.*agent_name*
- SYSTEM.FTE.REPLY.*agent_name*
- SYSTEM.FTE.STATE.*agent_name*

Because the agent is for use with the Web Gateway, two queues are created in addition to the previous list:

- SYSTEM.FTE.WEB.RESP.*agent_name*
- SYSTEM.FTE.WEB.*gateway_name*

These queues are internal system queues that you must not modify, delete, or read messages from unless you are deleting the agent. The MQSC commands to run are also supplied in a file in the following location: *MQ_DATA_PATH*\mqft\config*coordination_qmgr_name*\agents*agent_name*\ *agent_name*_create.mqsc.

If you later want to delete the agent, this command also provides you with the MQSC commands you must run to clear then delete the queues belonging to the agent. The MQSC commands are in a file in the following location: *MQ_DATA_PATH*\mqft\config*coordination_qmgr_name*\agents*agent_name*\ *agent_name*_delete.mqsc.

WebSphere MQ Managed File Transfer provides advanced agent properties that help you configure agents. These properties are described in Properties files for WebSphere MQ File Transfer Edition.

Note: The user that your web agent runs as must be the same as, or in the same group as, the user that your application server runs as.

Limitations of the web agent

- A web agent can be only the source agent for transfers initiated by a Web Gateway. If you attempt to perform a transfer with a web agent as the source by another method, the transfer fails with return code 68 (TRANSFER_NOT_SUPPORTED).
- A web agent can only be the destination agent for a transfer when the destination is specified as a file space. If you attempt to perform a transfer with a web agent as the destination agent but a different destination type the transfer will fail with the following error message: BFGCH0103: The transfer request specifies Web Gateway agent '*web_agent_name*' as the destination agent. Web Gateway agents can be the destination only for a transfer to a file space.
- A web agent cannot monitor a resource. If you attempt to create a resource monitor for a web agent, the command fails with return code 113 (MONITOR_NOT_SUPPORTED).

Syntax

fteCreateWebAgent

►►─fteCreateWebAgent──-agentName────*agent_name*──-agentQMgr────*agent_qmgr_name*──-webGatewayName────*gateway_name*───►

►────┬──────────────────────────────────────┬────┬─────────────────────────────────┬────────────────────►
    └─-agentQMgrHost────*agent_qmgr_host*─┘   └─-agentQMgrPort────*agent_qmgr_port*─┘

►────┬──┬──┬───────────────────────────────┬──┬──────────────────────┬──────►
    └─-agentQMgrChannel────*agent_qmgr_channel*─┘ └─-agentDesc────*agent_description*─┘ ├──-ac──────────────┤
                                                                └─-authorityChecking─┘

►────┬──────────────────────────────────┬──┬─────┬──►
    └─-p──*configuration_options*─┘ └─-f─┘

►────┬──-s──┬─────────────────┬──-su────*user_name*──┬───────────────────┬──┬──────────────┬──┬──────────────┬──┬─►◄
    │   └─*service_name*─┘   └─-sp──*password*─┘ └─-sj──*options*─┘ └─-sl──*options*─┘
    └──-n───┘

Parameters

-agentName *agent_name*
    Required. The name of the agent to create. The agent name must be unique to its coordination queue manager.

    For more information about naming agents, see Object naming conventions .

-agentQMgr *agent_qmgr_name*
    Required. The name of the agent queue manager.

-webGatewayName *gateway_name*
    Required. The name of the Web Gateway that the agent is a component of.

    For more information about naming Web Gateways, see Object naming conventions .

-agentQMgrHost *agent_qmgr_host*
    Optional. The host name or IP address of the agent queue manager. If you do not specify this parameter, a bindings mode connection is assumed.

-agentQMgrPort *agent_qmgr_port*
    Optional. The port number used for client connections to the agent queue manager. This parameter is used only if you have also specified the **agentQMgrHost** parameter. If you do not specify the **agentQMgrPort** parameter, a default port of 1414 is used.

-agentQMgrChannel *agent_qmgr_channel*
    Optional. This parameter is used only if you have also specified the **agentQMgrHost** parameter. If you do not specify the **agentQMgrChannel** parameter, a default channel of SYSTEM.DEF.SVRCONN is used.

-agentDesc *agent_description*

Optional. A description of the agent, which is displayed in WebSphere MQ Explorer.

-ac or -authorityChecking

Optional. This parameter enables authority checking. If you specify this parameter, the agent checks that users who are submitting requests are authorized to perform the requested action.

-p *configuration_options*

Optional. The name of the set of configuration options that is used to create the agent. By convention, this is the name of a coordination queue manager. If you do not specify this parameter, the default set of configuration options is used.

-f Optional. Forces the command to overwrite the existing configuration.

-s *service_name*

Optional (Windows only). Indicates that the agent is to run as a Windows service. If you do not specify *service_name*, the service is named mqmftAgent<*AGENT*><*QMGR*>, where <*AGENT*> is the agent name and <*QMGR*> is your agent queue manager name.

The display name for the service, which is shown in the Windows **Services** window in the **Name** column, is always **WebSphere MQ Managed File Transfer agent <*AGENT*>@<*QMGR*>**.

-su *user_name*

Optional (Windows only). When the agent is to run as a Windows service, this parameter specifies the name of the account under which the service should run. To run the agent using a Windows domain user account specify the value in the form DomainName\UserName. To run the service using an account from the local built-in domain specify the value in the form UserName.

The Windows user account that you specify using the **-su** parameter must have the **Log on as a service** right. For information about how to grant this right, see "Guidance for running an agent or logger as a Windows service" on page 366.

Required when **-s** specified. Equivalent to **-serviceUser**.

-sp *password*

Optional (Windows only). Password for the user account set by **-su** or **-serviceUser** parameter.

This parameter is only valid when **-s** is specified. Equivalent to **-servicePassword**. If you do not specify this parameter when you specify the **-s** parameter, a warning message is produced. This message warns you that you must set the password using the Windows Services tool before the service will start successfully.

-sj *options*

Optional (Windows only). When the agent is started as a Windows service, defines a list of options in the form of -D or -X that will be passed to the JVM. The options are separated using a number sign (#) or semicolon (;) character. If you need to embed any # or ; characters, put them inside single quotation marks.

This parameter is only valid when **-s** is specified. Equivalent to **-serviceJVMOptions**.

-sl *options*

Optional (Windows only). Sets the Windows service log level. Valid options are: error, info, warn, debug. The default is info. This option can be useful if you are having problems with the Windows service. Setting it to debug gives more detailed information in the service log file.

This parameter is only valid when **-s** is specified. Equivalent to **-serviceLogLevel**.

-n Optional (Windows only). Indicates that the agent is to be run as a normal process. This is mutually exclusive with the **-s** option. If neither the **-s** nor the **-n** option is specified, then the agent is configured as a normal Windows process.

Equivalent to **-normal**.

-? or -h
   Optional. Displays the command syntax.

Example

In this example, the agent WEBAGENT1 is created with an agent queue manager QM_NEPTUNE and the Web Gateway GATEWAY_ONE. The agent uses the default coordination queue manager:

```
fteCreateWebAgent -agentName WEBAGENT1 -webGatewayName GATEWAY_ONE -agentQMgr QM_NEPTUNE
              -agentQMgrHost myhost.ibm.com -agentQMgrPort 1415 -agentQMgrChannel CHANNEL1
```

Return codes

0   Command completed successfully.

1   Command ended unsuccessfully.

Database tables used by the Web Gateway

The WebSphere MQ Managed File Transfer Web Gateway uses the following database tables to configure and secure user file spaces.

The following database tables are used by the Web Gateway: do not delete or modify these tables or any of the data contained in them.
- FILE_SPACE
- FILE_SPACE_ENTRY
- PERMISSIONS
- USER_MQMD_MAPPING
- WEBGATEWAY_CONFIG

The Web Gateway also uses the audit information in the database logger tables to provide the user with transfer information. For more information, see "Database tables used by the logger" on page 691.

The database tables used by the Web Gateway can be located in the same database as the tables used by the database logger, as long as the two sets of tables have different schema names.

Related tasks:

"Setting up a database for use with file spaces" on page 165
Before you can use file spaces you must set up database tables for the Web Gateway to store file space information in. You can create these tables in your existing log database, or create a new database to contain the tables.

Related reference:

"Database tables used by the logger" on page 691
When you have installed and configured the logger, the following database tables are created:

Using Apache Ant with WebSphere MQ Managed File Transfer

fteAnt (run Ant tasks in a WebSphere MQ Managed File Transfer environment)
The **fteAnt** command runs Ant scripts in an environment that has WebSphere MQ Managed File Transfer Ant tasks available.

Purpose

Use the **fteAnt** command to run an Ant script in an environment with WebSphere MQ Managed File Transfer. Unlike the standard **ant** command, **fteAnt** requires that you define a script file.

The **fteAnt** command cannot be run directly on an IBM 4690 system. However, an IBM 4690 system can be referred to by an Ant script. For more information about using WebSphere MQ Managed File Transfer

in the IBM 4690 environment, see "Using WebSphere MQ Managed File Transfer in a retail environment" on page 35

Syntax

fteAnt

Parameters

-debug or -d
Optional. Generate debugging output.

-quiet or -q
Optional. Generate minimal output.

-verbose or -v
Optional. Generate verbose output.

-keep-going or -k
Optional. Execute all targets that do not depend on failed targets.

-D *property=value*
Optional. Use *value* for a given *property*. Properties that are set with **-D** take precedence over those set in a properties file.

Use the property **com.ibm.wmqfte.propertyset** to specify the set of configuration options that are used for Ant tasks. Use the name of a non-default coordination queue manager as the value for this property. Ant tasks then use the set of configuration options that are associated with this non-default coordination queue manager. If you do not specify this property, the default set of configuration options that are based on the default coordination queue manager is used. If you specify the **cmdqm** attribute for an Ant task, this attribute takes precedence over the set of configuration options that are specified for the **fteAnt** command. This behavior applies regardless of whether you are using the default set of configuration options or specifying a set with the **com.ibm.wmqfte.propertyset** property.

-propertyfile (*name***)**
Optional. Load all properties from a file with **-D** properties taking precedence.

-f (*Ant script***), -file (***Ant script***), or -buildfile (***Ant script***)**
Required. Specifies the name of the Ant script to run.

targets
Optional. The name of one or more targets to run from the Ant script. If you do not specify a value for this parameter, the default target for the script is run.

-version
Optional. Displays the WebSphere MQ Managed File Transfer command and Ant versions.

-? or -h
Optional. Displays command syntax.

Example

In this example, the target **copy** in Ant script `fte_script.xml` is run and the command writes debugging output to standard out.

```
fteAnt -d -f fte_script.xml copy
```

Return codes

0   Command completed successfully.

1   Command ended unsuccessfully.

Other status return codes can also be specified from Ant scripts, for example by using the Ant fail task.

Ant tasks provided by WebSphere MQ Managed File Transfer

WebSphere MQ Managed File Transfer provides a number of Ant tasks that you can use to access file transfer capabilities.

Tasks

- "fte:awaitoutcome"
- fte:call
- fte:cancel
- fte:filecopy
- fte:filemove
- fte:ignoreoutcome
- fte:ping
- fte:uuid

Nested parameters

The following nested parameters describe nested sets of elements, which are common across several of the supplied Ant tasks:

- fte:filespec
- fte:metadata
- Parameters for program invocation

fte:awaitoutcome:

Waits for a **fte:filecopy**, **fte:filemove**, or **fte:call** operation to complete.

Attributes

id Required. Identifies the transfer to await an outcome from. Typically, this is a property set by the idProperty attribute of the fte:filecopy, fte:filemove, or fte:call tasks.

rcproperty
Required. Names a property to store the return code of the **fte:awaitoutcome** task in.

timeout
Optional. The maximum amount of time, in seconds, to wait for the operation to complete. The minimum timeout is one second. If you do not specify a timeout value, the **fte:awaitoutcome** task waits forever for the outcome of the operation to be determined.

Example

In this example a file copy is started, and its identifier is stored in the `copy.id` property. While the copy is progressing, other processing can take place. The **fte:awaitoutcome** statement is used to wait until the copy operation completes. The **fte:awaitoutcome** statement identifies which operation to wait for using the identifier stored in the `copy.id` property. The **fte:awaitoutcome** stores a return code indicating the outcome of the copy operation into a property called `copy.result`.

```
<-- issue a file copy request -->
<fte:filecopy cmdqm="qm1@localhost@1414@SYSTEM.DEF.SVRCONN"
              src="agent1@qm1"
              dst="agent1@qm1"
              idproperty="copy.id"
              outcome="defer"/>

<!-- do some other things -->

<!-- get the result of the file copy -->
<fte:awaitoutcome id="${copy.id}" rcProperty="copy.result"/>
```

Related reference:

"Using Apache Ant with WebSphere MQ Managed File Transfer" on page 328
WebSphere MQ Managed File Transfer provides tasks that you can use to integrate file transfer function into the Apache Ant tool.

"Ant tasks provided by WebSphere MQ Managed File Transfer" on page 899
WebSphere MQ Managed File Transfer provides a number of Ant tasks that you can use to access file transfer capabilities.

fte:call:

You can use the **fte:call** task to remotely call scripts and programs.

This task allows you to send a **fte:call** request to an agent. The agent processes this request by running a script or program and returning the outcome. The commands to call must be accessible to the agent. Ensure the commandPath property value in the `agent.properties` file includes the location of the commands to call. Any path information specified by the command nested element must be relative to the locations specified by the commandPath property. By default commandPath is empty so that the agent cannot call any commands. For more information about this property, see Using commandPath.

For more information about the `agent.properties` file, see "The agent.properties file" on page 548.

Attributes

agent
Required. Specifies the agent to submit the **fte:call** request to. Specify the agent information in the form: *agentname@qmgrname* where *agentname* is the name of the agent and *qmgrname* is the name of the queue manager that this agent is directly connected to.

cmdqm
Optional. The command queue manager to submit the request to. Specify this information in the form *qmgrname@host@port@channel*, where:

- *qmgrname* is the name of the queue manager
- *host* is the optional host name of the system where the queue manager is running
- *port* is the optional port number that the queue manager is listening on
- *channel* is the optional SVRCONN channel to use

If you omit the *host, port,* or *channel* information for the command queue manager, the connection information specified in the `command.properties` file is used. For more information, see "The command.properties file" on page 545.

You can use the **com.ibm.wmqfte.propertySet** property to specify which `command.properties` file to use. For more information, see com.ibm.wmqfte.propertySet.

If you do not use the cmdqm attribute, the task defaults to using the `com.ibm.wmqfte.ant.commandQueueManager` property, if this property is set. If the `com.ibm.wmqfte.ant.commandQueueManager` property is not set, a connection to the default queue manager, defined in the `command.properties` file, is attempted. The format of the `com.ibm.wmqfte.ant.commandQueueManager` property is the same as the cmdqm attribute, that is, `qmgrname@host@port@channel`.

idproperty
Optional unless you have specified an `outcome` of `defer`. Specifies the name of a property to assign the transfer identifier to. Transfer identifiers are generated at the point a transfer request is submitted and you can use transfer identifiers to track the progress of a transfer, diagnose problems with a transfer, and cancel a transfer.

You cannot specify this property if you have also specified an `outcome` property of `ignore`. However, you must specify idproperty if you have also specified an `outcome` property of `defer`.

jobname
Optional. Assigns a job name to the **fte:call** request. You can use job names to create logical groups of transfers. Use the "fte:uuid" on page 911 task to generate pseudo-unique job names. If you do not use the jobname attribute, the task defaults to using the `com.ibm.wmqfte.ant.jobName` property value, if this property is set. If you do not set this property, no job name is associated with the **fte:call** request.

origuser
Optional. Specifies the originating user identifier to associate with the **fte:call** request. If you do not use the origuser attribute, the task defaults to using the user ID that is used to run the Ant script.

outcome
Optional. Determines whether the task waits for the **fte:call** operation to complete before returning control to the Ant script. Specify one of the following options:

await The task waits for the **fte:call** operation to complete before returning. When an `outcome` of `await` is specified the `idproperty` attribute is optional.

defer The task returns as soon as the **fte:call** request has been submitted and assumes that the outcome of the call operation is dealt with later using either the awaitoutcome or ignoreoutcome tasks. When an `outcome` of `defer` is specified the `idproperty` attribute is required.

ignore If the outcome of the **fte:call** operation is not important, you can specify a value of `ignore`. The task then returns as soon as the **fte:call** request has been submitted, without allocating any resources for tracking the outcome of the command. When an `outcome` of `ignore` is specified the `idproperty` attribute cannot be specified.

If you do not specify the outcome attribute, the task defaults to using the value `await`.

rcproperty
Optional. Specifies the name of a property to assign the result code of the **fte:call** request to. The result code reflects the overall outcome of the **fte:call** request.

You cannot specify this property if you have also specified an `outcome` property of `ignore` or `defer`. However, you must specify rcproperty if you have specified an outcome of `await`.

Parameters specified as nested elements

fte:command
Specifies the command to be called by the agent. You can only associate a single `fte:command` element with a given **fte:call** operation. The command to be called must be located on the path specified by the commandPath property in the agent's `agent.properties` file.

fte:metadata

You can specify metadata to associate with the call operation. This metadata is recorded in the log messages generated by the call operation. You can only associate a single block of metadata with a given transfer element; however this block can contain many pieces of metadata.

Example

This example shows how to call a command at AGENT1 running on queue manager QM1. The command to call is the script `command.sh`, and the script is called with a single argument of `xyz`. The command `command.sh` is located on the path specified by the commandPath property in the agent's `agent.properties` file.

```
<fte:call cmdqm="QM0@localhost@1414@SYSTEM.DEF.SVRCONN"
        agent="AGENT1@QM1"
        rcproperty="call.rc"
        origuser="bob"
        jobname="${job.id}">

    <fte:command command="command.sh" successrc="1" retrycount="5" retrywait="30">
        <fte:arg value="xyz"/>
    </fte:command>

    <fte:metadata>
        <fte:entry name="org.foo.accountName" value="BDG3R"/>
    </fte:metadata>

</fte:call>
```

Related reference:

"Using Apache Ant with WebSphere MQ Managed File Transfer" on page 328
WebSphere MQ Managed File Transfer provides tasks that you can use to integrate file transfer function into the Apache Ant tool.

"Ant tasks provided by WebSphere MQ Managed File Transfer" on page 899
WebSphere MQ Managed File Transfer provides a number of Ant tasks that you can use to access file transfer capabilities.

fte:cancel:

Cancels a WebSphere MQ Managed File Transfer managed transfer or managed call. A managed transfer might have been created using the **fte:filecopy** or **fte:filemove** tasks. A managed call might have been created using the **fte:call** task.

Attributes

agent
Required. Specifies the agent to submit the **fte:cancel** request to. The value is in the form: *agentname@qmgrname* where *agentname* is the name of the agent and *qmgrname* is the name of the queue manager that this agent is directly connected to.

cmdqm
Optional. The command queue manager to submit the request to. Specify this information in the form *qmgrname@host@port@channel,* where:

- *qmgrname* is the name of the queue manager
- *host* is the optional host name of the system where the queue manager is running
- *port* is the optional port number that the queue manager is listening on
- *channel* is the optional SVRCONN channel to use

If you omit the *host, port,* or *channel* information for the command queue manager, the connection information specified in the `command.properties` file is used. For more information, see "The command.properties file" on page 545.

You can use the **com.ibm.wmqfte.propertySet** property to specify which `command.properties` file to use. For more information, see com.ibm.wmqfte.propertySet.

If you do not use the cmdqm attribute, the task defaults to using the `com.ibm.wmqfte.ant.commandQueueManager` property, if this property is set. If the `com.ibm.wmqfte.ant.commandQueueManager` property is not set, a connection to the default queue manager, defined in the `command.properties` file, is attempted. The format of the `com.ibm.wmqfte.ant.commandQueueManager` property is the same as the cmdqm attribute, that is, `qmgrname@host@port@channel`.

id Required. Specifies the transfer identifier of the transfer to cancel. Transfer identifiers are generated at the point a transfer request is submitted by both the fte:filecopy and fte:filemove tasks.

origuser
Optional. Specifies the originating user identifier to associate with the **cancel** request. If the origuser attribute is not used, the task defaults to using the user ID that is used to run the Ant script.

Example

The example sends a **fte:cancel** request to the command queue manager qm0. The **fte:cancel** request is targeted at agent1 on queue manager qm1 for the transfer identifier populated by the `transfer.id` variable. The request is run using the "bob" user ID.

```
<fte:cancel cmdqm="qm0@localhost@1414@SYSTEM.DEF.SVRCONN"
            agent="agent1@qm1"
            id="${transfer.id}"
            origuser="bob"/>
```

Related reference:
"Using Apache Ant with WebSphere MQ Managed File Transfer" on page 328
WebSphere MQ Managed File Transfer provides tasks that you can use to integrate file transfer function into the Apache Ant tool.

"Ant tasks provided by WebSphere MQ Managed File Transfer" on page 899
WebSphere MQ Managed File Transfer provides a number of Ant tasks that you can use to access file transfer capabilities.

fte:filecopy:

The **fte:filecopy** task copies files between WebSphere MQ Managed File Transfer agents. The file is not deleted from the source agent.

Attributes

cmdqm
Optional. The command queue manager to submit the request to. Specify this information in the form *qmgrname@host@port@channel*, where:
- *qmgrname* is the name of the queue manager
- *host* is the optional host name of the system where the queue manager is running
- *port* is the optional port number that the queue manager is listening on
- *channel* is the optional SVRCONN channel to use

If you omit the *host*, *port*, or *channel* information for the command queue manager, the connection information specified in the `command.properties` file is used. For more information, see "The command.properties file" on page 545.

You can use the **com.ibm.wmqfte.propertySet** property to specify which `command.properties` file to use. For more information, see com.ibm.wmqfte.propertySet.

If you do not use the cmdqm attribute, the task defaults to using the `com.ibm.wmqfte.ant.commandQueueManager` property, if this property is set. If the

com.ibm.wmqfte.ant.commandQueueManager property is not set, a connection to the default queue manager, defined in the command.properties file, is attempted. The format of the com.ibm.wmqfte.ant.commandQueueManager property is the same as the cmdqm attribute, that is, qmgrname@host@port@channel.

dst
> Required. Specifies the destination agent for the copy operation. Specify this information in the form: agentname@qmgrname where agentname is the name of the destination agent and qmgrname is the name of the queue manager that this agent is directly connected to.

idproperty
> Optional unless you have specified an outcome of defer. Specifies the name of a property to assign the transfer identifier to. Transfer identifiers are generated at the point a transfer request is submitted and you can use transfer identifiers to track the progress of a transfer, diagnose problems with a transfer, and cancel a transfer.
>
> You cannot specify this property if you have also specified an outcome property of ignore. However, you must specify idproperty if you have also specified an outcome property of defer.

jobname
> Optional. Assigns a job name to the copy request. You can use job names to create logical groups of transfers. Use the "fte:uuid" on page 911 task to generate pseudo-unique job names. If you do not use the jobname attribute, the task defaults to using the com.ibm.wmqfte.ant.jobName property value, if this property is set. If you do not set this property, no job name is associated with the copy request.

origuser
> Optional. Specifies the originating user identifier to associate with the copy request. If you do not use the origuser attribute, the task defaults to using the user ID that is used to run the Ant script.

outcome
> Optional. Determines whether the task waits for the copy operation to complete before returning control to the Ant script. Specify one of the following options:
>
> **await** The task waits for the copy operation to complete before returning. When an outcome of await is specified the idproperty attribute is optional.
>
> **defer** The task returns as soon as the copy request has been submitted and assumes that the outcome of the copy operation is dealt with later using either the awaitoutcome or "fte:ignoreoutcome" on page 909 tasks. When an outcome of defer is specified the idproperty attribute is required.
>
> **ignore** If the outcome of the copy operation is not important, you can specify a value of ignore. The task then returns as soon as the copy request has been submitted, without allocating any resources for tracking the outcome of the transfer. When an outcome of ignore is specified the idproperty attribute cannot be specified.
>
> If you do not specify the outcome attribute, the task defaults to using the value await.

priority
> Optional. Specifies the priority to associate with the copy request. In general, higher priority transfer requests take precedence over lower priority requests. The priority value must be in the range 0 - 9 (inclusive). A priority value of 0 is the lowest priority and a value of 9 is the highest priority. If you do not specify the priority attribute, the transfer defaults to a priority of 0.

rcproperty
> Optional. Specifies the name of a property to assign the result code of the copy request to. The result code reflects the overall outcome of the copy request.
>
> You cannot specify this property if you have also specified an outcome property of ignore or defer. However, you must specify rcproperty if you specify an outcome of await.

src

> Required. Specifies the source agent for the copy operation. Specify this information in the form: *agentname@qmgrname* where *agentname* is the name of the source agent and *qmgrname* is the name of the queue manager that this agent is directly connected to.

Parameters specified as nested elements

fte:filespec

> Required. You must specify at least one file specification that identifies the files to copy. You can specify more than one file specification if required. See the fte:filespec topic for more information.

fte:metadata

> You can specify metadata to associate with the copy operation. This metadata is carried with the transfer and is recorded in the log messages generated by the transfer. You can only associate a single block of metadata with a given transfer element; however this block can contain many pieces of metadata. See the fte:metadata topic for more information.

fte:presrc

> Specifies a program invocation to take place at the source agent before the transfer starts. You can only associate a single `fte:presrc` element with a given transfer. See the program invocation topic for more information.

fte:predst

> Specifies a program invocation to take place at the destination agent before the transfer starts. You can only associate a single `fte:predst` element with a given transfer. See the program invocation topic for more information.

fte:postsrc

> Specifies a program invocation to take place at the source agent after the transfer has completed. You can only associate a single `fte:postsrc` element with a given transfer. See the program invocation topic for more information.

fte:postdst

> Specifies a program invocation to take place at the destination agent after the transfer has completed. You can only associate a single `fte:postdst` element with a given transfer. See the program invocation topic for more information.

If fte:presrc, fte:predst, fte:postsrc, fte:postdst, and exits do not return a success status, the rules are as follows in the order specified:

1. Run the source start exits. If source start exits fail the transfer fails and nothing further is run.
2. Run the pre-source call (when present). If the pre-source call fails, the transfer fails and nothing further is run.
3. Run the destination start exits. If the destination start exits fail the transfer fails and nothing further is run.
4. Run the pre-destination call (when present). If the pre-destination call fails, the transfer fails and nothing further is run.
5. Perform the file transfers.
6. Run the destination end exits. There is no failure status for these exits.
7. If the transfer is successful (if some files transfer successfully, it is deemed successful) run the post-destination call (if present). If the post-destination call fails, the transfer fails.
8. Run the source end exits. There is no failure status for these exits.
9. If the transfer is successful run the post-source call (if present). If the post-source call fails, the transfer fails.

Examples

This example shows a basic file transfer between `agent1` and `agent2`. The command to start the file transfer is sent to a queue manager called `qm0`, using a client transport mode connection. The result of the file transfer operation is assigned to the property called `copy.result`.

```
<fte:filecopy cmdqm="qm0@localhost@1414@SYSTEM.DEF.SVRCONN"
             src="agent1@qm1" dst="agent2@qm2"
             rcproperty="copy.result">

 <fte:filespec srcfilespec="/home/fteuser1/file.bin" dstfile="/home/fteuser2/file.bin"/>

</fte:filecopy>
```

This example shows the same file transfer, but with the addition of metadata and a program start to take place at the source agent after the transfer has completed.

```
<fte:filecopy cmdqm="qm0@localhost@1414@SYSTEM.DEF.SVRCONN"
             src="agent1@qm"1 dst="agent2@qm2"
             rcproperty="copy.result">


 <fte:metadata>
   <fte:entry name="org.example.departId" value="ACCOUNTS"/>
   <fte:entry name="org.example.batchGroup" value="A1"/>
 </fte:metadata>

 <fte:filespec srcfilespec="/home/fteuser1/file.bin" dstfile="/home/fteuser2/file.bin"/>

 <fte:postsrc command="/home/fteuser2/scripts/post.sh" successrc="1" >
   <fte:arg value="/home/fteuser2/file.bin"/>
  </fte:postsrc>
</fte:filecopy>
```

Related reference:

"Using Apache Ant with WebSphere MQ Managed File Transfer" on page 328
WebSphere MQ Managed File Transfer provides tasks that you can use to integrate file transfer function into the Apache Ant tool.

"Ant tasks provided by WebSphere MQ Managed File Transfer" on page 899
WebSphere MQ Managed File Transfer provides a number of Ant tasks that you can use to access file transfer capabilities.

fte:filemove:

The **fte:filemove** task moves files between WebSphere MQ Managed File Transfer agents. When a file has been successfully transferred from the source agent to the destination agent, the file is deleted from the source agent.

Attributes

cmdqm

> Optional. The command queue manager to submit the request to. Specify this information in the form *qmgrname@host@port@channel,* where:
>
> - *qmgrname* is the name of the queue manager
> - *host* is the optional host name of the system where the queue manager is running
> - *port* is the optional port number that the queue manager is listening on
> - *channel* is the optional SVRCONN channel to use
>
> If you omit the *host, port,* or *channel* information for the command queue manager, the connection information specified in the `command.properties` file is used. For more information, see "The command.properties file" on page 545.

You can use the **com.ibm.wmqfte.propertySet** property to specify which `command.properties` file to use. For more information, see com.ibm.wmqfte.propertySet.

If you do not use the cmdqm attribute, the task defaults to using the `com.ibm.wmqfte.ant.commandQueueManager` property, if this property is set. If the `com.ibm.wmqfte.ant.commandQueueManager` property is not set, a connection to the default queue manager, defined in the `command.properties` file, is attempted. The format of the `com.ibm.wmqfte.ant.commandQueueManager` property is the same as the cmdqm attribute, that is, `qmgrname@host@port@channel`.

dst

Required. Specifies the destination agent for the copy operation. Specify this information in the form: *agentname@qmgrname* where *agentname* is the name of the destination agent and *qmgrname* is the name of the queue manager that this agent is directly connected to.

idproperty

Optional unless you have specified an `outcome` of `defer`. Specifies the name of a property to assign the transfer identifier to. Transfer identifiers are generated at the point a transfer request is submitted and you can use transfer identifiers to track the progress of a transfer, diagnose problems with a transfer, and cancel a transfer.

You cannot specify this property if you have also specified an `outcome` property of `ignore`. However, you must specify `idproperty` if you have also specified an `outcome` property of `defer`.

jobname

Optional. Assigns a job name to the move request. You can use job names to create logical groups of transfers. Use the fte:uuid task to generate pseudo-unique job names. If you do not use the jobname attribute, the task defaults to using the `com.ibm.wmqfte.ant.jobName` property value, if this property is set. If you do not set this property, no job name is associated with the move request.

origuser

Optional. Specifies the originating user identifier to associate with the move request. If you do not use the origuser attribute, the task defaults to using the user ID that is used to run the Ant script.

outcome

Optional. Determines whether the task waits for the move operation to complete before returning control to the Ant script. Specify one of the following options:

await The task waits for the move operation to complete before returning. When an `outcome` of `await` is specified the `idproperty` attribute is optional.

defer The task returns as soon as the move request has been submitted and assumes that the outcome of the move operation is dealt with later using either the "fte:awaitoutcome" on page 899 or "fte:ignoreoutcome" on page 909 task. When an `outcome` of `defer` is specified the `idproperty` attribute is required.

ignore If the outcome of the move operation is not important, you can specify a value of `ignore`. The task then returns as soon as the move request has been submitted, without allocating any resources for tracking the outcome of the transfer. When an `outcome` of `ignore` is specified the `idproperty` attribute cannot be specified.

If you do not specify the outcome attribute, the task defaults to using the value `await`.

priority

Optional. Specifies the priority to associate with the move request. In general, higher priority transfer requests take precedence over lower priority requests. The priority value must be in the range 0 - 9 (inclusive). A priority value of 0 is the lowest priority and a value of 9 is the highest priority. If you do not specify the `priority` attribute, the transfer defaults to a priority of 0.

rcproperty

Optional. Specifies the name of a property to assign the result code of the move request to. The result code reflects the overall outcome of the move request.

You cannot specify this property if you have also specified an `outcome` property of `ignore` or `defer`. However, you must specify `rcproperty` if you have specified an outcome of `await`.

src

Required. Specifies the source agent for the move operation. Specify this information in the form: *agentname@qmgrname* where *agentname* is the name of the source agent and *qmgrname* is the name of the queue manager that this agent is directly connected to.

Parameters specified as nested elements

fte:filespec

Required. You must specify at least one file specification that identifies the files to move. You can specify more than one file specification if required. See the fte:filespec topic for more information.

fte:metadata

Optional. You can specify metadata to associate with the file move operation. This metadata is carried with the transfer and is recorded in the log messages generated by the transfer. You can only associate a single block of metadata with a given transfer element; however this block can contain many pieces of metadata. See the fte:metadata topic for more information.

fte:presrc

Optional. Specifies a program invocation to take place at the source agent before the transfer starts. You can only associate a single `fte:presrc` element with a given transfer. See the program invocation topic for more information.

fte:predst

Optional. Specifies a program invocation to take place at the destination agent before the transfer starts. You can only associate a single `fte:predst` element with a given transfer. See the program invocation topic for more information.

fte:postsrc

Optional. Specifies a program invocation to take place at the source agent after the transfer has completed. You can only associate a single `fte:postsrc` element with a given transfer. See the program invocation topic for more information.

fte:postdst

Optional. Specifies a program invocation to take place at the destination agent after the transfer has completed. You can only associate a single `fte:postdst` element with a given transfer. See the program invocation topic for more information.

If fte:presrc, fte:predst, fte:postsrc, fte:postdst, and exits do not return a success status, the rules are as follows in the order specified:

1. Run the source start exits. If source start exits fail the transfer fails and nothing further is run.
2. Run the pre-source call (when present). If the pre-source call fails, the transfer fails and nothing further is run.
3. Run the destination start exits. If the destination start exits fail the transfer fails and nothing further is run.
4. Run the pre-destination call (when present). If the pre-destination call fails, the transfer fails and nothing further is run.
5. Perform the file transfers.
6. Run the destination end exits. There is no failure status for these exits.
7. If the transfer is successful (if some files transfer successfully, the transfer is considered successful), run the post-destination call (if present). If the post-destination call fails, the transfer fails.
8. Run the source end exits. There is no failure status for these exits.
9. If the transfer is successful, run the post-source call (if present). If the post-source call fails, the transfer fails.

Examples

This example shows a basic file move between `agent1` and `agent2`. The command to start the file move is sent to a queue manager called `qm0`, using a client transport mode connection. The result of the file transfer operation is assigned to the property called `move.result`.

```
<fte:filemove cmdqm="qm0@localhost@1414@SYSTEM.DEF.SVRCONN"
              src="agent1@qm1" dst="agent2@qm2"
              rcproperty="move.result">

    <fte:filespec srcfilespec="/home/fteuser1/file.bin" dstfile="/home/fteuser2/file.bin"/>

</fte:filemove>
```

Related reference:

"Using Apache Ant with WebSphere MQ Managed File Transfer" on page 328
WebSphere MQ Managed File Transfer provides tasks that you can use to integrate file transfer function into the Apache Ant tool.

"Ant tasks provided by WebSphere MQ Managed File Transfer" on page 899
WebSphere MQ Managed File Transfer provides a number of Ant tasks that you can use to access file transfer capabilities.

fte:ignoreoutcome:

Ignore the outcome of a **fte:filecopy**, **fte:filemove**, or **fte:call** command. When you specify a **fte:filecopy**, **fte:filemove**, or **fte:call** task to have an outcome of `defer`, the Ant task allocates resources to tracking this outcome. If you are no longer interested in the outcome, you can use the **fte:ignoreoutcome** task to free those resources.

Attributes

id Required. Identifies the outcome that is no longer of interest. Typically you specify this identifier using a property that you set using the `idproperty` attribute of the "fte:filecopy" on page 903, "fte:filemove" on page 906, or "fte:call" on page 900 task.

Example

This example shows how you can use the fte:ignoreoutcome task to free the resources allocated to tracking the outcome of the earlier "fte:filecopy" on page 903 task.

```
<!-- issue a file copy request -->
<fte:filecopy cmdqm="qm1@localhost@1414@SYSTEM.DEF.SVRCONN"
                src="agent1@qm1" dst="agent1@qm1"
                idproperty="copy.id"
                outcome="defer"/>

<!-- do some other things -->

<!-- decide that the result of the copy is not interesting -->
<fte:ignoreoutcome id="${copy.id}"/>
```

Related reference:

"Using Apache Ant with WebSphere MQ Managed File Transfer" on page 328
WebSphere MQ Managed File Transfer provides tasks that you can use to integrate file transfer function into the Apache Ant tool.

"Ant tasks provided by WebSphere MQ Managed File Transfer" on page 899
WebSphere MQ Managed File Transfer provides a number of Ant tasks that you can use to access file transfer capabilities.

fte:ping:

Pings an agent to elicit a response and so determines if the agent is able to process transfers.

Attributes

agent
Required. Specifies the agent to submit the **fte:ping** request to. The value is in the form: *agentname@qmgrname* where *agentname* is the name of the agent and *qmgrname* is the name of the queue manager that this agent is directly connected to.

cmdqm
Optional. The command queue manager to submit the request to. Specify this information in the form *qmgrname@host@port@channel*, where:
- *qmgrname* is the name of the queue manager
- *host* is the optional host name of the system where the queue manager is running
- *port* is the optional port number that the queue manager is listening on
- *channel* is the optional SVRCONN channel to use

If you omit the *host, port,* or *channel* information for the command queue manager, the connection information specified in the command.properties file is used. For more information, see "The command.properties file" on page 545.

You can use the **com.ibm.wmqfte.propertySet** property to specify which command.properties file to use. For more information, see com.ibm.wmqfte.propertySet.

If you do not use the cmdqm attribute, the task defaults to using the com.ibm.wmqfte.ant.commandQueueManager property, if this property is set. If the com.ibm.wmqfte.ant.commandQueueManager property is not set, a connection to the default queue manager, defined in the command.properties file, is attempted. The format of the com.ibm.wmqfte.ant.commandQueueManager property is the same as the cmdqm attribute, that is, qmgrname@host@port@channel.

rcproperty
Required. Names a property to store the return code of the **ping** operation in.

timeout
Optional. The maximum amount of time, in seconds, for the task to wait for the agent to respond. The minimum timeout is zero seconds, however a timeout of minus one can also be specified such that the command waits forever for the agent to respond. If no value is specified for the timeout then the default is to wait up to 5 seconds for the agent to respond.

Example

This example sends a **fte:ping** request to agent1 hosted by qm1. The **fte:ping** request waits 15 seconds for the agent to respond. The outcome of the **fte:ping** request is stored in a property called ping.rc.

```
<fte:ping agent="agent1@qm1" rcproperty="ping.rc" timeout="15"/>
```

Return codes

0   Command completed successfully.

2   Command timed out.

Related reference:

"Using Apache Ant with WebSphere MQ Managed File Transfer" on page 328
WebSphere MQ Managed File Transfer provides tasks that you can use to integrate file transfer function into the Apache Ant tool.

"Ant tasks provided by WebSphere MQ Managed File Transfer" on page 899
WebSphere MQ Managed File Transfer provides a number of Ant tasks that you can use to access file transfer capabilities.

fte:uuid:

Generates a pseudo-random unique identifier and assigns it to a given property. For example, you can use this identifier to generate job names for other file transfer operations.

Attributes

`length`
Required. The numeric length of UUID to generate. This length value does not include the length of any prefix, specified by the `prefix` parameter.

`property`
Required. The name of the property to assign the generated UUID to.

`prefix`
Optional. A prefix to add to the generated UUID. This prefix is not counted as part of the length of the UUID, as specified by the `length` parameter.

Example

This example defines a UUID that starts with the letters `ABC` followed by 16 pseudo-random hex characters. The UUID is assigned to a property named `uuid.property`.

```
<fte:uuid length="16" property="uuid.property" prefix="ABC"/>
```

Related reference:

"Using Apache Ant with WebSphere MQ Managed File Transfer" on page 328
WebSphere MQ Managed File Transfer provides tasks that you can use to integrate file transfer function into the Apache Ant tool.

"Ant tasks provided by WebSphere MQ Managed File Transfer" on page 899
WebSphere MQ Managed File Transfer provides a number of Ant tasks that you can use to access file transfer capabilities.

fte:filespec:

The **fte:filespec** parameter is used as a nested element in other tasks. Use **fte:filespec** to describe a mapping between one or more source files, directories, and a destination. Typically this element is used when expressing a set of files or directories to move or copy.

Nested by:
- The fte:filecopy task
- The fte:filemove task

Source specification attributes

You must specify one of srcfilespec or srcqueue.

srcfilespec
Specifies the source of the file operation. The value of this attribute can include a wildcard.

srcqueue
Specifies the source of the transfer is a queue. The transfer moves data from messages stored on the queue specified by this attribute. You cannot specify this attribute if the **fte:filespec** task is nested within the **fte:filecopy** task.

The srcqueue attribute is not supported when the source agent is a protocol bridge agent.

Destination specification attributes

You must specify one of dstdir, dstds, dstfilespace, dstfile, dstqueue or dstpds.

dstdir
Specifies a directory as the destination for a file operation.

dstfile
Specifies a file as the destination for a file operation.

dstfilespace
Specifies a file space as the destination for a file operation.

dstqueue
Specifies a queue as the destination for a file to message operation. You can optionally include a queue manager name in this specification, using the format QUEUE@QUEUEMANAGER. If you do not specify a queue manager name the destination agent queue manager is used. You must specify a valid queue name that exists on the queue manager.

If you specify the dstqueue attribute, you cannot specify the srcqueue attributes because these attributes are mutually exclusive.
The dstqueue attribute is not supported when the destination agent is a protocol bridge agent.

Source option attributes

srcencoding
Optional. The character set encoding used by the file to transfer.

You can specify this attribute only when the conversion attribute is set to a value of `text`. If you do not specify the srcencoding attribute, the character set of the source system is used for text transfers.

srceol
Optional. The end of line delimiter used by the file being transferred. The valid values are as follows:
- `CRLF` - Use a carriage return character followed by a line-feed character as the end of line delimiter. This convention is typical for Windows systems.
- `LF` - Use a line-feed character as the end of line delimiter. This convention is typical for UNIX systems.

You can specify this attribute only when the conversion attribute is set to a value of `text`. If you do not specify the srceol attribute, text transfers automatically determine the correct value based on the operating system of the source agent.

srcmsgdelimbytes
Optional. Specifies one or more byte values to insert as the delimiter when appending multiple messages to a binary file. Each value must be specified as two hexadecimal digits in the range 00-FF, prefixed by x. Multiple bytes must be comma-separated. For example, `srcmsgdelimbytes="x08,xA4"`.

You can specify the srcmsgdelimbytes attribute only if you have also specified the srcqueue attribute. You cannot specify the srcmsgdelimbytes attribute if you have also specified the value `text` for the conversion attribute.

srcmsgdelimtext

Optional. Specifies a sequence of text to insert as the delimiter when appending multiple messages to a text file. You can include Java escape sequences for String literals in the delimiter. For example, `srcmsgdelimtext="\u007d\n"`. The text delimiter is inserted after each message by the source agent. The text delimiter is encoded to binary format using the source encoding of the transfer. Each message is read in binary format, the encoded delimiter is appended in binary format to the message, and the result is transferred in binary format to the destination agent. If the source agent code page includes shift-in and shift-out states, the agent assumes that each message is in the shift-out state at the end of the message. At the destination agent the binary data is converted in the same way as a file to file text transfer. You can only specify the srcmsgdelimtext attribute if you have also specified the srcqueue attribute and a value of `text` for the conversion attribute.

srcmsgdelimposition

Optional. Specifies the position that the text or binary delimiter is inserted into. The valid values are as follows:

- `prefix` - the delimiters are inserted into the destination file before the data from each message.
- `postfix` - the delimiters are inserted into the destination file after the data from each message.

You can specify the srcmsgdelimposition attribute only if you have also specified one of the srcmsgdelimbytes or srcmsgdelimtext attributes.

srcmsggroups

Optional. Specifies that the messages are grouped by WebSphere MQ group ID. The first complete group is written to the destination file. If this attribute is not specified, all messages on the source queue are written to the destination file. You can specify the srcmsggroups attribute only if you have also specified the srcqueue attribute.

srcqueuetimeout

Optional. Specifies the time, in seconds, to wait for one of the following conditions to be met:

- For a new message to be written to the queue.
- If the srcmsggroups attribute was specified, for a complete group to be written on the queue.

If neither of these conditions are met within the time specified by the value of srcqueuetimeout, the source agent stops reading from the queue and completes the transfer. If the srcqueuetimeout attribute is not specified, the source agent stops reading from the source queue immediately if the source queue is empty or, in the case where the srcmsggroups attribute is specified, if there is no complete group on the queue. You can specify the srcqueuetimeout attribute only if you have also specified the srcqueue attribute.

For information about setting the srcqueuetimeout value, see "Guidance for specifying a wait time on a message-to-file transfer" on page 706.

srcrecdelimpos

Optional. Specifies the position that the binary delimiter is inserted into. The valid values are as follows:

- prefix - the delimiters are inserted into the destination file before the data from each source record-oriented file record.
- postfix - the delimiters are inserted into the destination file after the data from each source record-oriented file record.

You can specify the srcrecdelimpos attribute only if you have also specified the srcrecdelimbytes attribute.

Destination option attributes

dstAttributes

Optional. Specifies a semicolon-separated list of file attributes associated with the destination files in the transfer. You can specify attributes with or without a value.

For example, attributes without a value:

```
dstAttributes="ATTRIBUTE1;ATTRIBUTE2"
```

For example, attributes with a value:

```
dstAttributes="ATTRIBUTE1(VALUE);ATTRIBUTE2(VALUE)"
```

For example, one attribute with a value and one attribute without a value:

```
dstAttributes="ATTRIBUTE1;ATTRIBUTE2(VALUE)"
```

For information about file attributes for WebSphere MQ Managed File Transfer on IBM 4690, see "File distribution attributes" on page 87.

dstencoding

Optional. The character set encoding to use for the transferred file.

You can specify this attribute only when the conversion attribute is set to a value of `text.` If the dstencoding attribute is not specified, the character set of the destination system is used for text transfers.

dsteol

Optional. The end of line delimiter to use for the transferred file. The valid values are as follows:

- CRLF - Use a carriage return character followed by a line-feed character as the end of line delimiter. This convention is typical for Windows systems.
- LF - Use a line-feed character as the end of line delimiter. This convention is typical for UNIX systems.

You can specify this attribute only when the conversion attribute is set to a value of `text.` If you do not specify the dsteol attribute, text transfers automatically determine the correct value based on the operating system of the destination agent.

dstmsgdelimbytes

Optional. Specifies the hexadecimal delimiter to use when splitting a binary file into multiple messages. All the messages have the same WebSphere MQ group ID; the last message in the group has the WebSphere MQ LAST_MSG_IN_GROUP flag set. The format for specifying a hexadecimal byte as a delimiter is xNN, where N is a character in the range 0-9 or a-f. You can specify a sequence of hexadecimal bytes as a delimiter by specifying a comma-separated list of hexadecimal bytes, for example: `x3e,x20,x20,xbf`.

You can specify the dstmsgdelimbytes attribute only if you have also specified the dstqueue attribute and the transfer is in binary mode. You can specify only one of the dstmsgsize, dstmsgdelimbytes, and dstmsgdelimpattern attributes.

dstmsgdelimpattern

Optional. Specifies the Java regular expression to use when splitting a text file into multiple messages. All the messages have the same WebSphere MQ group ID; the last message in the group has the WebSphere MQ LAST_MSG_IN_GROUP flag set. The format for specifying a regular expression as a delimiter is a regular expression enclosed in parentheses, (*regular_expression*), or enclosed in double quotation marks, "*regular_expression*". For more information, see "Regular expressions used by WebSphere MQ Managed File Transfer" on page 685.

By default, the length of the string that the regular expression can match is limited by the destination agent to five characters. You can change this behavior using the **maxDelimiterMatchLength** agent property. For more information, see "Advanced agent properties" on page 549.

You can specify the dstmsgdelimpattern attribute only if you have also specified the dstqueue attribute and the transfer is in text mode. You can specify only one of the dstmsgsize, dstmsgdelimbytes, and dstmsgdelimpattern attributes.

dstmsgdelimposition

Optional. Specifies the position that the text or binary delimiter is expected to be in. The valid values are as follows:

- `prefix` - The delimiters are expected at the beginning of each line.
- `postfix` - The delimiters are expected at the end of each line.

You can specify the dstmsgdelimposition attribute only if you have also specified the dstmsgdelimpattern attribute.

dstmsgincludedelim

Optional. Specifies whether to include the delimiter that is used to split the file into multiple messages in the messages. If the dstmsgincludedelim attribute is specified, the delimiter is included at the end of the message that contains the file data preceding the delimiter. By default the delimiter is not included in the messages. You can specify the dstmsgincludedelim attribute only if you have also specified one of the dstmsgdelimpattern and dstmsgdelimbytes attributes.

dstmsgpersist

Optional. Specifies whether messages written to the destination queue are persistent. The valid values are as follows:

- `true` - Write persistent messages to the destination queue. This is the default value.
- `false` - Write non-persistent messages to the destination queue.
- `qdef` - The persistence value is taken from the DefPersistence attribute of the destination queue.

You can specify this attribute only when the dstqueue attribute is also specified.

dstmsgprops

Optional. Specifies whether the first message written to the destination queue by the transfer has WebSphere MQ message properties set. Possible values are:

- `true` - Set message properties on the first message created by the transfer.
- `false` - Do not set message properties on the first message created by the transfer. This is the default value.

For more information, see "WebSphere MQ message properties set on messages written to destination queues" on page 701.

You can specify this attribute only when the dstqueue attribute is also specified.

dstmsgsize

Optional. Specifies whether to split the file into multiple fixed-length messages. All of the messages have the same WebSphere MQ group ID; the last message in the group has the WebSphere MQ LAST_MSG_IN_GROUP flag set. The size of the messages is specified by the value of dstmsgsize. The format of dstmsgsize is *<length><units>*, where *length* is a positive integer value and *units* is one of the following values:

- `B` - Bytes. The minimum value allowed is two times the maximum bytes-per-character value of the code page of the destination messages.
- `K` - Kibibytes. This is equivalent to 1024 bytes.
- `M` - Mebibytes. This is equivalent to 1024 kibibytes.

If the file is transferred in text mode, and is in a double-byte character set or multibyte character set, the file is split into messages on the closest character boundary to the specified message size.

You can specify the dstmsgsize attribute only if you have also specified the dstqueue attribute. You can specify only one of the dstmsgsize, dstmsgdelimbytes, and dstmsgdelimpattern attributes.

dstunsupportedcodepage

Optional. Specifies the action to take if the destination queue manager, as specified by the dstqueue attribute, does not support the code page used when transferring file data to a queue as a text transfer. The valid values for this attribute are as follows:

- `binary` – continue the transfer but do not apply code page conversion to the data being transferred. Specifying this value is equivalent to not setting the conversion attribute to `text`.
- `fail` – do not continue with the transfer operation. The file is recorded as having failed to transfer. This is the default.

You can only specify the dstunsupportedcodepage attribute if you have also specified the dstqueue attribute and a value of `text` for the conversion attribute.

Optional. Specifies that destination records longer than the LRECL data set attribute are truncated. If set to true, the records are truncated. If set to false, the records are wrapped. The default setting is false. This parameter is valid only for text mode transfers where the destination is a data set.

Other attributes

checksum

Optional. Determines the algorithm used to checksum transferred files.

- `MD5` - use the MD5 hashing algorithm.
- `NONE` - do not use a checksum algorithm.

If you do not specify the checksum attribute, a default value of `MD5` is used.

conversion

Optional. Specifies the type of conversion to apply to the file as it is being transferred. Possible values are:

- `binary` - apply no conversion.
- `text` - apply code page conversion between the source and destination systems. Also apply conversion of line delimiters. The `srcencoding`, `dstencoding`, `srceol` and `dsteol` attributes influence the conversion that is applied.

If you do not specify the conversion attribute, a default value of `binary` is specified.

overwrite

Optional. Determines whether an existing destination file can be overwritten by the operation. When you specify a value of `true`, any existing destination file are overwritten. When you specify a value of `false`, the existence of a duplicate file at the destination results in the operation failing. If the overwrite attribute is not specified, a default value of `false` is specified.

recurse

Optional. Determines whether the file transfer recurses into subdirectories. When you specify a value of `true`, the transfer recurses into subdirectories. When you specify a value of `false`, the transfer does not recurse into subdirectories. If the recurse attribute is not specified, a default value of `false` is specified.

Example

This example specifies a fte:filespec with a source file of `file1.bin` and a destination file of `file2.bin`.

```
<fte:filespec srcfilespec="/home/fteuser/file1.bin" dstfile="/home/fteuser/file2.bin"/>
```

fte:metadata:

Metadata is used to carry additional user-defined information with a file transfer operation.

See "Metadata for user exit routines" on page 921 for more information about how WebSphere MQ Managed File Transfer uses metadata.

Nested by:
- The fte:filecopy task
- The fte:filemove task
- The fte:call task

Parameters specified as nested elements

`fte:entry`
> You must specify at least one entry inside the fte:metadata nested element. You can choose to specify more than one entry. Entries associate a key name with a value. Keys must be unique in a block of fte:metadata

Entry attributes

`name`
> Required. The name of the key belonging to this entry. This name must be unique across all `entry` parameters nested inside a fte:metadata element.

`value`
> Required. The value to assign to this entry.

Example

This example shows a fte:metadata definition that contains two entries.

```
<fte:metadata>
 <fte:entry name="org.foo.partColor" value="red"/>
 <fte:entry name="org.foo.partSize" value="medium"/>
</fte:metadata>
```

Related reference:

"Using Apache Ant with WebSphere MQ Managed File Transfer" on page 328
WebSphere MQ Managed File Transfer provides tasks that you can use to integrate file transfer function into the Apache Ant tool.

"Ant tasks provided by WebSphere MQ Managed File Transfer" on page 899
WebSphere MQ Managed File Transfer provides a number of Ant tasks that you can use to access file transfer capabilities.

Program invocation nested elements:

Programs can be started using one of five nested elements: `fte:presrc`, `fte:predst`, `fte:postdst`, `fte:postsrc`, and `fte:command`. These nested elements instruct an agent to call an external program as

part of its processing. Before you can start a program, you must ensure that the command is in a location specified by the commandPath property in the `agent.properties` file of the agent that runs the command.

Even though each program invocation element has a different name, they share the same set of attributes and the same set of nested elements. Programs can be started by the **fte:filecopy**, **fte:filemove**, and **fte:command** Ant tasks. If you have configured a Web Gateway to allow files to be uploaded to an agent, configure `fte:postdst` program invocations by specifying the `x-fte-postdest` header or using the `postdest` form field in the HTTP request.

You cannot invoke programs from a Connect:Direct bridge agent.

Ant tasks that can invoke programs:
- The fte:filecopy task nests program invocation parameters using the `fte:predst`, `fte:postdst`, `fte:presrc`, and `fte:postsrc` nested elements.
- The fte:filemove task nests program invocation parameters using the `fte:predst`, `fte:postdst`, `fte:presrc`, and `fte:postsrc` nested elements.
- The fte:call task nests program invocation parameters using the `fte:command` nested element.

Attributes

`command`
> Required. Names the program to call. For the agent to be able to run a command, the command must be in a location specified by the commandPath property in the agent's `agent.properties` file. For more information, see "The commandPath property" on page 414. Any path information specified in the `command` attribute is considered relative to a location specified by the commandPath property. When `type` is `executable`, an executable program is expected otherwise a script appropriate for the call type is expected.

`retrycount`
> Optional. The number of times to retry calling the program if the program does not return a success return code. The program named by the `command` attribute is called up to this number of times. The value assigned to this attribute must be non-negative. If you do not specify the `retrycount` attribute, a default value of zero is used.

`retrywait`
> Optional. The time to wait, in seconds, before trying the program invocation again. If the program named by the `command` attribute does not return a success return code and the `retrycount` attribute specifies a non-zero value, this parameter determines the time to wait between retries. The value assigned to this attribute must be non-negative. If you do not specify the `retrywait` attribute, a default value of zero is used.

`successrc`
> Optional. The value of this attribute is used to determine when the program invocation successfully runs. The process return code for the command is evaluated using this expression. The value can be composed of one or more expressions combined with a vertical bar character (|) to signify Boolean OR, or an ampersand (&) character to signify Boolean AND. Each expression can be one of the following types of expression:
> - A number to indicate an equality test between the process return code and the number.
> - A number prefixed with a ">" character to indicate a greater-than test between the number and the process return code.
> - A number prefixed with a "<" character to indicate a less-than test between the number and the process return code.
> - A number prefixed with a "!" character to indicate a not-equal-to test between the number and the process return code.

For example: `>2&<7&!5|0|14` is interpreted as the following return codes being successful: 0, 3, 4, 6, 14. All other return codes are interpreted as being unsuccessful. If you do not specify the `successrc` attribute, a default value of zero is used. This means that the command is judged to have successfully run if, and only if, it returns a code of zero.

priority

> Optional (os4690background only). The priority level to assign to a background task on an IBM 4690 system. Default value is 5 and valid values are within the range 1 - 9.

message

> Optional (os4690background only). The status message to display on an IBM 4690 system background control screen for the executed command.

type

> Optional. The value of this attribute specifies what type of program is being called. Specify one of the following options:

> **executable**
>> The task calls an executable program. Can have additional arguments specified using the arg nested element. The program is expected to be accessible on the commandPath and where applicable have execute permission set. UNIX scripts can be called as long as they specify a shell program (for example, first line of shell script file is: `#!/bin/sh`). Command output written to stderr or stdout is sent to the WebSphere MQ File Transfer Edition log for the call. However, the amount of data output is limited by the agent configuration. The default is 10K bytes of data, but you can override this default using the agent property: maxCommandOutput.

> **antscript**
>> The task runs the specified Ant script, using the **fteAnt** command. Properties can be specified using the property nested element. Ant targets can be specified using the target nested element. The Ant script is expected to be accessible on the commandPath. Ant output written to stderr or stdout is sent to the WebSphere MQ Managed File Transfer log for the call. However, the amount of data output is limited by the agent configuration. The default is 10K bytes of data but you can override this default using the agent property: maxCommandOutput.

> **os4690background**
>> The task calls an OS4690BACKGROUND program. Transfer program calls on an IBM® 4690 system can be run in background. These call types support the running of native IBM 4690 applications and batch command scripts. If you run a batch script the COMMAND.286 application is used with the -C option as the application, passing the batch script and any specified arguments as the parameters.

Parameters specified as nested elements

fte:arg

> Only valid where the value of the `type` attribute is executable. Use nested `fte:arg` elements to specify arguments to the program that is being called as part of the program invocation. The program arguments are built from the values specified by the `fte:arg` elements in the order that the `fte:arg` elements are encountered. You can choose to specify zero or more `fte:arg` elements as nested elements of a program invocation.

fte:property

> Only valid where the value of the `type` attribute is antscript. Use the `name` and `value` attributes of the nested `fte:property` elements to pass in name-value pairs to the Ant script. You can choose to specify zero or more `fte:property` elements as nested elements of a program invocation.

fte:target
> Only valid where the value of the `type` attribute is antscript. Specify a target in the Ant script to call. You can choose to specify zero or more `fte:target` elements as nested elements of a program invocation.

Arg attributes

value
> Required. The value of the argument to pass to the program being called.

Property attributes

name
> Required. The name of a property to pass to the Ant script.

value
> Required. The value to associate with the property name being passed to the Ant script.

Examples

This example shows an `fte:postsrc` program invocation being specified as part of an fte:filecopy task. The program invocation is for a program called `post.sh` and is supplied a single argument of `/home/fteuser2/file.bin`.

```
<fte:filecopy cmdqm="qm0@localhost@1414@SYSTEM.DEF.SVRCONN"
              src="agent1@qm1" dst="agent2@qm2"
              rcproperty="copy.result">
  <fte:filespec srcfilespec="/home/fteuser1/file.bin" dstfile="/home/fteuser2/file.bin"/>

  <fte:postsrc command="post.sh" successrc="1" >
    <fte:arg value="/home/fteuser2/file.bin"/>
  </fte:postsrc>

</fte:filecopy>
```

This example shows an `fte:command` program invocation being specified as part of a fte:call task. The program invocation is for an executable called `command.sh`, which is not passed any command-line arguments. If `command.sh` does not return a success return code of 1, the command is tried again after 30 seconds.

```
<fte:call cmdqm="qm0@localhost@1414@SYSTEM.DEF.SVRCONN"
          agent="agent1@qm1"
          rcproperty="call.rc"
          origuser="bob"
          jobname="${job.id}">
  <fte:command command="command.sh" successrc="1" retrycount="5" retrywait="30"/>
</fte:call>
```

This example shows an `fte:command` program invocation being specified as part of a fte:call task. The program invocation is for the copy and compress targets in an Ant script called `script.xml`, which is passed two properties.

```
<fte:call cmdqm="qm0@localhost@1414@SYSTEM.DEF.SVRCONN"
          agent="agent1@qm1"
          rcproperty="call.rc"
          origuser="bob"
          jobname="${job.id}">
  <fte:command command="script.xml" type="antscript">
        <property name="src" value="AGENT5@QM5"/>
        <property name="dst" value="AGENT3@QM3"/>
        <target name="copy"/>
        <target name="compress"/>
    </fte:command>
</fte:call>
```

Related concepts:

"Specifying programs to run" on page 277
You can run programs on a system where a IBM WebSphere MQ Managed File Transfer agent is running. As part of a file transfer request, you can specify a program to run either before a transfer starts, or after it finishes. Additionally, you can start a program that is not part of a file transfer request by submitting a managed call request.

Related reference:

"Using Apache Ant with WebSphere MQ Managed File Transfer" on page 328
WebSphere MQ Managed File Transfer provides tasks that you can use to integrate file transfer function into the Apache Ant tool.

"Ant tasks provided by WebSphere MQ Managed File Transfer" on page 899
WebSphere MQ Managed File Transfer provides a number of Ant tasks that you can use to access file transfer capabilities.

Working with user exits for customization

Metadata for user exit routines

There are three different types of metadata that can be supplied to user exit routines for WebSphere MQ Managed File Transfer: environment, transfer, and file metadata. This metadata is presented as maps of Java key-value pairs.

Environment metadata

Environment metadata is passed to all user exit routines and describes the agent runtime environment that the user exit routine is being called from. This metadata is read-only and cannot be updated by any user exit routine.

Table 58. Environment metadata

Key	Description
AGENT_CONFIGURATION_DIRECTORY_KEY	The name of the directory that contains the agent's configuration information.
AGENT_PRODUCT_DIRECTORY_KEY	The name of the directory that the agent code has been installed in.
AGENT_VERSION_KEY	Version number for the agent runtime that calls the exit routine.

The key names and value names given in Table 1 are constants that are defined in the EnvironmentMetaDataConstants interface.

Transfer metadata

Transfer metadata is passed to all user exit routines. The metadata consists of system-supplied values and user-supplied values. If you change any system-supplied values, these changes are ignored. The initial user-supplied values for the source transfer start user exit are based on those values you supply when you define the transfer. The source agent can change user-supplied values as part of the processing of the source transfer start user exit. This user exit is called before the entire file transfer starts. These changes are used in subsequent calls to other exit routines that relate to that transfer. Transfer metadata is applied to an entire transfer.

Although all user exits can read values from the transfer metadata, only the source transfer start user exit can change transfer metadata

You cannot use transfer metadata to propagate information between different file transfers.

The system-supplied transfer metadata is detailed in Table 2:

Table 59. Transfer metadata

Key	Description
DESTINATION_AGENT_KEY	The name of the agent that is the destination for the transfer.
JOB_NAME_KEY	The job name associated with the transfer request
MQMD_USER_KEY	The MQMD user field from the message used to submit the transfer request
ORIGINATING_HOST_KEY	The host name specified as the originating host name in the transfer request
ORIGINATING_USER_KEY	The user name specified as the originating user ID in the transfer request
SOURCE_AGENT_KEY	The name of the agent that is the source of the transfer
TRANSFER_ID_KEY	The identifier of the transfer

The key names and value names given in Table 2 are constants that are defined in the TransferMetaDataConstants interface.

File metadata

The file metadata is passed to the source transfer start exit as part of the file specification. There is separate file metadata for the source and destination files.

You cannot use file metadata to propagate information between different file transfers.

Table 60. File metadata

Key	Permitted values	Description
▶V 7.5.0.2 CONVERT_LINE_SEPARATORS		Key value used for text transfers to indicate whether CRLF (carriage return-line feed) or LF (line feed) line separator sequences in source data are converted to the line separator sequence at the destination.
▶V 7.5.0.2 DELIMITER_KEY		Key value used to define a delimiter to separate record data when transferring record-oriented data to normal files.
▶V 7.5.0.2 DELIMITER_POSITION_KEY	DELIMITER_POSITION_PREFIX_VALUE DELIMITER_POSITION_POSTFIX_VALUE	Use with the DELIMITER_KEY to define the position of the delimiter; either prefix or postfix.
▶V 7.5.0.2 DELIMITER_TYPE_KEY	DELIMITER_TYPE_BINARY_VALUE DELIMITER_TYPE_TEXT_VALUE DELIMITER_TYPE_SIZE_VALUE	Use with the DELIMITER_KEY to define the type of delimiter.
DESTINATION_EXIST_KEY	DESTINATION_EXIST_KEY_ERROR_VALUE DESTINATION_EXIST_KEY_OVERWRITE_VALUE	Determines the file transfer behavior if the destination file exists.
▶V 7.5.0.2 FILE_ALIAS_KEY		Key value used to define an alias for the file being transferred.
FILE_CHECKSUM_METHOD_KEY	FILE_CHECKSUM_METHOD_NONE_VALUE FILE_CHECKSUM_METHOD_MD5_VALUE	Determines the checksum method to use when transferring the file.
FILE_CONVERSION_KEY	FILE_CONVERSION_TEXT_VALUE FILE_CONVERSION_BINARY_VALUE	Determines the type of conversion applied to the file contents.
FILE_ENCODING_KEY		Determines the encoding used for a text file.

Table 60. File metadata (continued)

Key	Permitted values	Description
FILE_END_OF_LINE_KEY	FILE_END_OF_LINE_LF_VALUE FILE_END_OF_LINE_CRLF_VALUE	Determines the character sequence that denotes the end of a line: \<LF\> or \<CR\>\<LF\>.
FILE_SPACE_ALIAS		Determines the alias of a file in the file space. **Note:** This metadata can be used only if the FILE_TYPE_KEY is FILE_TYPE_FILE_SPACE_VALUE
FILE_SPACE_NAME		Determines the name of the file space. **Note:** This metadata can be used only if the FILE_TYPE_KEY is FILE_TYPE_FILE_SPACE_VALUE
FILE_TYPE_KEY	FILE_TYPE_FILE_VALUE FILE_TYPE_DIRECTORY_VALUE FILE_TYPE_DATASET_VALUE FILE_TYPE_PDS_VALUE FILE_TYPE_QUEUE_VALUE FILE_TYPE_FILE_SPACE_VALUE	Determines the destination file, queue, or file space specification.
►V7.5.0.2 INSERT_RECORD_LINE_SEPARATOR_KEY		Key value used for text transfers from record-oriented files to specify whether line separators are inserted into the data after each record.
►V7.5.0.2 KEEP_TRAILING_SPACES_KEY	KEEP_TRAILING_SPACES_TRUE_VALUE KEEP_TRAILING_SPACES_FALSE_VALUE	Key value used to determine whether trailing spaces are removed from records read from fixed-length-format data sets.
►V7.5.0.2 NEW_RECORD_ON_LINE_SEPARATOR_KEY		Key value used for text transfers to record- oriented files to specify whether line separators in the data are included with the record data or cause a new record (and are not written).

The key names and value names given in Table 3 are constants that are defined in the FileMetaDataConstants interface.

Resource monitor user exits

Resource monitor user exits allow you to configure custom code to run when a monitor's trigger condition is satisfied, before the associated task is started.

It is not recommended to invoke new transfers directly from user exit code. In some circumstances this causes files to be transferred multiple times as user exits are not resilient to agent restarts.

Resource monitor user exits use the existing infrastructure for user exits. The monitor user exits are called after a monitor has triggered but before the corresponding task has been run by the monitor's task. This allows the user exit to modify the task to be run and decide whether a task should proceed or not. You can modify the monitor task by updating the monitor metadata, which is then used for variable substitution in the task document created by the creation of the original monitor. Alternatively, the monitor exit can replace or update the task definition XML string passed as a parameter. The monitor exit can return a result code of either 'proceed' or 'cancel' for the task. If cancel is returned, the task will not be started and the monitor will not start again until the monitored resource matches the trigger conditions. If the resource has not changed, the trigger will not start. As with the other user exits, you can chain monitor exits together. If one of the exits returns a cancel result code, the overall result is cancel and the task is not started.

- A map of environment metadata (same as other user exits)

- A map of monitor metadata including immutable system metadata and mutable user metadata. The immutable system metadata is as follows:
  - FILENAME - name of the file that satisfied the trigger condition
  - FILEPATH - path to the file that satisfied the trigger condition
  - FILESIZE (in bytes - this metadata might not be present) - size of the file that satisfied the trigger condition
  - LASTMODIFIEDDATE (Local) - date that the file that satisfied the trigger condition was last changed. This date is expressed as the local date of the time zone the agent is running in and is formatted as an ISO 8601 date.
  - LASTMODIFIEDTIME (Local) - time in local format that the file that satisfied the trigger condition was last changed. This time is expressed as the local time of the time zone the agent is running in and is formatted as an ISO 8601 time.
  - LASTMODIFIEDDATEUTC - date in universal format that the file that satisfied the trigger condition was last changed. This date is expressed as the local date converted to the UTC time zone and is formatted as an ISO 8601 date.
  - LASTMODIFIEDTIMEUTC - time in universal format that the file that satisfied the trigger condition was last changed. This time is expressed the local time converted to the UTC time zone and is formatted as an ISO 8601 time.
  - AGENTNAME - the monitor agent name
- An XML string representing the task to be run as a result of the monitor trigger.

Monitor exits return the following data:
- An indicator that specifies whether to progress further (proceed or cancel)
- A string to insert into the trigger-satisfied log message

As a result of running the monitor exit code, the monitor metadata and task definition XML string that were originally passed as parameters might also have been updated.

The value of the agent property monitorExitClasses (in the `agent.properties` file) specifies which monitor exit classes to load, with each exit class separated by a comma. For example:

`monitorExitClasses=testExits.TestExit1,testExits.testExit2`

The interface to the monitor user exit is:

```
package com.ibm.wmqfte.exitroutine.api;

import java.util.Map;

/**
 * An interface that is implemented by classes that want to be invoked as part of
 * user exit routine processing.  This interface defines a method that will be
 * invoked immediately prior to starting a task as the result of a monitor trigger
 */
public interface MonitorExit {

 /**
  * Invoked immediately prior to starting a task as the result of a monitor
  * trigger.
  *
  * @param environmentMetaData
  *          meta data about the environment in which the implementation
  *          of this method is running.  This information can only be read,
  *          it cannot be updated by the implementation.  The constant
  *          defined in <code>EnvironmentMetaDataConstants</code> class can
  *          be used to access the data held by this map.
  *
  * @param monitorMetaData
  *          meta data to associate with the monitor. The meta data passed
```

```
 *              to this method can be altered, and the changes will be
 *              reflected in subsequent exit routine invocations. This map
 *              also contains keys with IBM reserved names.  These entries are
 *              defined in the <code>MonitorMetaDataConstants</code> class and
 *              have special semantics. The the values of the IBM reserved names
 *              cannot be modified by the exit
 *
 * @param taskDetails
 *              An XML String representing the task to be executed as a result of
 *              the monitor triggering. This XML string may be modified by the
 *              exit
 *
 * @return     a monitor exit result object which is used to determine if the
 *              task should proceed, or be cancelled.
 */
 MonitorExitResult onMonitor(Map<String, String> environmentMetaData,
                     Map<String, String> monitorMetaData,
                   Reference<String> taskDetails);
}
```

The constants for the IBM-reserved values in the monitor metadata are as follows:

```
package com.ibm.wmqfte.exitroutine.api;

/**
 * Constants for IBM reserved values placed into the monitor meta data
 * maps used by the monitor exit routines.
 */
public interface MonitorMetaDataConstants {

/**
 * The value associated with this key is the name of the trigger
 * file associated with the monitor. Any modification performed
 * to this property by user exit routines will be ignored.
 */
final String FILE_NAME_KEY = "FILENAME";

/**
 * The value associated with this key is the path to the trigger
 * file associated with the monitor. Any modification performed
 * to this property by user exit routines will be ignored.
 */
final String FILE_PATH_KEY = "FILEPATH";

/**
 * The value associated with this key is the size of the trigger
 * file associated with the monitor. This will not be present in
 * the cases where the size cannot be determined. Any modification
 * performed to this property by user exit routines will be ignored.
 */
final String FILE_SIZE_KEY = "FILESIZE";

/**
 * The value associated with this key is the local date on which
 * the trigger file associated with the monitor was last modified.
 * Any modification performed to this property by user exit routines
 * will be ignored.
 */
final String LAST_MODIFIED_DATE_KEY = "LASTMODIFIEDDATE";

/**
 * The value associated with this key is the local time at which
 * the trigger file associated with the monitor was last modified.
 * Any modification performed to this property by user exit routines
 * will be ignored.
 */
final String LAST_MODIFIED_TIME_KEY = "LASTMODIFIEDTIME";
```

```
/**
* The value associated with this key is the UTC date on which
* the trigger file associated with the monitor was last modified.
* Any modification performed to this property by user exit routines
* will be ignored.
*/
final String LAST_MODIFIED_DATE_KEY_UTC = "LASTMODIFIEDDATEUTC";

/**
* The value associated with this key is the UTC time at which
* the trigger file associated with the monitor was last modified.
* Any modification performed to this property by user exit routines
* will be ignored.
*/
final String LAST_MODIFIED_TIME_KEY_UTC = "LASTMODIFIEDTIMEUTC";

/**
* The value associated with this key is the name of the agent on which
* the monitor is running. Any modification performed to this property by
* user exit routines will be ignored.
*/
final String MONITOR_AGENT_KEY = "AGENTNAME";

}
```

Example monitor user exit

This example class implements the MonitorExit interface. This example adds a custom substitution
variable into the monitor metadata called *REDIRECTEDAGENT* that will be populated with a value of
LONDON if the hour of the day is odd, and a value of PARIS for even hours. The monitor exit result code is
set to always return proceed.

```
package com.ibm.wmqfte.monitor;

import java.util.Calendar;
import java.util.Map;

import com.ibm.wmqfte.exitroutine.api.MonitorExit;
import com.ibm.wmqfte.exitroutine.api.MonitorExitResult;
import com.ibm.wmqfte.exitroutine.api.Reference;

/**
 * Example resource monitor user exit that changes the monitor mutable
 * metadata value between 'LONDON' and 'PARIS' depending on the hour of the day.
 *
 */
public class TestMonitorExit implements MonitorExit {

 // custom variable that will substitute destination agent
 final static String REDIRECTED_AGENT = "REDIRECTEDAGENT";

 public MonitorExitResult onMonitor(
Map<String, String> environmentMetaData,
   Map<String, String> monitorMetaData,
Reference<String> taskDetails) {

  // always succeed
  final MonitorExitResult result = MonitorExitResult.PROCEED_RESULT;

  final int hour = Calendar.getInstance().get(Calendar.HOUR_OF_DAY);

  if (hour%2 == 1) {
   monitorMetaData.put(REDIRECTED_AGENT, "LONDON");
  } else {
   monitorMetaData.put(REDIRECTED_AGENT, "PARIS");
```

```
  }

  return result;
 }
}
```

The corresponding task for a monitor that makes use of the *REDIRECTEDAGENT* substitution variable could look similar to the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<request version="4.00"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
  <managedTransfer>
    <originator>
        <hostName>reportserver.com</hostName>
        <userID>USER1</userID>
    </originator>
    <sourceAgent agent="AGENT1"
                 QMgr="QM1"/>
    <destinationAgent agent="${REDIRECTEDAGENT}"
                      QMgr="QM2"/>
    <transferSet>
      <item mode="binary" checksumMethod="MD5">
        <source recursive="false" disposition="delete">
          <file>c:\sourcefiles\reports.doc</file>
        </source>
        <destination type="file" exist="overwrite">
          <file>c:\destinationfiles\reports.doc</file>
        </destination>
      </item>
    </transferSet>
  </managedTransfer>
</request>
```

Before this transfer is started, the value of the <destinationAgent> element's `agent` attribute is replaced with either `LONDON` or `PARIS`.

You must specify the substitution variable in the monitor exit class and the task definition XML in uppercase.

Related concepts:

"Customizing WebSphere MQ Managed File Transfer with user exit routines" on page 330
You can customize the features of WebSphere MQ Managed File Transfer by using your own programs known as user exit routines.

"Metadata for user exit routines" on page 921
There are three different types of metadata that can be supplied to user exit routines for WebSphere MQ Managed File Transfer: environment, transfer, and file metadata. This metadata is presented as maps of Java key-value pairs.

"Java interfaces for user exit routines" on page 930
Use the topics in this section for reference information about Java interfaces for user exit routines.

Related reference:

"Agent properties for user exits"
In addition to the standard properties in the `agent.properties` file, there are several advanced properties specifically for user exit routines. These properties are not included by default so if you want to use any of them, you must manually edit the `agent.properties` file. If you make a change to `agent.properties` file while that agent is running, stop and restart the agent to pick up the changes.

Agent properties for user exits

In addition to the standard properties in the `agent.properties` file, there are several advanced properties specifically for user exit routines. These properties are not included by default so if you want to use any

of them, you must manually edit the `agent.properties` file. If you make a change to `agent.properties` file while that agent is running, stop and restart the agent to pick up the changes.

For WebSphere MQ V7.5, there is the ability for environment variables to be used in some Managed File Transfer properties that represent file or directory locations. This allows the locations of files or directories used when running parts of the product, to vary depending on environment changes, such as which user is running the process. For more information, see "Environment variables in WebSphere MQ Managed File Transfer properties" on page 537.

The user exit routines are called in the order listed.

Table 61. Agent properties for user exits

Property name	Description
sourceTransferEndExitClasses	Specifies a comma-separated list of classes that implement a source transfer end exit routine.
sourceTransferStartExitClasses	Specifies a comma-separated list of classes that implement a source transfer start exit routine.
destinationTransferStartExitClasses	Specifies a comma-separated list of classes that implement a destination transfer start user exit routine.
destinationTransferEndExitClasses	Specifies a comma-separated list of classes that implement a destination transfer end user exit routine.
exitClassPath	Specifies a platform-specific, character-delimited list of directories that act as the class path for user exit routines.
	The agent's exit directory is searched before any entries in this class path.
	If you are using this property on Windows, use a forward slash character (/) as a path delimiter, not the backslash character (\). For example:
	`exitClassPath=C:/IBM/MQ/Java/lib/com.ibm.mqjms.jar; C:/IBM/MQ/Java/lib/com.ibm.mq.jar`
	For WebSphere MQ V7.5, the value of this property can contain environment variables.
	For a IBM 4690 system this property can contain logical names, but the directory must be located on the `F:` drive.
exitNativeLibraryPath	Specifies a platform-specific, character-delimited list of directories that act as the native library path for user exit routines.
	For WebSphere MQ V7.5, the value of this property can contain environment variables.
monitorExitClasses	Specifies a comma-separated list of classes that implement a monitor exit routine. For more information, see "Resource monitor user exits" on page 923.
protocolBridgeCredentialExitClasses	Specifies a comma-separated list of classes that implement a protocol bridge credential user exit routine. For more information, see "Mapping credentials for a file server using exit classes" on page 259.
protocolBridgePropertiesExitClasses	Specifies a comma-separated list of classes that implement a protocol bridge server properties user exit routine.For more information, see "Looking up protocol file server properties by using exit classes (ProtocolBridgePropertiesExit2)" on page 253.
IOExitClasses	Specifies a comma-separated list of classes that implement an I/O user exit routine. List only the classes that implement the IOExit interface, that is, do not list classes that implement the other I/O user exit interfaces, for example IOExitResourcePath and IOExitChannel. For more information, see "Using WebSphere MQ Managed File Transfer transfer I/O user exits" on page 333.

Order of exit invocation

The source and destination exits are invoked in the following order:

1. SourceTransferStartExit
2. DestinationTransferStartExit
3. DestinationTransferEndExit
4. SourceTransferEndExit

Chaining source and destination exits

If you specify multiple exits, the first exit in the list is invoked first, followed by the second exit, and so on. Any changes made by the first exit are passed as input to the exit that is subsequently invoked and so on. For example, if the there are two source transfer start exits any changes made to the transfer metadata by the first exit are input to the second exit. Each exit returns its own result. If all the exits of a given type return PROCEED as a transfer result code, the overall result is PROCEED. If one or more exits return CANCEL_TRANSFER, the overall result is CANCEL_TRANSFER. All of the result codes and strings returned by the exits are output in the transfer log.

If the overall result from the source transfer start exit is PROCEED, the transfer proceeds using any changes made by the exits. If the overall result is CANCEL_TRANSFER, the source transfer end exits are invoked and then the transfer is canceled. The completion status in the transfer log is "cancelled".

If the overall result from the destination transfer start exits is PROCEED, the transfer proceeds using any changes made by the exits. If the overall result is CANCEL_TRANSFER, the destination transfer end exits are invoked, then the source transfer end exits are invoked. Finally the transfer is canceled. The completion status in the transfer log is "cancelled".

If a source or destination exit needs to pass information to following exits either in the chain or in the order of execution it must be done by updating the transfer metadata. The usage of the transfer metadata is exit implementation specific. For instance, if an exit sets the return result to CANCEL_TRANSFER and needs to communicate to the following exits that the transfer has been canceled it must done by setting a transfer metadata value in a way understood by the other exits.

Example

```
sourceTransferStartExitClasses=com.ibm.wmqfte.test.MFTTestSourceTransferStartExit
sourceTransferEndExitClasses=com.ibm.wmqfte.test.MFTTestSourceTransferEndExit
destinationTransferStartExitClasses=com.ibm.wmqfte.test.MFTTestDestinationTransferStartExit
destinationTransferEndExitClasses=com.ibm.wmqfte.test.MFTTestDestinationTransferEndExit
exitClassPath=C:/IBM/MQ/Java/lib/com.ibm.mqjms.jar;C:/IBM/MQ/Java/lib/com.ibm.mq.jar
```

Related concepts:

"Customizing WebSphere MQ Managed File Transfer with user exit routines" on page 330
You can customize the features of WebSphere MQ Managed File Transfer by using your own programs known as user exit routines.

"Metadata for user exit routines" on page 921
There are three different types of metadata that can be supplied to user exit routines for WebSphere MQ Managed File Transfer: environment, transfer, and file metadata. This metadata is presented as maps of Java key-value pairs.

"Java interfaces for user exit routines" on page 930
Use the topics in this section for reference information about Java interfaces for user exit routines.

Related reference:

"Resource monitor user exits" on page 923
Resource monitor user exits allow you to configure custom code to run when a monitor's trigger condition is satisfied, before the associated task is started.

In WebSphere MQ V7.5, it is possible for environment variables to be used in Managed File Transfer properties that represent file or directory locations. This allows the locations of files or directories used when running parts of the product, to vary depending on environment changes. For example, which user is running the process.

Java interfaces for user exit routines

Use the topics in this section for reference information about Java interfaces for user exit routines.

CDCredentialExit.java interface:
CDCredentialExit.java

```
/*
 *   Licensed Materials - Property of IBM
 *
 *   "Restricted Materials of IBM"
 *
 *   5724-H72
 *
 *   △ Copyright IBM Corp. 2011 All Rights Reserved.
 *
 *   US Government Users Restricted Rights - Use, duplication or
 *   disclosure restricted by GSA ADP Schedule Contract with
 *   IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;

import java.util.Map;

/**
 * An interface that is implemented by classes that are invoked as part of
 * user exit routine processing.  This interface defines methods that are
 * invoked by a Connect:Direct bridge agent to map the WebSphere MQ user ID of the transfer to credentials
 * that are used to access the Connect:Direct node.
 * There will be one instance of each implementation class per Connect:Direct bridge agent. The methods
 * can be called from different threads so the methods must be synchronized.
 */
public interface CDCredentialExit {

 /**
  * Invoked once when a Connect:Direct bridge agent is started. It is intended to initialize
  * any resources that are required by the exit
  *
  * @param bridgeProperties
  *             The values of properties defined for the Connect:Direct bridge.
  *             These values can only be read, they cannot be updated by
  *             the implementation.
  *
  * @return     true if the initialisation is successful and false if unsuccessful
  *             If false is returned from an exit the Connect:Direct bridge agent does not
  *             start.
  */
 public boolean initialize(final Map<String, String> bridgeProperties);

 /**
  * Invoked once per transfer to map the WebSphere MQ user ID in the transfer message to the
  * credentials to be used to access the Connect:Direct node.
  *
  * @param mqUserId The WebSphere MQ user ID from which to map to the credentials to be used
  *             to access the Connect:Direct node
 * @param snode     The name of the Connect:Direct SNODE specified as the cdNode in the
  *             file path. This is used to map the correct user ID and password for the
  *              SNODE.
 * @return         A credential exit result object that contains the result of the map and
  *                 the credentials to use to access the Connect:Direct node
  */

 public CDCredentialExitResult mapMQUserId(final String mqUserId, final String snode);

 /**
 * Invoked once when a Connect:Direct bridge agent is shutdown. This method releases
 * any resources that were allocated by the exit
 *
  * @param bridgeProperties
 *             The values of properties defined for the Connect:Direct bridge.
```

```
 *           These values can only be read, they cannot be updated by
  *            the implementation.
  *
  * @return
*/
  public void shutdown(final Map<String, String> bridgeProperties);    }
```

DestinationTransferEndExit.java interface:

DestinationTransferEndExit.java

```
package com.ibm.wmqfte.exitpoint.api;


/**
 * An interface that is implemented by classes that want to be invoked as part of
 * user exit routine processing.  This interface defines a method that will be
 * invoked immediately after completing a transfer on the agent acting as the
 * destination of the transfer.
 */
public interface DestinationTransferEndExit {

 /**
  * Invoked immediately after the completion of a transfer on the agent acting as
  * the destination of the transfer.
  *
  * @param transferExitResult
  *          a result object reflecting whether or not the transfer completed
  *          successfully.
  *
  * @param sourceAgentName
  *          the name of the agent acting as the source of the transfer.
  *
  * @param destinationAgentName
  *          the name of the agent acting as the destination of the
  *          transfer.  This is the name of the agent that the
  *          implementation of this method will be invoked from.
  *
  * @param environmentMetaData
  *          meta data about the environment in which the implementation
  *          of this method is running.  This information can only be read,
  *          it cannot be updated by the implementation.  The constants
  *          defined in <code>EnvironmentMetaDataConstants</code> class can
  *          be used to access the data held by this map.
  *
  * @param transferMetaData
  *          meta data to associate with the transfer.  The information can
  *          only be read, it cannot be updated by the implementation.  This
  *          map may also contain keys with IBM reserved names.  These
  *          entries are defined in the <code>TransferMetaDataConstants</code>
  *          class and have special semantics.
  *
  * @param fileResults
  *          a list of file transfer result objects that describe the source
  *          file name, destination file name and result of each file transfer
  *          operation attempted.
```

```
 *
 * @return    an optional description to enter into the log message describing
 *            transfer completion.  A value of <code>null</code> can be used
 *            when no description is required.
 */
String onDestinationTransferEnd(TransferExitResult transferExitResult,
    String sourceAgentName,
    String destinationAgentName,
    Map<String, String>environmentMetaData,
    Map<String, String>transferMetaData,
    List<FileTransferResult>fileResults);
}
```

Related concepts:

"Customizing WebSphere MQ Managed File Transfer with user exit routines" on page 330
You can customize the features of WebSphere MQ Managed File Transfer by using your own programs
known as user exit routines.

Related reference:

"SourceTransferStartExit.java interface" on page 966

"SourceTransferEndExit.java interface" on page 964

"DestinationTransferStartExit.java interface"

"MonitorExit.java interface" on page 958

"ProtocolBridgeCredentialExit.java interface" on page 959

DestinationTransferStartExit.java interface:

DestinationTransferStartExit.java

```
/*
 *    Licensed Materials - Property of IBM
 *
 *    "Restricted Materials of IBM"
 *
 *    5724-H72
 *
 *    △ Copyright IBM Corp. 2008, 2009 All Rights Reserved.
 *
 *    US Government Users Restricted Rights - Use, duplication or
 *    disclosure restricted by GSA ADP Schedule Contract with
 *    IBM Corp.
 */
package com.ibm.wmqfte.exitpoint.api;

/**
 * An interface that is implemented by classes that want to be invoked as part of
 * user exit routine processing.  This interface defines a method that will be
 * invoked immediately prior to starting a transfer on the agent acting as the
 * destination of the transfer.
 */
public interface DestinationTransferStartExit {

 /**
  * Invoked immediately prior to starting a transfer on the agent acting as
  * the destination of the transfer.
  *
  * @param sourceAgentName
  *           the name of the agent acting as the source of the transfer.
  *
```

```
 *  @param destinationAgentName
 *              the name of the agent acting as the destination of the
 *              transfer.  This is the name of the agent that the
 *              implementation of this method will be invoked from.
 *
 *  @param environmentMetaData
 *              meta data about the environment in which the implementation
 *              of this method is running.  This information can only be read,
 *              it cannot be updated by the implementation.  The constants
 *              defined in <code>EnvironmentMetaDataConstants</code> class can
 *              be used to access the data held by this map.
 *
 *  @param transferMetaData
 *              meta data to associate with the transfer.  The information can
 *              only be read, it cannot be updated by the implementation.  This
 *              map may also contain keys with IBM reserved names.  These
 *              entries are defined in the <code>TransferMetaDataConstants</code>
 *              class and have special semantics.
 *
 *  @param fileSpecs
 *              a list of file specifications that govern the file data to
 *              transfer.  The implementation of this method can modify the
 *              entries in this list and the changes will be reflected in the
 *              files transferred. However, new entries may not be added and
 *              existing entries may not be removed.
 *
 *  @return     a transfer exit result object which is used to determine if the
 *              transfer should proceed, or be cancelled.
 */
 TransferExitResult onDestinationTransferStart(String sourceAgentName,
                            String destinationAgentName,
                     Map<String, String> environmentMetaData,
                     Map<String, String> transferMetaData,
                     List<Reference<String>> fileSpecs);
```

Related concepts:

"Customizing WebSphere MQ Managed File Transfer with user exit routines" on page 330
You can customize the features of WebSphere MQ Managed File Transfer by using your own programs
known as user exit routines.

Related reference:

"SourceTransferStartExit.java interface" on page 966

"SourceTransferEndExit.java interface" on page 964

"DestinationTransferEndExit.java interface" on page 931

"MonitorExit.java interface" on page 958

"ProtocolBridgeCredentialExit.java interface" on page 959

IOExit.java interface:

IOExit.java
```
/*
 *   Licensed Materials - Property of IBM
 *
 *   "Restricted Materials of IBM"
 *
 *   5724-H72
 *
 *   △ Copyright IBM Corp. 2011, 2012 All Rights Reserved.
```

```
 *
 *    US Government Users Restricted Rights - Use, duplication or
 *    disclosure restricted by GSA ADP Schedule Contract with
 *    IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;

import java.io.IOException;
import java.util.Map;

import com.ibm.wmqfte.exitroutine.api.IOExitRecordResourcePath.RecordFormat;

/**
 * An interface that is implemented by classes that you want to be invoked as
 * part of user exit routine processing. This interface defines methods that
 * will be invoked during transfers to perform the underlying file system I/O
 * work for WMQFTE transfers.
 * <p>
 * The {@link #initialize(Map)} method will be called once when the exit is
 * first installed. The WMQFTE agent properties are passed to this method, thus
 * enabling the exit to understand its environment.
 * <p>
 * The {@link #isSupported(String)} method will be invoked during WMQFTE
 * transfers to determine whether the user exit should be used. If the
 * {@link #isSupported(String)} method returns a value of {@code true}, the
 * {@link #newPath(String)} method will be invoked for the paths specified for
 * the transfer request. The returned {@link IOExitPath} instance from a
 * {@link #newPath(String)} method invocation will then be used by the WMQFTE
 * transfer to obtain information about the resource and to transfer data to or
 * from the resource.
 * <p>
 * To obtain transfer context for an I/O exit, a {@link SourceTransferStartExit}
 * or {@link DestinationTransferStartExit} as appropriate, should be installed
 * to enable information to be seen by this exit. The
 * {@link SourceTransferStartExit} or {@link DestinationTransferStartExit} are
 * passed the transfer's environment, metadata, and a list of file
 * specifications for the transfer. The paths for the file specifications are
 * the paths passed to the I/O exit's {@link #newPath(String)} method.
 * <p>
 * Note also that the {@link #isSupported(String)} and {@link #newPath(String)}
 * methods might be called at other times by a WMQFTE agent and not just during
 * transfers. For example, at transfer setup time the I/O system is queried to
 * resolve the full resource paths for transfer.
 */
public interface IOExit {

 /**
  * Invoked once when the I/O exit is first required for use. It is intended
  * to initialize any resources that are required by the exit.
  *
  * @param agentProperties
  *            The values of properties defined for the WMQFTE agent. These
  *            values can only be read, they cannot be updated by the
  *            implementation.
  * @return {@code true} if the initialization is successful and {@code
  *         false} if unsuccessful. If {@code false} is returned from an
  *         exit, the exit will not be used.
  */
 boolean initialize(final Map<String, String> agentProperties);

 /**
  * Indicates whether this I/O user exit supports the specified path.
  * <p>
  * This method is used by WMQFTE to determine whether the I/O user exit
  * should be used within a transfer. If no I/O user exit returns true for
  * this method, the default WMQFTE file I/O function will be used.
  *
```

```
 * @param path
 *             The path to the required I/O resource.
 * @return {@code true} if the specified path is supported by the I/O exit,
 *         {@code false} otherwise
 */
boolean isSupported(String path);

/**
 * Obtains a new {@link IOExitPath} instance for the specified I/O resource
 * path.
 * <p>
 * This method will be invoked by WMQFTE only if the
 * {@link #isSupported(String)} method has been called for the path and
 * returned {@code true}.
 *
 * @param path
 *             The path to the required I/O resource.
 * @return A {@link IOExitPath} instance for the specified path.
 * @throws IOException
 *             If the path cannot be created for any reason.
 */
IOExitPath newPath(String path) throws IOException;

/**
 * Obtains a new {@link IOExitPath} instance for the specified I/O resource
 * path and passes record format and length information required by the
 * WMQFTE transfer.
 * <p>
 * Typically this method will be called for the following cases:
 * <ul>
 * <li>A path where a call to {@link #newPath(String)} has previously
 * returned a {@link IOExitRecordResourcePath} instance and WMQFTE is
 * re-establishing a new {@link IOExitPath} instance for the path, from an
 * internally-serialized state. The passed recordFormat and recordLength
 * will be the same as those for the original
 * {@link IOExitRecordResourcePath} instance.</li>
 * <li>A transfer destination path where the source of the transfer is
 * record oriented. The passed recordFormat and recordLength will be the
 * same as those for the source.</li>
 * </ul>
 * The implementation can act on the record format and length information as
 * deemed appropriate. For example, for a destination agent if the
 * destination does not already exist and the source of the transfer is
 * record oriented, the passed recordFormat and recordLength information
 * could be used to create an appropriate record-oriented destination path.
 * If the destination path already exists, the passed recordFormat and
 * recordLength information could be used to perform a compatibility check
 * and throw an {@link IOException} if the path is not compatible. A
 * compatibility check could ensure that a record oriented path's record
 * format is the same as the passed record format or that the record length
 * is greater or equal to the passed record length.
 * <p>
 * This method will be invoked by WMQFTE only if the
 * {@link #isSupported(String)} method has been called for the path and
 * returned {@code true}.
 *
 * @param path
 *             The path to the required I/O resource.
 * @param recordFormat
 *             The advised record format.
 * @param recordLength
 *             The advised record length.
 * @return A {@link IOExitPath} instance for the specified path.
 * @throws IOException
 *             If the path cannot be created for any reason. For example,
 *             the passed record format or length is incompatible with the
```

```
 *             path's actual record format or length.
 */
 IOExitPath newPath(String path, RecordFormat recordFormat, int recordLength)
   throws IOException;
```

Related concepts:

"Customizing WebSphere MQ Managed File Transfer with user exit routines" on page 330
You can customize the features of WebSphere MQ Managed File Transfer by using your own programs
known as user exit routines.

Related reference:

"Using WebSphere MQ Managed File Transfer transfer I/O user exits" on page 333
You can use WebSphere MQ Managed File Transfer transfer I/O user exits to configure custom code to
perform the underlying file system I/O work for WebSphere MQ Managed File Transfer transfers.

IOExit2.java interface:

IOExit2.java

```
/*
 *   Licensed Materials - Property of IBM
 *
 *   "Restricted Materials of IBM"
 *
 *   5724-H72
 *
 *   △ Copyright IBM Corp. 2012 All Rights Reserved.
 *
 *   US Government Users Restricted Rights - Use, duplication or
 *   disclosure restricted by GSA ADP Schedule Contract with
 *   IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;

import java.io.IOException;
import java.util.List;

import com.ibm.wmqfte.exitroutine.api.IOExitRecordResourcePath.RecordFormat;

public interface IOExit2 extends IOExit {

 /**
  * An extension to the {@link IOExit#newPath(String)} which
  * allows path attributes to be specified, for use when
  * creating or updating a path.
  *
  * @param path as per {@link IOExit#newPath(String)}
  *
  * @param attributes a list of path attributes which the
  *                   exit can choose to apply to file system
  *                   objects that are created, updated
  *                   or read using this path.
  *
  * @return as per {@link IOExit#newPath(String)}
  *
  * @throws IOException as per {@link IOExit#newPath(String)}.
  *                   Can also be thrown if the
  *                   <code>attributes</code> parameter
  *                   contains a
  *                   <code>IOExitPathAttribute</code> which
  *                   the exit implementation does not
  *                   understand.
  */
 IOExitPath newPath(String path,
            List<IOExitPathAttribute> attributes)
 throws IOException;
```

```
 /**
  * An extension to the
  * {@link IOExit#newPath(String, RecordFormat, int)} which
  * allows path attributes to be specified, for use when
  * creating or updating a path.
  *
  * @param path as per {@link IOExit#newPath(String)}
  *
  * @param attributes a list of path attributes which the exit
  *                     can choose to apply to file system
  *                     objects that are created, updated
  *                     or read using this path.
  *
  * @param recordFormat as per {@link IOExit#newPath(String)}
  *
  * @param recordLength as per {@link IOExit#newPath(String)}
  *
  * @return as per {@link IOExit#newPath(String)}
  *
  * @throws IOException as per {@link IOExit#newPath(String)}.
  *                     Can also be thrown if the
  *                     <code>attributes</code> parameter
  *                     contains a
  *                     <code>IOExitPathAttribute</code>
  *                     which the exit implementation
  *                     does not understand.
  */
 IOExitPath newPath(String path,
             List<IOExitPathAttribute> attributes,
             RecordFormat recordFormat,
             int recordLength)
 throws IOException;

}
```

Related concepts:

"Customizing WebSphere MQ Managed File Transfer with user exit routines" on page 330
You can customize the features of WebSphere MQ Managed File Transfer by using your own programs
known as user exit routines.

Related reference:

"Using WebSphere MQ Managed File Transfer transfer I/O user exits" on page 333
You can use WebSphere MQ Managed File Transfer transfer I/O user exits to configure custom code to
perform the underlying file system I/O work for WebSphere MQ Managed File Transfer transfers.

IOExitChannel.java interface:

IOExitChannel.java

```
/*
 *    Licensed Materials - Property of IBM
 *
 *    "Restricted Materials of IBM"
 *
 *    5724-H72
 *
 *    △ Copyright IBM Corp. 2011, 2012 All Rights Reserved.
 *
 *    US Government Users Restricted Rights - Use, duplication or
 *    disclosure restricted by GSA ADP Schedule Contract with
 *    IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;

import java.io.IOException;
```

```
import java.nio.ByteBuffer;

/**
 * Represents a channel that enables data to be read from or written to an
 * {@link IOExitResourcePath} resource.
 */
public interface IOExitChannel {

 /**
  * Obtains the data size for the associated {@link IOExitResourcePath} in
  * bytes.
  *
  * @return The data size in bytes.
  * @throws IOException
  *                 If a problem occurs while attempting obtain the size.
  */
 long size() throws IOException;

 /**
  * Closes the channel, flushing any buffered write data to the resource and
  * releasing any locks.
  *
  * @throws RecoverableIOException
  *                 If a recoverable problem occurs while closing the resource.
  *                 This means that WMQFTE can attempt to recover the transfer.
  * @throws IOException
  *                 If some other I/O problem occurs. For example, the channel might
  *                 already be closed.
  */
 void close() throws RecoverableIOException, IOException;

 /**
  * Reads data from this channel into the given buffer, starting at this
  * channel's current position, and updates the current position by the
  * amount of data read.
  * <p>
  * Data is copied into the buffer starting at its current position and up to
  * its limit. On return, the buffer's position is updated to reflect the
  * number of bytes read.
  *
  * @param buffer
  *                 The buffer that the data is to be copied into.
  * @return The number of bytes read, which might be zero, or -1 if the end of
  *         data has been reached.
  * @throws RecoverableIOException
  *                 If a recoverable problem occurs while reading the data. For a
  *                 WMQFTE transfer this means that it will attempt to recover.
  * @throws IOException
  *                 If some other I/O problem occurs. For a WMQFTE transfer this
  *                 means that it will be failed.
  */
 int read(ByteBuffer buffer) throws RecoverableIOException, IOException;

 /**
  * Writes data to this channel from the given buffer, starting at this
  * channel's current position, and updates the current position by the
  * amount of data written. The channel's resource is grown to accommodate
  * the data, if necessary.
  * <p>
  * Data is copied from the buffer starting at its current position and up to
  * its limit. On return, the buffer's position is updated to reflect the
  * number of bytes written.
  *
  * @param buffer
  *                 The buffer containing the data to be written.
  * @return The number of bytes written, which might be zero.
  * @throws RecoverableIOException
```

```
 *             If a recoverable problem occurs while writing the data. For a
 *             WMQFTE transfer this means that it will attempt to recover.
 * @throws IOException
 *             If some other I/O problem occurs. For a WMQFTE transfer this
 *             means that it will be failed.
 */
 int write(ByteBuffer buffer) throws RecoverableIOException, IOException;

 /**
  * Forces any updates to this channel's resource to be written to its
  * storage device.
  * <p>
  * This method is required to force changes to both the resource's content
  * and any associated metadata to be written to storage.
  *
  * @throws RecoverableIOException
  *             If a recoverable problem occurs while performing the force.
  *             For a WMQFTE transfer this means that it will attempt to
  *             recover.
  * @throws IOException
  *             If some other I/O problem occurs. For a WMQFTE transfer this
  *             means that it will be failed.
  */
 void force() throws RecoverableIOException, IOException;

 /**
  * Attempts to lock the entire resource associated with the channel for
  * shared or exclusive access.
  * <p>
  * The intention is for this method not to block if the lock is currently
  * unavailable.
  *
  * @param shared
  *             {@code true} if a shared lock is required, {@code false} if an
  *             exclusive lock is required.
  * @return A {@link IOExitLock} instance representing the newly acquired
  *         lock or null if the lock cannot be obtained.
  * @throws IOException
  *             If a problem occurs while attempting to acquire the lock.
  */
 IOExitLock tryLock(boolean shared) throws IOException;
}
```

Related concepts:

"Customizing WebSphere MQ Managed File Transfer with user exit routines" on page 330
You can customize the features of WebSphere MQ Managed File Transfer by using your own programs
known as user exit routines.

Related reference:

"Using WebSphere MQ Managed File Transfer transfer I/O user exits" on page 333
You can use WebSphere MQ Managed File Transfer transfer I/O user exits to configure custom code to
perform the underlying file system I/O work for WebSphere MQ Managed File Transfer transfers.

IOExitLock.java interface:

IOExitLock.java

```
/*
 *   Licensed Materials - Property of IBM
 *
 *   "Restricted Materials of IBM"
 *
 *   5724-H72
 *
 *   △ Copyright IBM Corp. 2011, 2012 All Rights Reserved.
 *
```

```
 *    US Government Users Restricted Rights - Use, duplication or
 *    disclosure restricted by GSA ADP Schedule Contract with
 *    IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;

import java.io.IOException;

/**
 * Represents a lock on a resource for either shared or exclusive access.
 * {@link IOExitLock} instances are returned from
 * {@link IOExitChannel#tryLock(boolean)} calls and WMQFTE will request the
 * release of the lock at the appropriate time during a transfer. Additionally, when
 * a {@link IOExitChannel#close()} method is called it will be the
 * responsibility of the channel to release any associated locks.
 */
public interface IOExitLock {

    /**
     * Releases the lock.
     * <p>
     * After this method has been successfully called the lock is to be deemed as invalid.
     *
     * @throws IOException
     *                If the channel associated with the lock is not open or
     *                another problem occurs while attempting to release the lock.
     */
    void release() throws IOException;

    /**
     * Indicates whether this lock is valid.
     * <p>
     * A lock is considered valid until its @ {@link #release()} method is
     * called or the associated {@link IOExitChannel} is closed.
     *
     * @return {@code true} if this lock is valid, {@code false} otherwise.
     */
    boolean isValid();

    /**
     * @return {@code true} if this lock is for shared access, {@code false} if
     *         this lock is for exclusive access.
     */
    boolean isShared();
}
```

Related concepts:

"Customizing WebSphere MQ Managed File Transfer with user exit routines" on page 330
You can customize the features of WebSphere MQ Managed File Transfer by using your own programs known as user exit routines.

Related reference:

"Using WebSphere MQ Managed File Transfer transfer I/O user exits" on page 333
You can use WebSphere MQ Managed File Transfer transfer I/O user exits to configure custom code to perform the underlying file system I/O work for WebSphere MQ Managed File Transfer transfers.

IOExitPath.java interface:

IOExitPath.java

```
/*
 *    Licensed Materials - Property of IBM
 *
 *    "Restricted Materials of IBM"
 *
 *    5724-H72
```

```
 *
 *   △ Copyright IBM Corp. 2011, 2012 All Rights Reserved.
 *
 *   US Government Users Restricted Rights - Use, duplication or
 *   disclosure restricted by GSA ADP Schedule Contract with
 *   IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;

/**
 * Represents an abstract path that can be inspected and queried by WMQFTE for
 * transfer purposes.
 * <p>
 * There are two types of path supported:
 * <ul>
 * <li>{@link IOExitResourcePath} - Represents a path that denotes a data
 * resource. For example, a file, directory, or group of database records.</li>
 * <li>{@link IOExitWildcardPath} - Represents a wildcard path that can be
 * expanded to multiple {@link IOExitResourcePath} instances.</li>
 * </ul>
 */
public abstract interface IOExitPath {

 /**
  * Obtains the abstract path as a {@link String}.
  *
  * @return The abstract path as a {@link String}.
  */
 String getPath();

 /**
  * Obtains the name portion of this abstract path as a {@link String}.
  * <p>
  * For example, a UNIX-style file system implementation evaluates the
  * path {@code /home/fteuser/file1.txt} as having a name of {@code
  * file1.txt}.
  *
  * @return the name portion of this abstract path as a {@link String}.
  */
 String getName();

 /**
  * Obtains the parent path for this abstract path as a {@link String}.
  * <p>
  * For example, a UNIX-style file system implementation evaluates the
  * path {@code /home/fteuser/file1.txt} as having a parent path of {@code
  * /home/fteuser}.
  *
  * @return The parent portion of the path as a {@link String}.
  */
 String getParent();

 /**
  * Obtains the abstract paths that match this abstract path.
  * <p>
  * If this abstract path denotes a directory resource, a list of paths
  * for all resources within the directory are returned.
  * <p>
  * If this abstract path denotes a wildcard, a list of all paths
  * matching the wildcard are returned.
  * <p>
  * Otherwise null is returned, because this abstract path probably denotes a
  * single file resource.
  *
  * @return An array of {@IOExitResourcePath}s that
```

```
 *          match this path, or null if this method is not applicable.
 */
 IOExitResourcePath[] listPaths();
}
```

Related concepts:

"Customizing WebSphere MQ Managed File Transfer with user exit routines" on page 330
You can customize the features of WebSphere MQ Managed File Transfer by using your own programs
known as user exit routines.

Related reference:

"Using WebSphere MQ Managed File Transfer transfer I/O user exits" on page 333
You can use WebSphere MQ Managed File Transfer transfer I/O user exits to configure custom code to
perform the underlying file system I/O work for WebSphere MQ Managed File Transfer transfers.

IOExitPathAttribute.java interface:

IOExitPathAttribute.java

```
/*
 *    Licensed Materials - Property of IBM
 *
 *    "Restricted Materials of IBM"
 *
 *    5724-H72
 *
 *    △ Copyright IBM Corp. 2011, 2012 All Rights Reserved.
 *
 *    US Government Users Restricted Rights - Use, duplication or
 *    disclosure restricted by GSA ADP Schedule Contract with
 *    IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;

/**
 * Represents an attribute associated with an IOExit path.
 * The exit can choose to apply attributes to file system objects
 *
 */
public class IOExitPathAttribute {

 private final String name;
 private final String value;

 /**
  * Constructor for an attribute with a name but no value
  * @param name
  */
 public IOExitPathAttribute(final String name) {
  this.name = name;
  this.value = null;
 }

 /**
  *  Constructor for an attribute with a name and value
  * @param name The name of the attribute
  * @param value The value of the attribute
  */
 public IOExitPathAttribute(final String name,
                                  final String value) {
  this.name = name;
  this.value = value;
 }

 public boolean hasValue() {
  return value != null;
```

```
  }

  public String getName() {
   return name;
  }

  public String getValue() {
   return value;
  }
}
```

Related concepts:

"Customizing WebSphere MQ Managed File Transfer with user exit routines" on page 330
You can customize the features of WebSphere MQ Managed File Transfer by using your own programs
known as user exit routines.

Related reference:

"Using WebSphere MQ Managed File Transfer transfer I/O user exits" on page 333
You can use WebSphere MQ Managed File Transfer transfer I/O user exits to configure custom code to
perform the underlying file system I/O work for WebSphere MQ Managed File Transfer transfers.

IOExitProperties.java interface:

IOExitProperties.java

```
/*
 *   Licensed Materials - Property of IBM
 *
 *   "Restricted Materials of IBM"
 *
 *   5724-H72
 *
 *   △ Copyright IBM Corp. 2011, 2012 All Rights Reserved.
 *
 *   US Government Users Restricted Rights - Use, duplication or
 *   disclosure restricted by GSA ADP Schedule Contract with
 *   IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;

/**
 * Properties that determine how WMQFTE treats an {@link IOExitPath} for certain
 * aspects of I/O. For example, whether to use intermediate files.
 */
public class IOExitProperties {

 private boolean rereadSourceOnRestart = true;
 private boolean rechecksumSourceOnRestart = true;
 private boolean rechecksumDestinationOnRestart = true;
 private boolean useIntermediateFileAtDestination = true;
 private boolean requiresSingleThreadedChannelIO = false;

 /**
  * Determines whether the I/O exit implementation expects the resource to be
  * re-read from the start if a transfer is restarted.
  *
  * @return {@code true} if, on restart, the I/O exit expects the source
  *         resource to be opened at the beginning and re-read from the
  *         beginning (the {@link IOExitPath#openForRead(long)} method is
  *         always invoked with 0L as an argument). {@code false} if, on
  *         restart, the I/O exit expects the source to be opened at the
  *         offset that the source agent intends to start reading from (the
  *         {@link IOExitPath#openForRead(long)} method can be invoked with a
  *         non-zero value as its argument).
  */
 public boolean getRereadSourceOnRestart() {
```

```
    return rereadSourceOnRestart;
}

/**
 * Sets the value to determine whether the I/O exit implementation expects
 * the resource to be re-read from the beginning if a transfer is restarted.
 * <p>
 * The default is {@code true}. The I/O exit should call this method when
 * required to change this value.
 *
 * @param rereadSourceOnRestart
 *            {@code true} if, on restart, the I/O exit expects the source
 *            resource to be opened at the beginning and re-read from the
 *            beginning (the {@link IOExitPath#openForRead(long)} method
 *            is always invoked with 0L as an argument). {@code false}
 *            if, on restart, the I/O exit expects the source to be opened
 *            at the offset that the source agent intends to start reading
 *            from (the {@link IOExitPath#openForRead(long)} method can be
 *            invoked with a non-zero value as its argument).
 */
public void setRereadSourceOnRestart(boolean rereadSourceOnRestart) {
    this.rereadSourceOnRestart = rereadSourceOnRestart;
}

/**
 * Determines whether the I/O exit implementation requires the source
 * resource to be re-checksummed if the transfer is restarted.
 * Re-checksumming takes place only if the
 * {@link #getRereadSourceOnRestart()} method returns {@code true}.
 *
 * @return {@code true} if, on restart, the I/O exit expects the already-
 *         transferred portion of the source to be re-checksummed for
 *         inconsistencies. Use this option in environments
 *         where the source could be changed during a restart. {@code
 *         false} if, on restart, the I/O exit does not require the
 *         already-transferred portion of the source to be re-checksummed.
 */
public boolean getRechecksumSourceOnRestart() {
    return rechecksumSourceOnRestart;
}

/**
 * Sets the value to determine whether the I/O exit implementation requires
 * the source resource to be re-checksummed if the transfer is restarted.
 * Re-checksumming takes place only if the
 * {@link #getRereadSourceOnRestart()} method returns {@code true}.
 * <p>
 * The default is {@code true}. The I/O exit should call this method when
 * required to change this value.
 *
 * @param rechecksumSourceOnRestart
 *            {@code true} if, on restart, the I/O exit expects the already
 *            transferred portion of the source to be re-checksummed
 *            for inconsistencies. Use this option in environments
 *            where the source could be changed during a restart.
 *            {@code false} if, on restart, the I/O exit does not
 *            require the already-transferred portion of the source to be
 *            re-checksummed.
 */
public void setRechecksumSourceOnRestart(boolean rechecksumSourceOnRestart) {
    this.rechecksumSourceOnRestart = rechecksumSourceOnRestart;
}

/**
 * Determines whether the I/O exit implementation requires the destination
 * resource to be re-checksummed if the transfer is restarted.
 *
```

```
 * @return {@code true} if, on restart, the I/O exit expects the already
 *         transferred portion of the destination to be re-checksummed to
 *         check for inconsistencies. This option should be used in
 *         environments where the destination could have been changed while
 *         a restart is occurring. {@code false} if, on restart, the I/O exit
 *         does not require the already transferred portion of the
 *         destination to be re-checksummed.
 */
public boolean getRechecksumDestinationOnRestart() {
 return rechecksumDestinationOnRestart;
}

/**
 * Sets the value to determine whether the I/O exit implementation requires
 * the destination resource to be re-checksummed if the transfer is
 * restarted.
 * <p>
 * The default is {@code true}. The I/O exit should call this method when
 * required to change this value.
 *
 * @param rechecksumDestinationOnRestart
 *         {@code true} if, on restart, the I/O exit expects the already-
 *         transferred portion of the destination to be re-checksummed
 *         for inconsistencies. Use this option in environments
 *         where the destination could have been changed during a
 *         restart. {@code false} if, on restart, the I/O exit does not
 *         require the already-transferred portion of the destination
 *         to be re-checksummed.
 */
public void setRechecksumDestinationOnRestart(
  boolean rechecksumDestinationOnRestart) {
 this.rechecksumDestinationOnRestart = rechecksumDestinationOnRestart;
}

/**
 * Determines whether the I/O exit implementation requires the use of an
 * intermediate file when writing the data at the destination. The
 * intermediate file mechanism is typically used to prevent an incomplete
 * destination resource from being processed.
 *
 * @return {@code true} if data should be written to an intermediate file at
 *         the destination and then renamed (to the requested destination
 *         path name as specified in the transfer request) after the transfer is
 *         complete. {@code false} if data should be written directly to the
 *         requested destination path name without the use of an
 *         intermediate file.
 */
public boolean getUseIntermediateFileAtDestination() {
 return useIntermediateFileAtDestination;
}

/**
 * Sets the value to determine whether the I/O exit implementation requires
 * the use of an intermediate file when writing the data at the destination.
 * The intermediate file mechanism is typically used to prevent an
 * incomplete destination resource from being processed.
 *
 * <p>
 * The default is {@code true}. The I/O exit should call this method when
 * required to change this value.
 *
 * @param useIntermediateFileAtDestination
 *         {@code true} if data should be written to an intermediate file
 *         at the destination and then renamed (to the requested
 *         destination path name as specified in the transfer request) after
 *         the transfer is complete. {@code false} if data should be written
 *         directly to the requested destination path name without the
```

```
 *           use of an intermediate file
 */
public void setUseIntermediateFileAtDestination(
  boolean useIntermediateFileAtDestination) {
 this.useIntermediateFileAtDestination = useIntermediateFileAtDestination;
}

/**
 * Determines whether the I/O exit implementation requires
 * {@link IOExitChannel} instances to be accessed by a single thread only.
 *
 * @return {@code true} if {@link IOExitChannel} instances are to be
 *         accessed by a single thread only.
 */
public boolean requiresSingleThreadedChannelIO() {
 return requiresSingleThreadedChannelIO;
}

/**
 * Sets the value to determine whether the I/O exit implementation requires
 * channel operations for a particular instance to be accessed by a
 * single thread only.
 * <p>
 * For certain I/O implementations it is necessary that resource path
 * operations such as open, read, write, and close are invoked only from a
 * single execution {@link Thread}. When set {@code true}, WMQFTE ensures
 * that the following are invoked on a single thread:
 * <ul>
 * <li>{@link IOExitResourcePath#openForRead(long) method and all methods of
 * the returned {@link IOExitChannel} instance.</li>
 * <li>{@link IOExitResourcePath#openForWrite(boolean)) method and all
 * methods of the returned {@link IOExitChannel} instance.</li>
 * </ul>
 * <p>
 * This has a slight performance impact, hence enable single-threaded channel
 * I/O only when absolutely necessary.
 * <p>
 * The default is {@code false}. The I/O exit should call this method when
 * required to change this value.
 *
 * @param requiresSingleThreadedChannelIO
 *           {@code true} if {@link IOExitChannel} instances are to be
 *           accessed by a single thread only.
 */
public void setRequiresSingleThreadedChannelIO(boolean requiresSingleThreadedChannelIO) {
 this.requiresSingleThreadedChannelIO = requiresSingleThreadedChannelIO;
}
}
```

Related concepts:

"Customizing WebSphere MQ Managed File Transfer with user exit routines" on page 330
You can customize the features of WebSphere MQ Managed File Transfer by using your own programs
known as user exit routines.

Related reference:

"Using WebSphere MQ Managed File Transfer transfer I/O user exits" on page 333
You can use WebSphere MQ Managed File Transfer transfer I/O user exits to configure custom code to
perform the underlying file system I/O work for WebSphere MQ Managed File Transfer transfers.

IOExitRecordChannel.java interface:

IOExitRecordChannel.java

```
/*
 *   Licensed Materials - Property of IBM
 *
```

```
 *   "Restricted Materials of IBM"
 *
 *   5724-H72
 *
 *   △ Copyright IBM Corp. 2011, 2012 All Rights Reserved.
 *
 *   US Government Users Restricted Rights - Use, duplication or
 *   disclosure restricted by GSA ADP Schedule Contract with
 *   IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;

import java.io.IOException;
import java.nio.ByteBuffer;

/**
 * Represents a channel that enables records of data to be read from or written
 * to an {@link IOExitRecordResourcePath} resource.
 * <p>
 * This is an extension of the {@link IOExitChannel} interface such that the
 * {@link #read(java.nio.ByteBuffer)} and {@link #write(java.nio.ByteBuffer)}
 * methods are expected to deal in whole records of data only. That is, the
 * {@link java.nio.ByteBuffer} returned from the read method and passed to the
 * write method is assumed to contain one or more complete records.
 */
public interface IOExitRecordChannel extends IOExitChannel {

 /**
  * Reads records from this channel into the given buffer, starting at this
  * channel's current position, and updates the current position by the
  * amount of data read.
  * <p>
  * Record data is copied into the buffer starting at its current position
  * and up to its limit. On return, the buffer's position is updated to
  * reflect the number of bytes read.
  * <p>
  * Only whole records are copied into the buffer.
  * <p>
  * For a fixed-record-format resource, this might be multiple records. The
  * amount of data in the return buffer does not necessarily need to be a
  * multiple of the record length, but the last record is still to be treated
  * as a complete record and padded as required by the caller.
  * <p>
  * For a variable-format resource, this is a single whole record of a size
  * corresponding to the amount of return data or multiple whole records with
  * all except the last being treated as records of maximum size.
  *
  * @param buffer
  *           The buffer that the record data is to be copied into.
  * @return The number of bytes read, which might be zero, or -1 if the end of
  *         data has been reached.
  * @throws RecoverableIOException
  *           If a recoverable problem occurs while reading the data. For a
  *           WMQFTE transfer this means that it will attempt to recover.
  * @throws IOException
  *           If some other I/O problem occurs, for example, if the passed
  *           buffer is insufficient to contain at least one complete
  *           record). For a WMQFTE transfer this means that it will be
  *           failed.
  */
 int read(ByteBuffer buffer) throws RecoverableIOException, IOException;

 /**
  * Writes records to this channel from the given buffer, starting at this
  * channel's current position, and updates the current position by the
  * amount of data written. The channel's resource is grown to accommodate
  * the data, if necessary.
```

```
 *  <p>
 *  Record data is copied from the buffer starting at its current position
 *  and up to its limit. On return, the buffer's position is updated to
 *  reflect the number of bytes written.
 *  <p>
 *  The buffer is expected to contain only whole records.
 *  <p>
 *  For a fixed-record-format resource, this might be multiple records and if
 *  there is insufficient data in the buffer for a complete record, the
 *  record is to be padded as required to complete the record.
 *  <p>
 *  For a variable-record format resource the buffer is normally expected to
 *  contain a single record of length corresponding to the amount of data
 *  within the buffer. However, if the amount of data within the buffer
 *  exceeds the maximum record length, the implementation can either:
 *  <ol>
 *  <li>throw an {@link IOException} indicating that it cannot handle the
 *  situation.</li>
 *  <li>Consume a record's worth of data from the buffer, leaving the remaining
 *  data within the buffer.</li>
 *  <li>Consume all the buffer data and just write what it can to the current
 *  record. This effectively truncates the data.</li>
 *  <li>Consume all the buffer data and write to multiple records.</li>
 *  </ol>
 *
 *  @param buffer
 *              The buffer containing the data to be written.
 *  @return The number of bytes written, which might be zero.
 *  @throws RecoverableIOException
 *              If a recoverable problem occurs while writing the data. For a
 *              WMQFTE transfer this means that it will attempt to recover.
 *  @throws IOException
 *              If some other I/O problem occurs. For a WMQFTE transfer this
 *              means that it will be failed.
 */
 int write(ByteBuffer buffer) throws RecoverableIOException, IOException;

}
```

Related concepts:

"Customizing WebSphere MQ Managed File Transfer with user exit routines" on page 330
You can customize the features of WebSphere MQ Managed File Transfer by using your own programs known as user exit routines.

Related reference:

"Using WebSphere MQ Managed File Transfer transfer I/O user exits" on page 333
You can use WebSphere MQ Managed File Transfer transfer I/O user exits to configure custom code to perform the underlying file system I/O work for WebSphere MQ Managed File Transfer transfers.

IOExitRecordResourcePath.java interface:

IOExitRecordResourcePath.java

```
/*
 *   Licensed Materials - Property of IBM
 *
 *   "Restricted Materials of IBM"
 *
 *   5724-H72
 *
 *   △ Copyright IBM Corp. 2011, 2012 All Rights Reserved.
 *
 *   US Government Users Restricted Rights - Use, duplication or
 *   disclosure restricted by GSA ADP Schedule Contract with
 *   IBM Corp.
 */
```

```
package com.ibm.wmqfte.exitroutine.api;

import java.io.IOException;

/**
 * Represents a path that denotes a record-oriented data resource (for example,
 * a z/OS data set). It allows the data to be located, the record format to be
 * understood, and {@link IOExitRecordChannel} instances to be created for read
 * or write operations.
 */
public interface IOExitRecordResourcePath extends IOExitResourcePath {

 /**
  * Record formats for record-oriented resources.
  */
 public enum RecordFormat {
  FIXED, VARIABLE
 }

 /**
  * Obtains the record length for records that are maintained by the resource
  * denoted by this abstract path.
  * <p>
  * For a resource with fixed-length records, the data for each record read
  * and written is assumed to be this length.
  * <p>
  * For a resource with variable-length records, this is the maximum length
  * for a record's data.
  * <p>
  * This method should return a value greater than zero, otherwise it can
  * result in the failure of a WMQFTE transfer that involves this abstract
  * path.
  *
  * @return The record length, in bytes, for records maintained by the
  *         resource.
  */
 int getRecordLength();

 /**
  * Obtains record format, as a {@link RecordFormat} instance, for records
  * that are maintained by the resource denoted by this abstract path.
  *
  * @return A {@link RecordFormat} instance for the record format for records
  *         that are maintained by the resource denoted by this abstract
  *         path.
  */
 RecordFormat getRecordFormat();

 /**
  * Opens a {@link IOExitRecordChannel} instance for reading data from the
  * resource denoted by this abstract path. The current data byte position
  * for the resource is expected to be the passed position value, such that
  * when {@link IOExitRecordChannel#read(java.nio.ByteBuffer)} is called,
  * data starting from that position is read.
  * <p>
  * Note that the data byte read position will be on a record boundary.
  *
  * @param position
  *            The required data byte read position.
  * @return A new {@link IOExitRecordChannel} instance allowing data to be
  *         read from the resource denoted by this abstract path.
  * @throws RecoverableIOException
  *            If a recoverable problem occurs while attempting to open the
  *            resource for reading. This means that WMQFTE can attempt to
  *            recover the transfer.
  * @throws IOException
  *            If some other I/O problem occurs.
```

```
 */
 IOExitRecordChannel openForRead(long position)
   throws RecoverableIOException, IOException;

 /**
  * Opens a {@link IOExitRecordChannel} instance for writing data to the
  * resource denoted by this abstract path. Writing of data, using the
  * {@link IOExitRecordChannel#write(java.nio.ByteBuffer)} method, starts at
  * either the beginning of the resource or end of the current data for the
  * resource, depending on the specified append parameter.
  *
  * @param append
  *           When {@code true} indicates that data written to the resource
  *           should be appended to the end of the current data. When
  *           {@code false} indicates that writing of data is to start at
  *           the beginning of the resource; any existing data is lost.
  * @return A new {@link IOExitRecordChannel} instance allowing data to be
  *         written to the resource denoted by this abstract path.
  * @throws RecoverableIOException
  *             If a recoverable problem occurs while attempting to open the
  *             resource for writing. This means that WMQFTE can attempt to
  *             recover the transfer.
  * @throws IOException
  *             If some other I/O problem occurs.
  */
 IOExitRecordChannel openForWrite(boolean append)
   throws RecoverableIOException, IOException;
}
```

Related concepts:

"Customizing WebSphere MQ Managed File Transfer with user exit routines" on page 330
You can customize the features of WebSphere MQ Managed File Transfer by using your own programs
known as user exit routines.

Related reference:

"Using WebSphere MQ Managed File Transfer transfer I/O user exits" on page 333
You can use WebSphere MQ Managed File Transfer transfer I/O user exits to configure custom code to
perform the underlying file system I/O work for WebSphere MQ Managed File Transfer transfers.

IOExitRecordResourcePath2.java interface:

IOExitRecordResourcePath2.java

```
/*
 *    Licensed Materials - Property of IBM
 *
 *    "Restricted Materials of IBM"
 *
 *    5724-H72
 *
 *    △ Copyright IBM Corp. 2011, 2012 All Rights Reserved.
 *
 *    US Government Users Restricted Rights - Use, duplication or
 *    disclosure restricted by GSA ADP Schedule Contract with
 *    IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;

public interface IOExitRecordResourcePath2
extends IOExitResourcePath2, IOExitRecordResourcePath {

}
```

Related concepts:

"Customizing WebSphere MQ Managed File Transfer with user exit routines" on page 330
You can customize the features of WebSphere MQ Managed File Transfer by using your own programs
known as user exit routines.

Related reference:

"Using WebSphere MQ Managed File Transfer transfer I/O user exits" on page 333
You can use WebSphere MQ Managed File Transfer transfer I/O user exits to configure custom code to
perform the underlying file system I/O work for WebSphere MQ Managed File Transfer transfers.

IOExitResourcePath.java interface:

IOExitResourcePath.java

```
/*
 *    Licensed Materials - Property of IBM
 *
 *    "Restricted Materials of IBM"
 *
 *    5724-H72
 *
 *    △ Copyright IBM Corp. 2011, 2012 All Rights Reserved.
 *
 *    US Government Users Restricted Rights - Use, duplication or
 *    disclosure restricted by GSA ADP Schedule Contract with
 *    IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;

import java.io.IOException;

/**
 * Represents a path that denotes a data resource (for example, a file,
 * directory, or group of database records). It allows the data to be located
 * and {@link IOExitChannel} instances to be created for read or write
 * operations.
 * <p>
 * There are two types of data resources as follows:
 * <ul>
 * <li>Directory - a container for other data resources. The
 * {@link #isDirectory()} method returns {@code true} for these.</li>
 * <li>File - a data container. This allows data to be read from or written to
 * it. The {@link #isFile()} method returns {@code true} for these.</li>
 * </ul>
 */
public interface IOExitResourcePath extends IOExitPath {

 /**
  * Creates a new {@link IOExitResourcePath} instance for a child path of the
  * resource denoted by this abstract path.
  * <p>
  * For example, with a UNIX-style path, {@code
  * IOExitResourcePath("/home/fteuser/test").newPath("subtest")} could be
  * equivalent to: {@code IOExitResourcePath("/home/fteuser/test/subtest")}
  *
  * @param child
  *            The child path name.
  * @return A new {@link IOExitResourcePath} instance that represents a child
  *         of this path.
  */
 IOExitResourcePath newPath(final String child);

 /**
  * Creates the directory path for the resource denoted by this abstract
  * path, including any necessary but nonexistent parent directories. If the
  * directory path already exists, this method has no effect.
```

```
 * <p>
 * If this operation fails, it might have succeeded in creating some of the
 * necessary parent directories.
 *
 * @throws IOException
 *                If the directory path cannot be fully created, when it does
 *                not already exist.
 */
void makePath() throws IOException;

/**
 * Obtains the canonical path of the abstract path as a {@link String}.
 * <p>
 * A canonical path is defined as being absolute and unique. For example,
 * the path can be represented as UNIX-style relative path: {@code
 * test/file.txt} but the absolute and unique canonical path representation
 * is: {@code /home/fteuser/test/file.txt}
 *
 * @return The canonical path as a {@link String}.
 * @throws IOException
 *                If the canonical path cannot be determined for any reason.
 */
String getCanonicalPath() throws IOException;

/**
 * Tests if this abstract path is an absolute path.
 * <p>
 * For example, a UNIX-style path, {@code /home/fteuser/test} is an absolute
 * path, whereas {@code fteuser/test} is not.
 *
 * @return {@code true} if this abstract path is an absolute path, {@code
 *         false} otherwise.
 */
boolean isAbsolute();

/**
 * Tests if the resource denoted by this abstract path exists.
 *
 * @return {@code true} if the resource denoted by this abstract path
 *         exists, {@code false} otherwise.
 * @throws IOException
 *                If the existence of the resource cannot be determined for any
 *                reason.
 */
boolean exists() throws IOException;

/**
 * Tests whether the calling application can read the resource denoted by
 * this abstract path.
 *
 * @return {@code true} if the resource for this path exists and can be
 *         read, {@code false} otherwise.
 * @throws IOException
 *                If a problem occurs while attempting to determine if the
 *                resource can be read.
 */
boolean canRead() throws IOException;

/**
 * Tests whether the calling application can modify the resource denoted by
 * this abstract path.
 *
 * @return {@code true} if the resource for this path exists and can be
 *         modified, {@code false} otherwise.
 * @throws IOException
 *                If a problem occurs while attempting to determine if the
 *                resource can be modified.
```

```
 */
boolean canWrite() throws IOException;

/**
 * Tests whether the specified user is permitted to read the resource
 * denoted by this abstract path.
 * <p>
 * When WMQFTE invokes this method, the user identifier is the MQMD user
 * identifier for the requesting transfer.
 *
 * @param userId
 *            User identifier to test for access.
 * @return {@code true} if the resource for this abstract path exists and is
 *         permitted to be read by the specified user, {@code false}
 *         otherwise.
 * @throws IOException
 *            If a problem occurs while attempting to determine if the user
 *            is permitted to read the resource.
 */
boolean readPermitted(String userId) throws IOException;

/**
 * Tests whether the specified user is permitted to modify the resource
 * denoted by this abstract path.
 * <p>
 * When WMQFTE invokes this method, the user identifier is the MQMD user
 * identifier for the requesting transfer.
 *
 * @param userId
 *            User identifier to test for access.
 * @return {@code true} if the resource for this abstract path exists and is
 *         permitted to be modified by the specified user, {@code false}
 *         otherwise.
 * @throws IOException
 *            If a problem occurs while attempting to determine if the user
 *            is permitted to modify the resource.
 */
boolean writePermitted(String userId) throws IOException;

/**
 * Tests if the resource denoted by this abstract path is a directory-type
 * resource.
 *
 * @return {@code true} if the resource denoted by this abstract path is a
 *         directory type resource, {@code false} otherwise.
 */
boolean isDirectory();

/**
 * Creates the resource denoted by this abstract path, if it does not
 * already exist.
 *
 * @return {@code true} if the resource does not exist and was successfully
 *         created, {@code false} if the resource already existed.
 * @throws RecoverableIOException
 *            If a recoverable problem occurs while attempting to create
 *            the resource. This means that WMQFTE can attempt to recover
 *            the transfer.
 * @throws IOException
 *            If some other I/O problem occurs.
 */
boolean createNewPath() throws RecoverableIOException, IOException;

/**
 * Tests if the resource denoted by this abstract path is a file-type
 * resource.
 *
```

```
 * @return {@code true} if the resource denoted by this abstract path is a
 *         file type resource, {@code false} otherwise.
 */
boolean isFile();

/**
 * Obtains the last modified time for the resource denoted by this abstract
 * path.
 * <p>
 * This time is measured in milliseconds since the epoch (00:00:00 GMT,
 * January 1, 1970).
 *
 * @return The last modified time for the resource denoted by this abstract
 *         path, or a value of 0L if the resource does not exist or a
 *         problem occurs.
 */
long lastModified();

/**
 * Deletes the resource denoted by this abstract path.
 * <p>
 * If the resource is a directory, it must be empty for the delete to work.
 *
 * @throws IOException
 *                 If the delete of the resource fails for any reason.
 */
void delete() throws IOException;

/**
 * Renames the resource denoted by this abstract path to the specified
 * destination abstract path.
 * <p>
 * The rename should still be successful if the resource for the specified
 * destination abstract path already exists and it is possible to replace
 * it.
 *
 * @param destination
 *                 The new abstract path for the resource denoted by this
 *                 abstract path.
 * @throws IOException
 *                 If the rename of the resource fails for any reason.
 */
void renameTo(IOExitResourcePath destination) throws IOException;

/**
 * Creates a new path to use for writing to a temporary resource that did
 * not previously exist.
 * <p>
 * The implementation can choose the abstract path name for the temporary
 * resource. However, for clarity and problem diagnosis, the abstract path
 * name for the temporary resource should be based on this abstract path
 * name with the specified suffix appended and additional characters to make
 * the path unique (for example, sequence numbers), as required.
 * <p>
 * When WMQFTE transfers data to a destination it normally attempts to first
 * write to a temporary resource then on transfer completion renames the
 * temporary resource to the required destination. This method is called by
 * WMQFTE to create a new temporary resource path. The returned path should
 * be new and the resource should not previously exist.
 *
 * @param suffix
 *                 Recommended suffix to use for the generated temporary path.
 *
 * @return A new {@link IOExitResourcePath} instance for the temporary
 *         resource path, that did not previously exist.
 * @throws RecoverableIOException
 *                 If a recoverable problem occurs whilst attempting to create
```

```
 *             the temporary resource. This means that WMQFTE can attempt to
 *             recover the transfer.
 * @throws IOException
 *             If some other I/O problem occurs.
 */
IOExitResourcePath createTempPath(String suffix)
  throws RecoverableIOException, IOException;

/**
 * Opens a {@link IOExitChannel} instance for reading data from the resource
 * denoted by this abstract path. The current data byte position for the
 * resource is expected to be the passed position value, such that when
 * {@link IOExitChannel#read(java.nio.ByteBuffer)} is called, data starting
 * from that position is read.
 *
 * @param position
 *             The required data byte read position.
 * @return A new {@link IOExitChannel} instance allowing data to be read
 *         from the resource denoted by this abstract path.
 * @throws RecoverableIOException
 *             If a recoverable problem occurs while attempting to open the
 *             resource for reading. This means that WMQFTE can attempt to
 *             recover the transfer.
 * @throws IOException
 *             If some other I/O problem occurs.
 */
IOExitChannel openForRead(long position) throws RecoverableIOException,
  IOException;

/**
 * Opens a {@link IOExitChannel} instance for writing data to the resource
 * denoted by this abstract path. Writing of data, using the
 * {@link IOExitChannel#write(java.nio.ByteBuffer)} method, starts at either
 * the beginning of the resource or end of the current data for the
 * resource, depending on the specified append parameter.
 *
 * @param append
 *             When {@code true} indicates that data written to the resource
 *             should be appended to the end of the current data. When
 *             {@code false} indicates that writing of data is to start at
 *             the beginning of the resource; any existing data is lost.
 * @return A new {@link IOExitChannel} instance allowing data to be written
 *         to the resource denoted by this abstract path.
 * @throws RecoverableIOException
 *             If a recoverable problem occurs whilst attempting to open the
 *             resource for writing. This means that WMQFTE can attempt to
 *             recover the transfer.
 * @throws IOException
 *             If some other I/O problem occurs.
 */
IOExitChannel openForWrite(boolean append) throws RecoverableIOException,
  IOException;

/**
 * Tests if the resource denoted by this abstract path is in use by another
 * application. Typically, this is because another application has a lock on
 * the resource either for shared or exclusive access.
 *
 * @return {code true} if resource denoted by this abstract path is in use
 *         by another application, {@code false} otherwise.
 */
boolean inUse();

/**
 * Obtains a {@link IOExitProperties} instance for properties associated
 * with the resource denoted by this abstract path.
 * <p>
```

```
 *  WMQFTE will read these properties to govern how a transfer behaves when
 *  interacting with the resource.
 *
 *  @return A {@link IOExitProperties} instance for properties associated
 *          with the resource denoted by this abstract path.
 */
IOExitProperties getProperties();

}
```

Related concepts:

"Customizing WebSphere MQ Managed File Transfer with user exit routines" on page 330
You can customize the features of WebSphere MQ Managed File Transfer by using your own programs
known as user exit routines.

Related reference:

"Using WebSphere MQ Managed File Transfer transfer I/O user exits" on page 333
You can use WebSphere MQ Managed File Transfer transfer I/O user exits to configure custom code to
perform the underlying file system I/O work for WebSphere MQ Managed File Transfer transfers.

IOExitResourcePath2.java interface:

IOExitResourcePath2.java

```
/*
 *    Licensed Materials - Property of IBM
 *
 *    "Restricted Materials of IBM"
 *
 *    5724-H72
 *
 *    △ Copyright IBM Corp. 2011, 2012 All Rights Reserved.
 *
 *    US Government Users Restricted Rights - Use, duplication or
 *    disclosure restricted by GSA ADP Schedule Contract with
 *    IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;

import java.io.IOException;
import java.util.List;

public interface IOExitResourcePath2 extends IOExitResourcePath {

 /**
  * @return a list of path attributes which the exit wishes
  *         to associate with data read from the path.
  */
 List<IOExitPathAttribute> getAttributes();

 /**
  * An extension to the
  * {@link IOExitResourcePath#newPath(String)} which allows
  * path attributes to be specified, for use when creating
  * or updating a path.
  *
  * @param child as per
  *              {@link IOExitResourcePath#newPath(String)}
  *
  * @param attributes a list of path attributes which the
  *                   exit can choose to apply to file system
  *                   objects that are created, updated
  *                   or read using this path.
  *
  * @return as per {@link IOExitResourcePath#newPath(String)}
  *
```

```
 *  @throws IOException as per
 *                {@link IOExitResourcePath#newPath(String)}.
 *                Can also be thrown if the
 *                <code>attributes</code> parameter
 *                contains a
 *                <code>IOExitPathAttribute</code> which
 *                the exit implementation does not
 *                understand.
 */
IOExitResourcePath newPath(final String child,
            List<IOExitPathAttribute> attributes)
throws IOException;
}
```

Related concepts:

"Customizing WebSphere MQ Managed File Transfer with user exit routines" on page 330
You can customize the features of WebSphere MQ Managed File Transfer by using your own programs
known as user exit routines.

Related reference:

"Using WebSphere MQ Managed File Transfer transfer I/O user exits" on page 333
You can use WebSphere MQ Managed File Transfer transfer I/O user exits to configure custom code to
perform the underlying file system I/O work for WebSphere MQ Managed File Transfer transfers.

IOExitWildcardPath.java interface:

IOExitWildcardPath.java

```
/*
 *   Licensed Materials - Property of IBM
 *
 *   "Restricted Materials of IBM"
 *
 *   5724-H72
 *
 *   △ Copyright IBM Corp. 2011, 2012 All Rights Reserved.
 *
 *   US Government Users Restricted Rights - Use, duplication or
 *   disclosure restricted by GSA ADP Schedule Contract with
 *   IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;

/**
 * Represents a path that denotes a wildcard. This can be used to match multiple
 * resource paths.
 */
public interface IOExitWildcardPath extends IOExitPath {
```

Related concepts:

"Customizing WebSphere MQ Managed File Transfer with user exit routines" on page 330
You can customize the features of WebSphere MQ Managed File Transfer by using your own programs
known as user exit routines.

Related reference:

"Using WebSphere MQ Managed File Transfer transfer I/O user exits" on page 333
You can use WebSphere MQ Managed File Transfer transfer I/O user exits to configure custom code to
perform the underlying file system I/O work for WebSphere MQ Managed File Transfer transfers.

MonitorExit.java interface:

MonitorExit.java
```
/*
 *   Licensed Materials - Property of IBM
 *
 *   "Restricted Materials of IBM"
 *
 *   5724-H72
 *
 *   △ Copyright IBM Corp. 2009 All Rights Reserved.
 *
 *   US Government Users Restricted Rights - Use, duplication or
 *   disclosure restricted by GSA ADP Schedule Contract with
 *   IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;

import java.util.Map;

/**
 * An interface that is implemented by classes that want to be invoked as part of
 * user exit routine processing.  This interface defines a method that will be
 * invoked immediately prior to starting a task as the result of a monitor trigger
 */
public interface MonitorExit {

 /**
  * Invoked immediately prior to starting a task as the result of a monitor
  * trigger.
  *
  * @param environmentMetaData
  *            meta data about the environment in which the implementation
  *            of this method is running.  This information can only be read,
  *            it cannot be updated by the implementation.  The constant
  *            defined in <code>EnvironmentMetaDataConstants</code> class can
  *            be used to access the data held by this map.
  *
  * @param monitorMetaData
  *            meta data to associate with the monitor. The meta data passed
  *            to this method can be altered, and the changes will be
  *            reflected in subsequent exit routine invocations. This map
  *            also contains keys with IBM reserved names.  These entries are
  *            defined in the <code>MonitorMetaDataConstants</code> class and
  *            have special semantics. The the values of the IBM reserved names
  *            cannot be modified by the exit
  *
  * @param taskDetails
  *            An XML String representing the task to be executed as a result of
  *            the monitor triggering. This XML string may be modified by the
  *            exit
  *
  * @return    a monitor exit result object which is used to determine if the
  *            task should proceed, or be cancelled.
  */
 MonitorExitResult onMonitor(Map<String, String> environmentMetaData,
                    Map<String, String> monitorMetaData,
                  Reference<String> taskDetails);
}
```

Related concepts:

"Resource monitoring" on page 215
You can monitor WebSphere MQ Managed File Transfer resources; for example, a queue or a directory.
When a condition on this resource is satisfied, the resource monitor starts a task, such as a file transfer.
You can create a resource monitor by using the `fteCreateMonitor` command or the Monitors view in the
WebSphere MQ Managed File Transfer plug-in for WebSphere MQ Explorer.

"Customizing WebSphere MQ Managed File Transfer with user exit routines" on page 330
You can customize the features of WebSphere MQ Managed File Transfer by using your own programs
known as user exit routines.

Related reference:

"SourceTransferStartExit.java interface" on page 966

"SourceTransferEndExit.java interface" on page 964

"DestinationTransferStartExit.java interface" on page 932

"DestinationTransferEndExit.java interface" on page 931

"ProtocolBridgeCredentialExit.java interface"

ProtocolBridgeCredentialExit.java interface:

ProtocolBridgeCredentialExit.java

```
/*
 *   Licensed Materials - Property of IBM
 *
 *   "Restricted Materials of IBM"
 *
 *   5724-H72
 *
 *   △ Copyright IBM Corp. 2008, 2009 All Rights Reserved.
 *
 *   US Government Users Restricted Rights - Use, duplication or
 *   disclosure restricted by GSA ADP Schedule Contract with
 *   IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;

import java.util.Map;

/**
 * An interface that is implemented by classes that are to be invoked as part of
 * user exit routine processing.  This interface defines methods that will
 * be invoked by a protocol bridge agent to map the MQ user id of the transfer to credentials
 * that are to be used to access the protocol server.
 * There will be one instance of each implementation class per protocol bridge agent. The methods
 * can be called from different threads so the methods must be synchronized.
 */
public interface ProtocolBridgeCredentialExit {

 /**
  * Invoked once when a protocol bridge agent is started. It is intended to initialize
  * any resources that are required by the exit
  *
  * @param bridgeProperties
  *           The values of properties defined for the protocol bridge.
  *           These values can only be read, they cannot be updated by
  *           the implementation.
  *
  *
  * @return    true if the initialization is successful and false if unsuccessful
  *           If false is returned from an exit the protocol bridge agent will not
  *           start
  */
```

```
    public boolean initialize(final Map<String> bridgeProperties);

    /**
    * Invoked once for each transfer to map the MQ user id in the transfer message to the
    * credentials to be used to access the protocol server
    *
    * @param mqUserId The MQ user id from which to map to the credentials to be used
    *                 access the protocol server
    * @return         A credential exit result object that contains the result of the map and
    *                 the credentials to use to access the protocol server
    */

    public CredentialExitResult mapMQUserId(final String mqUserId);

    /**
    * Invoked once when a protocol bridge agent is shutdown. It is intended to release
    * any resources that were allocated by the exit
    *
    * @param bridgeProperties
    *             The values of properties defined for the protocol bridge.
    *             These values can only be read, they cannot be updated by
    *              the implementation.
    *
    * @return
    */

    public void shutdown(final Map<String> bridgeProperties);

}
```

Related concepts:

"Customizing WebSphere MQ Managed File Transfer with user exit routines" on page 330
You can customize the features of WebSphere MQ Managed File Transfer by using your own programs
known as user exit routines.

Related tasks:

"Mapping credentials for a file server using exit classes" on page 259
If you do not want to use the default credential mapping function of the protocol bridge agent, you can
map user credentials in WebSphere MQ Managed File Transfer to user credentials on the file server by
writing your own user exit. WebSphere MQ Managed File Transfer provides a sample user exit that
performs user credential mapping. If you configure credential mapping user exits, they take the place of
the default credential mapping function.

ProtocolBridgeCredentialExit2.java interface:

ProtocolBridgeCredentialExit2.java

```
/*
 *    Licensed Materials - Property of IBM
 *
 *    "Restricted Materials of IBM"
 *
 *    5724-H72
 *
 *    △ Copyright IBM Corp. 2011 All Rights Reserved.
 *
 *    US Government Users Restricted Rights - Use, duplication or
 *    disclosure restricted by GSA ADP Schedule Contract with
 *    IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;


/**
 * An interface that is implemented by classes that are invoked as part of user
 * exit routine processing. This interface defines methods that are invoked by a
 * protocol bridge agent to map the MQ user ID of the transfer to credentials
 * used to access a specified protocol bridge server. There will be one instance
 * of each implementation class for each protocol bridge agent. The methods can
 * be called from different threads so the methods must be synchronized.
 */
```

```
public interface ProtocolBridgeCredentialExit2 extends
  ProtocolBridgeCredentialExit {

 /**
  * Invoked once for each transfer to map the MQ user ID in the transfer
  * message to the credentials used to access a specified protocol server.
  *
  * @param endPoint
  *            Information that describes the protocol server to be accessed.
  * @param mqUserId
  *            The MQ user ID from which to map the credentials used to
  *            access the protocol server.
  * @return A {@link CredentialExitResult} instance that contains the result
  *         of the map and the credentials to use to access the protocol
  *         server.
  */
 public CredentialExitResult mapMQUserId(
    final ProtocolServerEndPoint endPoint, final String mqUserId);
}
```

Related concepts:

"Customizing WebSphere MQ Managed File Transfer with user exit routines" on page 330
You can customize the features of WebSphere MQ Managed File Transfer by using your own programs
known as user exit routines.

Related tasks:

"Mapping credentials for a file server using exit classes" on page 259
If you do not want to use the default credential mapping function of the protocol bridge agent, you can
map user credentials in WebSphere MQ Managed File Transfer to user credentials on the file server by
writing your own user exit. WebSphere MQ Managed File Transfer provides a sample user exit that
performs user credential mapping. If you configure credential mapping user exits, they take the place of
the default credential mapping function.

ProtocolBridgePropertiesExit2.java interface:

ProtocolBridgePropertiesExit2.java

```
/*
 *    Licensed Materials - Property of IBM
 *
 *    "Restricted Materials of IBM"
 *
 *    5724-H72
 *
 *    △ Copyright IBM Corp. 2011 All Rights Reserved.
 *
 *    US Government Users Restricted Rights - Use, duplication or
 *    disclosure restricted by GSA ADP Schedule Contract with
 *    IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;

import java.util.Map;
import java.util.Properties;

/**
 * An interface that is implemented by classes that are to be invoked as part of
 * user exit routine processing. This interface defines methods that will be
 * invoked by a protocol bridge agent to look up properties for protocol servers
 * that are referenced in transfers.
 * <p>
 * There will be one instance of each implementation class for each protocol
 * bridge agent. The methods can be called from different threads so the methods
 * must be synchronised.
 */
public interface ProtocolBridgePropertiesExit2 {
```

```
/**
 * Invoked once when a protocol bridge agent is started. It is intended to
 * initialize any resources that are required by the exit.
 *
 * @param bridgeProperties
 *            The values of properties defined for the protocol bridge.
 *            These values can only be read, they cannot be updated by the
 *            implementation.
 * @return {@code true} if the initialization is successful and {@code
 *            false} if unsuccessful. If {@code false} is returned from an exit
 *            the protocol bridge agent will not start.
 */
public boolean initialize(final Map<String, String> bridgeProperties);


    /**
 * Invoked when the Protocol Bridge needs to access the protocol bridge credentials XML file.
 *
 * @return a {@link String} object giving the location of the ProtocolBridgeCredentials.xml
 */
public String getCredentialLocation ();

/**
 * Obtains a set of properties for the specified protocol server name.
 * <p>
 * The returned {@link Properties} must contain entries with key names
 * corresponding to the constants defined in
 * {@link ProtocolServerPropertyConstants} and in particular must include an
 * entry for all appropriate constants described as required.
 *
 * @param protocolServerName
 *            The name of the protocol server whose properties are to be
 *            returned. If a null or a blank value is specified, properties
 *            for the default protocol server are to be returned.
 * @return The {@link Properties} for the specified protocol server, or null
 *            if the server cannot be found.
 */
public Properties getProtocolServerProperties(
  final String protocolServerName);

/**
 * Invoked once when a protocol bridge agent is shut down. It is intended to
 * release any resources that were allocated by the exit.
 *
 * @param bridgeProperties
 *            The values of properties defined for the protocol bridge.
 *            These values can only be read, they cannot be updated by the
 *            implementation.
 */
public void shutdown(final Map<String, String> bridgeProperties);

}
```

Related concepts:

"Looking up protocol file server properties by using exit classes (ProtocolBridgePropertiesExit.java)" on page 255

If you have a large number of protocol file servers, you can implement the com.ibm.wmqfte.exitroutine.api.ProtocolBridgePropertiesExit interface to look up protocol file server properties that are referenced in transfers. You can implement this interface in preference to maintaining a ProtocolBridgeProperties.xml file. You are recommended to use the ProtocolBridgePropertiesExit2.java interface but the ProtocolBridgePropertiesExit.java interface is also supported. If you have an existing implementation of the ProtocolBridgePropertiesExit.java interface from WebSphere MQ File Transfer Edition, you can use it in WebSphere MQ V7.5. The new getCredentialLocation method in ProtocolBridgePropertiesExit2.java uses the default location of the ProtocolBridgeCredentials.xml file, which is your home directory.

"Customizing WebSphere MQ Managed File Transfer with user exit routines" on page 330
You can customize the features of WebSphere MQ Managed File Transfer by using your own programs
known as user exit routines.

Related tasks:

"Mapping credentials for a file server using exit classes" on page 259
If you do not want to use the default credential mapping function of the protocol bridge agent, you can
map user credentials in WebSphere MQ Managed File Transfer to user credentials on the file server by
writing your own user exit. WebSphere MQ Managed File Transfer provides a sample user exit that
performs user credential mapping. If you configure credential mapping user exits, they take the place of
the default credential mapping function.

SourceFileExitFileSpecification.java class:

SourceFileExitFileSpecification.java

```
/*
 *    Licensed Materials - Property of IBM
 *
 *    "Restricted Materials of IBM"
 *
 *    5724-H72
 *
 *    △ Copyright IBM Corp. 2012 All Rights Reserved.
 *
 *    US Government Users Restricted Rights - Use, duplication or
 *    disclosure restricted by GSA ADP Schedule Contract with
 *    IBM Corp.
 */
package com.ibm.wmqfte.exitroutine.api;

import java.util.Map;

/**
 * A specification of the file names to use for a file transfer, as evaluated by the
 * agent acting as the source of the transfer.
 */
public final class SourceFileExitFileSpecification {

 private final String sourceFileSpecification;
 private final String destinationFileSpecification;
 private final Map<String, String> sourceFileMetaData;
 private final Map<String, String> destinationFileMetaData;

 /**
  * Constructor.  Creates a source file exit file specification.
  *
  * @param sourceFileSpecification
  *            the source file specification to associate with the source file
  *            exit file specification.
  *
  * @param destinationFileSpecification
  *            the destination file specification to associate with the
  *            source file exit file specification.
  *
  * @param sourceFileMetaData
  *            the source file meta data.
  *
  * @param destinationFileMetaData
  *            the destination file meta data   .
  */
 public SourceFileExitFileSpecification(final String sourceFileSpecification,
                                        final String destinationFileSpecification,
                                        final Map<String, String> sourceFileMetaData,
                                        final Map<String, String> destinationFileMetaData) {
  this.sourceFileSpecification = sourceFileSpecification;
```

```
  this.destinationFileSpecification = destinationFileSpecification;
  this.sourceFileMetaData = sourceFileMetaData;
  this.destinationFileMetaData = destinationFileMetaData;
}

/**
 * Returns the destination file specification.
 *
 * @return    the destination file specification.  This represents the location,
 *            on the agent acting as the destination for the transfer, where the
 *            file should be written.  Exit routines installed into the agent
 *            acting as the destination for the transfer may override this value.
 */
public String getDestination() {
 return destinationFileSpecification;
}

/**
 * Returns the source file specification.
 *
 * @return    the source file specification.  This represents the location where
 *            the file data will be read from.
 */
public String getSource() {
 return sourceFileSpecification;
}

/**
 * Returns the file meta data that relates to the source file specification.
 *
 * @return the file meta data that relates to the source file specification.
 */
public Map<String, String> getSourceFileMetaData() {
 return sourceFileMetaData;
}

/**
 * Returns the file meta data that relates to the destination file specification.
 *
 * @return the file meta data that relates to the destination file specification.
 */
public Map<String, String> getDestinationFileMetaData() {
 return destinationFileMetaData;
 }
}
```

Related concepts:

"Metadata for user exit routines" on page 921
There are three different types of metadata that can be supplied to user exit routines for WebSphere MQ
Managed File Transfer: environment, transfer, and file metadata. This metadata is presented as maps of
Java key-value pairs.

SourceTransferEndExit.java interface:

SourceTransferEndExit.java

```
/*
 *    Licensed Materials - Property of IBM
 *
 *    "Restricted Materials of IBM"
 *
 *    5724-H72
 *
 *    △ Copyright IBM Corp. 2008, 2009 All Rights Reserved.
 *
 *    US Government Users Restricted Rights - Use, duplication or
```

```
 *    disclosure restricted by GSA ADP Schedule Contract with
 *    IBM Corp.
 */
package com.ibm.wmqfte.exitpoint.api;

/**
 * An interface that is implemented by classes that want to be invoked as part of
 * user exit routine processing.  This interface defines a method that will be
 * invoked immediately after completing a transfer on the agent acting as the
 * source of the transfer.
 */
public interface SourceTransferEndExit {

 /**
  * Invoked immediately after the completion of a transfer on the agent acting as
  * the source of the transfer.
  *
  * @param transferExitResult
  *              a result object reflecting whether or not the transfer completed
  *              successfully.
  *
  * @param sourceAgentName
  *              the name of the agent acting as the source of the transfer.
  *              This is the name of the agent that the implementation of this
  *              method will be invoked from.
  *
  * @param destinationAgentName
  *              the name of the agent acting as the destination of the
  *              transfer.
  *
  * @param environmentMetaData
  *              meta data about the environment in which the implementation
  *              of this method is running.  This information can only be read,
  *              it cannot be updated by the implementation.  The constants
  *              defined in <code>EnvironmentMetaDataConstants</code> class can
  *              be used to access the data held by this map.
  *
  * @param transferMetaData
  *              meta data to associate with the transfer.  The information can
  *              only be read, it cannot be updated by the implementation.  This
  *              map may also contain keys with IBM reserved names.  These
  *              entries are defined in the <code>TransferMetaDataConstants</code>
  *              class and have special semantics.
  *
  * @param fileResults
  *              a list of file transfer result objects that describe the source
  *              file name, destination file name and result of each file transfer
  *              operation attempted.
  *
  * @return      an optional description to enter into the log message describing
  *              transfer completion.  A value of <code>null</code> can be used
  *              when no description is required.
  */
 String onSourceTransferEnd(TransferExitResult transferExitResult,
       String sourceAgentName,
       String destinationAgentName,
       Map<String, String>environmentMetaData,
       Map<String, String>transferMetaData,
       List<FileTransferResult>fileResults);


}
```

Related concepts:

"Customizing WebSphere MQ Managed File Transfer with user exit routines" on page 330
You can customize the features of WebSphere MQ Managed File Transfer by using your own programs known as user exit routines.

Related reference:

"SourceTransferStartExit.java interface"

"DestinationTransferStartExit.java interface" on page 932

"DestinationTransferEndExit.java interface" on page 931

"MonitorExit.java interface" on page 958

"ProtocolBridgeCredentialExit.java interface" on page 959

SourceTransferStartExit.java interface:

SourceTransferStartExit.java

```
/*
 *   Licensed Materials - Property of IBM
 *
 *   "Restricted Materials of IBM"
 *
 *   5724-H72
 *
 *   △ Copyright IBM Corp. 2008, 2009 All Rights Reserved.
 *
 *   US Government Users Restricted Rights - Use, duplication or
 *   disclosure restricted by GSA ADP Schedule Contract with
 *   IBM Corp.
 */
package com.ibm.wmqfte.exitpoint.api;

import java.util.List;
import java.util.Map;

/**
 * An interface that is implemented by classes that want to be invoked as part of
 * user exit routine processing.  This interface defines a method that will be
 * invoked immediately prior to starting a transfer on the agent acting as the
 * source of the transfer.
 */
public interface SourceTransferStartExit {

 /**
  * Invoked immediately prior to starting a transfer on the agent acting as
  * the source of the transfer.
  *
  * @param sourceAgentName
  *          the name of the agent acting as the source of the transfer.
  *          This is the name of the agent that the implementation of this
  *          method will be invoked from.
  *
  * @param destinationAgentName
  *          the name of the agent acting as the destination of the
  *          transfer.
  *
  * @param environmentMetaData
  *          meta data about the environment in which the implementation
  *          of this method is running.  This information can only be read,
```

```
 *              it cannot be updated by the implementation.  The constants
 *              defined in <code>EnvironmentMetaDataConstants</code> class can
 *              be used to access the data held by this map.
 *
 * @param transferMetaData
 *              meta data to associate with the transfer.  The meta data passed
 *              to this method can be altered, and the changes to will be
 *              reflected in subsequent exit routine invocations.  This map may
 *              also contain keys with IBM reserved names.  These entries are
 *              defined in the <code>TransferMetaDataConstants</code> class and
 *              have special semantics.
 *
 * @param fileSpecs
 *              a list of file specifications that govern the file data to
 *              transfer.  The implementation of this method can add entries,
 *              remove entries, or modify entries in this list and the changes
 *              will be reflected in the files transferred.
 *
 * @return    a transfer exit result object which is used to determine if the
 *              transfer should proceed, or be cancelled.
 */
 TransferExitResult onSourceTransferStart(String sourceAgentName,
      String destinationAgentName,
      Map<String, String> environmentMetaData,
      Map<String, String>transferMetaData,
      List<SourceFileExitFileSpecification>fileSpecs);
}
```

Related concepts:

"Customizing WebSphere MQ Managed File Transfer with user exit routines" on page 330
You can customize the features of WebSphere MQ Managed File Transfer by using your own programs
known as user exit routines.

Related reference:

"SourceFileExitFileSpecification.java class" on page 963

"SourceTransferEndExit.java interface" on page 964

"DestinationTransferStartExit.java interface" on page 932

"DestinationTransferEndExit.java interface" on page 931

"MonitorExit.java interface" on page 958

"ProtocolBridgeCredentialExit.java interface" on page 959

Message formats for messages you can put on the agent command queue

The following XML schemas define the formats for messages that can be put on the agent command
queue to request that the agent perform an action. The XML message can be placed on the agent
command queue by using the command-line commands or by an application.

File transfer request message format

File transfers are initiated by XML messages arriving at an agent command queue, typically as a result of a user issuing a file transfer command or by using the WebSphere MQ Explorer plug-in. The transfer request XML must conform to the FileTransfer.xsd schema and have the <request> element as the root element. The FileTransfer.xsd schema document is located in the *MQ_INSTALLATION_PATH*/mqft/samples/ schema directory. The FileTransfer.xsd schema imports fteutils.xsd, which is in the same directory.

File transfer messages can have one of following three root elements:

- <request> - for new file transfer requests, managed call requests, or deleting scheduled transfers that are pending
- <cancel> - for canceling file transfers in progress
- <transferSpecifications> - for specifying multiple transfer file groups, used by the **fteCreateTransfer** command

For information about specifying multiple transfer groups by using the <transferSpecifications> element, see Using transfer definition files.

Schema

The following schema describes which elements are valid in a transfer request XML message.

▶ V 7.5.0.1

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

<xsd:include schemaLocation="fteutils.xsd"/>

    <!--
        Defines the request of a managed transfer and version number
        <request version="1.00" ...
            <managedTransfer>
                ...
            </managedTransfer>
        </request>
    -->
    <xsd:element name="request">
        <xsd:complexType>
            <xsd:choice>
                <xsd:element name="managedTransfer" type="managedTransferType"/>
                <xsd:element name="deleteScheduledTransfer" type="deleteScheduledTransferType" />
                <xsd:element name="managedCall" type="managedCallType"/>
            </xsd:choice>
            <xsd:attribute name="version" type="versionType" use="required" />
        </xsd:complexType>
    </xsd:element>

    <!--
        Defines the cancel request of a managed transfer and version number
        <cancel version="1.00"
                xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
            <originator>
                <hostName>myMachine</hostName>
                <userID>myUserId</userID>
            </originator>       - Delete a scheduled transfer.

            <transfer>
                Transfer ID to Cancel
            </transfer>
        </cancel>
    -->
    <xsd:element name="cancel">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="originator" type="hostUserIDType" maxOccurs="1" minOccurs="1" />
                <xsd:choice>
                    <xsd:element name="transfer" type="IDType" maxOccurs="1" minOccurs="1" />
                    <xsd:element name="call"     type="IDType" maxOccurs="1" minOccurs="1" />
                </xsd:choice>
                <xsd:element name="reply"      type="replyType"      maxOccurs="1" minOccurs="0" />
```

```
            </xsd:sequence>
            <xsd:attribute name="version" type="versionType" use="required" />
        </xsd:complexType>
</xsd:element>

<!--
    Defines the transfer definition element structure.
    <transferSpecifications>
        <item ...
        <item ...
    </transferSpecifications>
 -->
<xsd:element name="transferSpecifications">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="item" type="itemType" minOccurs="1" maxOccurs="unbounded" />
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<!--
    Define a managed transfer of an instigator and request
    <managedTransfer>

        <originator>
            ...
        </originator>

        <schedule>
            <submit timebase="source"|"UTC">2008-12-07T16:07"</submit>
            <repeat>
                <frequency interval="hours">2</frequency>
                <expireTime>2008-12-0816:07</expireTime>
            </repeat>
        </schedule>

        <sourceAgent agent="here" QMgr="near" />
        <destinationAgent agent="there" QMgr="far" />

        <trigger>
        ...
        </trigger>

        <transferSet>
            ...
        </transferSet>
    </managedTransfer>
 -->

 <xsd:complexType name="managedTransferType">
     <xsd:sequence>
        <xsd:element name="originator"       type="origTransferRequestType"  maxOccurs="1" minOccurs="1"/>
        <xsd:element name="schedule"         type="scheduleType"     maxOccurs="1" minOccurs="0"/>
        <xsd:element name="sourceAgent"      type="agentType"        maxOccurs="1" minOccurs="1"/>
        <xsd:element name="destinationAgent" type="agentClientType"  maxOccurs="1" minOccurs="1" />
        <xsd:element name="trigger"          type="triggerType"      maxOccurs="1" minOccurs="0" />
        <xsd:element name="reply"            type="replyType"        maxOccurs="1" minOccurs="0" />
        <xsd:element name="transferSet"      type="transferSetType"  maxOccurs="1" minOccurs="1" />
        <xsd:element name="job"              type="jobType"          maxOccurs="1" minOccurs="0"/>
     </xsd:sequence>
</xsd:complexType>

<!--
     This is a modified form of origRequestType which is used on a managed transfer request.
     The hostName and userID are mandatory attributes in this case.
 -->
<xsd:complexType name="origTransferRequestType">
    <xsd:sequence>
        <xsd:element name="hostName" type="xsd:string" minOccurs="1" maxOccurs="1"/>
        <xsd:element name="userID" type="xsd:string" minOccurs="1" maxOccurs="1"/>
        <xsd:element name="mqmdUserID" type="xsd:string" minOccurs="0" maxOccurs="1"/>
        <xsd:element name="webBrowser" type="xsd:string" minOccurs="0" maxOccurs="1"/>
        <xsd:element name="webUserID" type="xsd:string" minOccurs="0" maxOccurs="1"/>
    </xsd:sequence>
</xsd:complexType>

<!--
    Defines the transferset of source and destination agent and one or more files
    <transferset priority="1">
```

```
                <metaDataSet>
                    <metaData key="keyname">keyvalue</metaData>
                    <metaData key="keyname">keyvalue</metaData>
                </metaDataSet>

                <item>
                    ...
                </item>
        </transferset>
 -->
<xsd:complexType name="transferSetType">
    <xsd:sequence>
        <xsd:element name="metaDataSet"        type="metaDataSetType" maxOccurs="1" minOccurs="0" />
        <xsd:element name="preSourceCall"      type="commandActionType" maxOccurs="1" minOccurs="0" />
        <xsd:element name="postSourceCall"     type="commandActionType" maxOccurs="1" minOccurs="0" />
        <xsd:element name="preDestinationCall"  type="commandActionType" maxOccurs="1" minOccurs="0" />
        <xsd:element name="postDestinationCall" type="commandActionType" maxOccurs="1" minOccurs="0" />
        <xsd:element name="item" type="itemType" maxOccurs="unbounded" minOccurs="0" />
    </xsd:sequence>
    <xsd:attribute name="priority" type="priorityType" use="optional" />
</xsd:complexType>


<!--
    Define a file pair with source and destination
    <item mode=[binary|text]>
        <source recursive="false" disposition="leave">
            <file>filename</file>
        </source>

        <destination type="file" exist="error">
            <file>filename</file>
        </destination>

    </item>
-->
<xsd:complexType name="itemType">
    <xsd:sequence>
        <xsd:element name="source"      type="fileSourceType"       maxOccurs="1" minOccurs="1" />
        <xsd:element name="destination" type="fileDestinationType"  maxOccurs="1" minOccurs="1" />
    </xsd:sequence>
    <xsd:attribute name="mode" type="modeType" use="required" />
    <xsd:attribute name="checksumMethod" type="checkSumMethod" use="required" />
</xsd:complexType>

<!--
    Defines the request to delete scheduled file transfer.
    <deleteScheduledTransfer>
        <originator>
            <delete>
                <hostName>myMachine</hostName>
                <userID>myUserId</userID>
            </delete>
        </originator>
        <ID>56</ID>
    </deleteScheduledTransfer>
 -->
<xsd:complexType name ="deleteScheduledTransferType">
    <xsd:sequence>
        <xsd:element name="originator" type="origDeleteType" maxOccurs="1" minOccurs="1" />
        <xsd:element name="ID"         type="idType"         maxOccurs="1" minOccurs="1" />
        <xsd:element name="reply"      type="replyType"      maxOccurs="1" minOccurs="0" />
    </xsd:sequence>
</xsd:complexType>


<xsd:complexType name="managedCallType">
    <xsd:sequence>
        <xsd:element name="originator"  type="origRequestType"  maxOccurs="1" minOccurs="1"/>
        <xsd:element name="agent"       type="agentType"        maxOccurs="1" minOccurs="1"/>
        <xsd:element name="reply"       type="replyType"        maxOccurs="1" minOccurs="0" />
        <xsd:element name="transferSet" type="callTransferSetType"  maxOccurs="1" minOccurs="1" />
        <xsd:element name="job"         type="jobType"          maxOccurs="1" minOccurs="0"/>
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="callTransferSetType">
    <xsd:sequence>
        <xsd:element name="metaDataSet" type="metaDataSetType" maxOccurs="1" minOccurs="0" />
```

```
            <xsd:element name="call" type="commandActionType" maxOccurs="1" minOccurs="1" />
        </xsd:sequence>
        <xsd:attribute name="priority" type="priorityType" use="optional" />
    </xsd:complexType>
</xsd:schema>
```

Understanding the transfer request message

The elements and attributes used in transfer request messages are described in the following list:

Element descriptions

`<request>`
Group element containing all the elements required to specify a file transfer request.

Attribute	Description
version	Specifies the version of this element as supplied by WebSphere MQ Managed File Transfer.

`<managedTransfer>`
Group element that contains all the elements required for a single file transfer or single group of file transfers.

`<deleteScheduledTransfer>`
Group element that contains originator and ID information to cancel a schedule transfer.

`<managedCall>`
Group element that contains all the elements required for a single managed call of a program or executable.

`<ID>`
Unique identifier that specifies the transfer request to delete from the list of pending scheduled transfers.

`<originator>`
Group element that contains the elements specifying the originator of the request.

`<hostName>`
The host name of the system where the source file is located.

`<userID>`
The user ID that originated the file transfer.

`<mqmdUserID>`
Optional. The WebSphere MQ user ID that was supplied in the message descriptor (MQMD).

`<schedule>`
Group element describing the scheduled time for the file transfer, the repeat behavior, and when the next occurrence is due.

`<submit>`
Specifies the date and time that the scheduled transfer is due to start.

Attribute	Description
timebase	Specifies which time zone to use. This attribute can have one of the following values: • source - use the time zone of the source agent • admin - use the time zone of the administrator issuing the command • UTC - use Coordinated Universal Time

Attribute	Description
timezone	The time zone description according to the timebase value

`<repeat>`
    Group element that contains details about how often a scheduled transfer repeats, how many times a scheduled transfer repeats, and when a scheduled transfer stops repeating.

`<frequency>`
    The time period that must elapse before the transfer repeats.

Attribute	Description
interval	The interval units, which must be one of the following values: • minutes • hours • days • weeks • months • years

`<expireTime>`
    Optional element that specifies the date and time that a repeating scheduled transfer stops. This element and the <expireCount> element are mutually exclusive.

`<expireCount>`
    Optional element that specifies the number of times the scheduled file transfer occurs before stopping. This element and the <expireTime> element are mutually exclusive.

`<sourceAgent>`
    Specifies the name of the agent on the system where the source file is located.

Attribute	Description
agent	Specifies the name of the agent.
QMgr	The name of the agent queue manager.

`<destinationAgent>`
    Specifies the name of the agent on the system you want to transfer the file to.

Attribute	Description
agent	Specifies the name of the agent.
QMgr	The name of the agent queue manager.
hostName	The host name or IP address of the agent queue manager.
portNumber	The port number used for client connections to the destination agent queue manager.
channel	The channel name used to connect to the destination agent queue manager.

`<trigger>`
    Optional element that specifies a condition that must be true for the file transfer to take place.

Attribute	Description
log	A flag indicating whether trigger failures are logged. The valid values are as follows: • yes - log entries are created for failed triggered transfers • no - log entries are not created for failed triggered transfers

`<fileExist>`

Specifies a comma-separated list of file names located on the same system as the source agent. If a file in this name list satisfies the condition of the trigger, the transfer occurs. This element and the <fileSize> element are mutually exclusive.

Attribute	Description
comparison	Indicates how to evaluate source file names against the name list. The valid values are as follows: • = at least one file name in the name list must match • != a minimum of one of the files in the name list does not exist
value	Indicates the comparison type: • exist: file must exist

`<fileSize>`

Specifies a comma-separated list of file names located on the same system as the source agent. If a file in this name list satisfies the condition of the trigger, the transfer occurs. This element and the <fileExist> element are mutually exclusive.

Attribute	Description
comparison	Indicates how to evaluate source file names against the name list. The valid value is as follows: • >= one of the file names in the name list exists and has a minimum size as specified in the value attribute
value	File size specified as an integer value with units specified as one of the following: • B - bytes • KB - kilobytes • MB - megabytes • GB - gigabytes (the units value is not case-sensitive)

`<reply>`

Specifies the name of the temporary reply queue generated for synchronous file transfers (specified with the **-w** parameter on the command line). The name of the queue is defined by the key **`dynamicQueuePrefix`** in the `command.properties` configuration file or the default of WMQFTE.* if not specified.

Attribute	Description
QMGR	The name of the command queue manager on which the temporary dynamic queue is generated to receive replies.

Attribute	Description
persistent	Whether the message written to the reply queue is persistent. The valid values are as follows: • true - the message is persistent • false - the message is not persistent • qdef - the persistence of the message is defined by the properties of the reply queue The default value is false.

`<transferSet>`
   Specifies a group of file transfers you want to perform together or a group of managed calls that you want to perform together. During transmission <transferSet> is a group element containing <item> elements.

Attribute	Description
priority	Priority level of the transfer. Priority is a value in the range 0-9, where 0 is the lowest priority. The default priority level is 0 and by default the transfer uses the priority level of the source agent.

`<metaDataSet>`
   Optional group element containing one or more metadata items.

`<metaData>`
   Specifies the user-defined metadata that is passed to the exit points called by the agent. The element contains the metadata value as a string.

Attribute	Description
key	Metadata name as a string

`<call>`
   Group element that contains <command> elements specifying the program or executable to call.

`<command>`
   Specifies the program or executable to call. The command must be located on the agent command path. For more information, see Table 34 on page 550. This element can contain optional <argument> elements.

Attribute	Description
name	The name of the command.
successRC	The successful return code that this command returns. Default is 0.
retryCount	The number of times that the command is to be retried if it fails.
retryWait	The time, in seconds, to wait between retries of the command.
type	The type of program to be called. The valid values are antscript, jcl, or executable.

`<argument>`
   Specifies an argument to pass to the command.

`<item>`

Group element that contains elements specifying the source and destination file names and locations.

Attribute	Description
mode	Specifies the transfer mode as either binary or text.
checksumMethod	Specifies the type of hash algorithm that generates the message digest to create the digital signature. The valid values are MD5 or none.

`<source>`

Group element that specifies files on the source system and whether they are removed after the transfer completes

Attribute	Description
recursive	Specifies that files are transferred recursively in subdirectories when the <source> element is a directory or contains wildcard characters.
disposition	Specifies the action that is taken on the <source> element when <source>has successfully been transferred to its destination. The valid values are as follows: • leave - the source files are left unchanged. • delete - the source files are deleted from the source system after the source file is successfully transferred.

`<file>`

Specifies the transfer source, which can be a file or a directory name. Use the fully qualified path in the format consistent with your operating system, for example `C:/from/here.txt`. Do not use file URIs.

Attribute	Description
alias	Specifies an alias for the source file. This alias is the name of the source file, excluding any directory path specified for the transfer.
EOL	Specifies the end of line marker for text transfers. Valid values are: • LF - line feed character only • CRLF - carriage return and line feed character sequence
encoding	The encoding of the source file for a text file transfer.
delimiterType	Specifies the type of delimiter that is included in the destination file after individual message data. The valid values is as follows: • binary - a hexadecimal delimiter This attribute is available only if you have enabled the V7.0.4.1 function.

Attribute	Description
delimiterPosition	Specifies the position to insert delimiters when writing record-oriented source file records to a normal file. The valid values are as follows: • prefix - the delimiter is inserted into the destination file before the data from each source record-oriented file record. • postfix - the delimiter is inserted into the destination file after the data from each source record-oriented file record.
includeDelimiterInFile	Specifies whether to include a delimiter between records in record-oriented source files.

\<queue\>

When used with the \<source\> element, specifies the name of the queue to transfer from, which must be located on the source agent queue manager. Use the format *QUEUE*. Do not include the queue manager name, the queue must be present on the source agent queue manager. You cannot use the \<queue\> element inside the \<source\> element, if you have used it inside of the \<destination\> element.

Attribute	Description
useGroups	Specifies whether to transfer only the first complete group of messages from the source queue. The valid values are as follows: • true - transfer only the first complete group of messages • false - transfer all messages on the source queue
groupId	Specifies the group of messages to read from the source queue. This attribute is valid only when the value of the useGroups attribute is true.
delimiterType	Specifies the type of delimiter that is included in the destination file after individual message data. The valid values are as follows: • text - a text or Java literal delimiter • binary - a hexadecimal delimiter
delimiter	Specifies the delimiter that is included in the destination file between individual message data.
delimiterPosition	Specifies whether the delimiter is included in the destination file before or after individual message data. The valid values are as follows: • prefix - the delimiter is included before the data • postfix - the delimiter is included after the data
encoding	Specifies the source queue encoding.

Attribute	Description
waitTime	Specifies the time, in seconds, for the source agent to wait for either: • a message to appear on the source queue, if the queue is empty or has become empty • a complete group to appear on the source queue, if the useGroups attribute has been set to true For information about setting the waitTime value, see "Guidance for specifying a wait time on a message-to-file transfer" on page 706.

`<destination>`

Group element that specifies the destination and the behavior if files exist at the destination agent.

You can specify only one of <file> and <queue> as a child element of destination.

Attribute	Description
type	The type of destination. The valid values are as follows: • file - specifies a file as the destination • directory - specifies a directory as the destination • queue - specifies a WebSphere MQ queue as the destination • filespace- specifies a file space as the destination The options file and directory are valid only when the **<destination>** element has a child element of **<file>**. The option queue is valid only when the **<destination>** element has a child element of **<queue>**. The option filespace is valid only when the **<destination>** element has a child element of **<filespace>**.
exist	Specifies the action that is taken if a destination file exists on the destination system. The valid values are as follows: • error - reports an error and the file is not transferred. • overwrite - overwrites the existing destination file. This attribute is not valid if the **<destination>** element has a child element of **<queue>** or **<filespace>**.

`<file>`

Specifies the transfer destination, which can be a file or a directory name. Use the fully qualified path in the format consistent with your operating system, for example `C:/from/here.txt`. Do not use file URIs.

Attribute	Description
alias	Specifies an alias for the destination file. This alias is the name of the source file, excluding any directory path specified for the transfer.
encoding	The encoding of the destination file for a text file transfer.

Attribute	Description
EOL	Specifies the end of line marker for text transfers. Valid values are: • LF - line feed character only • CRLF - carriage return and line feed character sequence

<queue>

When used with the <destination> element, specifies the name of the queue to transfer to, which can be located on any queue manager that is connected to the destination agent queue manager. Use the format *QUEUE@QM* where *QUEUE* is the name of the queue to put the messages on and *QM* is the queue manager where the queue is located. You cannot use the <queue> element inside the <destination> element, if you have used it inside of the <source> element.

Attribute	Description
delimiter	The delimiter to split the file into multiple messages.
delimiterType	Specifies the type of delimiter. The valid values are as follows: • text - a Java regular expression • binary - a sequence of hexadecimal bytes • size - a number of bytes, kibibytes, or mebibytes. For example, 1 B, 1 K, or 1 M.
delimiterPosition	Specifies whether the delimiter is expected before or after the data to include in individual messages. The valid options are as follows: • prefix - the delimiter is expected before the data • postfix - the delimiter is expected after the data
includeDelimiterInMessage	A boolean specifying whether to include the delimiters that were used to split the file into multiple messages at the end of the messages.
encoding	Specifies the destination queue encoding.
persistent	Specifies whether the messages are persistent. The valid values are as follows: • true - the messages are persistent • false - the messages are not persistent • qdef - the persistence value of the messages is defined by the settings on the destination queue
setMqProps	A boolean specifying whether WebSphere MQ message properties are set on the first message in a file, and any messages written to the queue when an error occurs.
unrecognisedCodePage	Specifies whether a text mode transfer fails or conversion is performed, if the code page of the data is not recognized by the destination queue manager. The valid values are as follows: • fail - the transfer reports a failure • binary - the data is converted to the destination code page and the WebSphere MQ message header describing the format of the data is set to MQFMT_NONE. The default behavior is fail.

<filespace>

Group element specifying the name of the file space to transfer to.

<name>

When used with the <filespace> element, the value of this element specifies the name of the file space.

<attributes>

Optional group element that contains one or more <attribute> elements to specify distribution attribute information if you are transferring files to a IBM 4690 store controller.

<attribute>

Optional element that specifies file distribution attributes. Specify either the symbolic or numeric value.

Table 62. Valid values for file distribution attributes in WebSphere MQ Managed File Transfer

Symbolic value	Numeric value	Description
DIST(LOCAL)	DIST(1)	Local file
DIST(MIRRORED,UPDATE)	DIST(2)	Mirrored file, distribute at update
DIST(MIRRORED, CLOSE)	DIST(3)	Mirrored file, distribute at close
DIST(COMPOUND,UPDATE)	DIST(4)	Compound file, distribute at update
DIST(COMPOUND,CLOSE	DIST(5)	Compound file, distribute at close

For more information about distribution attributes for WebSphere MQ Managed File Transfer on IBM 4690, see "File distribution attributes" on page 87.

<preSourceCall>

Group element specifying a command to call at the source of the transfer, before the transfer starts.

<postSourceCall>

Group element specifying a command to call at the source of the transfer, after the transfer completes.

<preDestinationCall>

Group element specifying a command to call at the destination of the transfer, before the transfer starts.

<postDestinationCall>

Group element specifying a command to call at the destination of the transfer, after the transfer completes.

<command>

When used with the <preSourceCall>, <postSourceCall>, <preDestinationCall>, or <postDestinationCall> element, this element specifies the command to be called. The command must be located on the agent command path. For more information, see Table 34 on page 550.

Attribute	Description
name	The name of the command to run.
successRC	The return code that is expected if the command runs successfully.

<argument>

When used with the <command> element, this element specifies an argument to be passed in to the command. You can have any number of <argument> elements inside a <command> element.

<job>

Optional group element containing job information for the entire transfer specification. <job> is a user-defined job name identifier that is added to the log message when the transfer has started. This

<job> element is the same as the <job> element that appears in the transfer log message, which is described in the following topic: "File transfer log message formats" on page 626.

<name>
When used with the <job> element, the value of this element specifies the name of the job.

<transferSpecifications>
Group element that contains <item> elements for multiple transfer groups. See Using transfer definition files for further details about how to use this element.

<cancel>
Group element containing all the elements required to cancel a file transfer in progress.

Attribute	Description
version	Specifies the version of this element as supplied by WebSphere MQ Managed File Transfer.

<transfer>
When used with the <cancel> element, the value of this element specifies the transfer request ID to be canceled.

<job>
Group element containing job information.

<jobName>
Specifies logical job identifier.

File transfer cancel message format

A file transfer request returns a 48-character ID that identifies the transfer for a specific agent. This ID is used to cancel transfers.

Understanding the transfer cancel message

The elements and attributes used in transfer cancel messages are described:

<cancel>
Group element containing all the elements required to cancel a file transfer in progress.

Attribute	Description
version	Specifies the version of this element as supplied by WebSphere MQ Managed File Transfer.

<originator>
Group element that contains the elements specifying the originator of the request.

<hostName>
The host name of the system where the source file is located.

<userID>
The user ID that originated the file transfer.

<mqmdUserID>
Optional. The WebSphere MQ user ID that was supplied in the message descriptor (MQMD).

<transfer>
When used with the <cancel> element, the value of this element specifies the transfer request ID to be canceled.

<job>
Optional. Group element containing job information.

\<jobName\>
   Specifies logical job identifier.

Examples

Examples of XML messages that conform to this schema are provided for each of the following requests:

- Create a file transfer
- Create an asynchronous file transfer request
- Cancel a file transfer
- Create a scheduled transfer
- Delete a scheduled transfer
- Create a managed call
- Create a file transfer that includes managed calls

Transfer request examples:

Examples of the messages that you can put on the agent command queue to request that the agent create or cancel a transfer.

Create transfer request

```
<?xml version="1.0" encoding="UTF-8"?>
<request xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
version="4.00"
xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
    <managedTransfer>
        <originator>
            <hostName>example.com.</hostName>
            <userID>mqm</userID>
        </originator>
        <sourceAgent QMgr="QM_JUPITER" agent="AGENT_JUPITER"/>
        <destinationAgent QMgr="QM_JUPITER" agent="AGENT_JUPITER"/>
        <transferSet>
            <item checksumMethod="MD5" mode="binary">
            <source disposition="leave" recursive="false">
                <file>/etc/passwd</file>
            </source>
            <destination exist="overwrite" type="directory">
                <file>/tmp</file>
            </destination>
        </item>
        </transferSet>
    </managedTransfer>
</request>
```

Create transfer request - transfer to IBM 4690

In this example XML, the file xyz.txt is set to mirrored on close when transferred to the directory c:\adx_test on a IBM 4690 store controller.

```
<?xml version="1.0" encoding="UTF-8"?>
<request version="5.00"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
  <managedTransfer>
    <originator>
      <hostName>example.com.</hostName><userID>bob</userID>
    </originator>
    <sourceAgent agent="AGENT_A" QMgr="qm_a"/>
    <destinationAgent agent="AGENT_B" QMgr="qm_b"/>
    <transferSet>
      <item mode="binary" checksumMethod="MD5">
        <source recursive="false" disposition="leave">
          <file>xyz.txt</file>
        </source>
        <destination type="directory" exist="error">
          <file>c:\adx_test</file>
```

```
            <attributes>
              <attribute>DIST(MIRRORED,CLOSE)</attribute>
            </attributes>
          </destination>
        </item>
      </transferSet>
    </managedTransfer>
</request>
```

For more information about distribution type attributes for WebSphere MQ Managed File Transfer on IBM 4690, see "File distribution attributes" on page 87.

Create transfer request - asynchronous

When a user requests a blocking synchronous request, that is, they wait for the transfer to complete and receive status messages, the message placed on the command queue contains a reply element that specifies the queue that a reply message is sent to. The following example shows the message placed on the command queue used by FTEAGENT:

```
<?xml version="1.0" encoding="UTF-8"?>
<request version="4.00"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
  <managedTransfer>
    <originator>
        <hostName>reportserver.com</hostName>
        <userID>USER1</userID>
    </originator>
    <sourceAgent agent="FTEAGENT"
                 QMgr="QM1"/>
    <destinationAgent agent="AGENT2"
                      QMgr="QM2"/>
    <reply QMGR="QM1">WMQFTE.492D0D5502770020</reply>
    <transferSet>
      <item mode="binary" checksumMethod="MD5">
        <source recursive="false" disposition="leave">
          <file>c:\sourcefiles\source1.doc</file>
        </source>
        <destination type="file" exist="overwrite">
          <file>c:\destinationfiles\dest1.doc</file>
        </destination>
      </item>
    </transferSet>
  </managedTransfer>
</request>
```

The <reply> element is populated with the name of the command queue manager where a temporary dynamic queue has been created to receive reply about the successful (or otherwise) completion of the transfer. The name of the temporary dynamic queue is composed of two parts:

- The prefix as defined by the key **dynamicQueuePrefix** in the command.properties configuration file (it is WMQFTE. by default)
- The ID of the queue as generated by IBM WebSphere MQ

Cancel transfer request

```
<?xml version="1.0" encoding="UTF-8"?>
<cancel xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        version="4.00"
        xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
    <originator>
        <hostName>example.com.</hostName>
        <userID>mqm</userID>
    </originator>
    <transfer>414D51205553322E42494E44494E47538B0F404D032C0020</transfer>
    <reply QMGR="QM_JUPITER">WMQFTE.4D400F8B20002007</reply>
</cancel>
```

Related reference:

"File transfer request message format" on page 803
File transfers are initiated by XML messages arriving at an agent command queue, typically as a result of a user issuing a file transfer command or by using the WebSphere MQ Explorer plug-in. The transfer request XML must conform to the `FileTransfer.xsd` schema and have the <request> element as the root element. The `FileTransfer.xsd` schema document is located in the *MQ_INSTALLATION_PATH*/mqft/samples/ schema directory. The `FileTransfer.xsd` schema imports `fteutils.xsd`, which is in the same directory.

Scheduled transfer message examples:

Examples of the messages that you can put on the agent command queue to request that the agent create or delete a schedule.

Create scheduled transfer

```
<?xml version="1.0" encoding="UTF-8"?>
<request xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        version="4.00"
        xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
    <managedTransfer>
        <originator>
            <hostName>example.com.</hostName>
            <userID>mqm</userID>
        </originator>
        <schedule>
            <submit timebase="admin" timezone="Europe/London">2010-01-01T21:00</submit>
        </schedule>
        <sourceAgent QMgr="US2.BINDINGS" agent="US2.BINDINGS.FILE"/>
        <destinationAgent QMgr="US2.BINDINGS" agent="US2.BINDINGS.FILE"/>
        <transferSet>
            <item checksumMethod="MD5" mode="binary">
                <source disposition="leave" recursive="false">
                    <file>/etc/passwd</file>
                </source>
                <destination exist="overwrite" type="directory">
                    <file>/tmp</file>
                </destination>
            </item>
        </transferSet>
    </managedTransfer>
</request>
```

Delete scheduled transfer

```
<?xml version="1.0" encoding="UTF-8"?>
<request xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        version="4.00"
        xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
    <deleteScheduledTransfer>
        <originator>
            <delete>
                <hostName>example.com.</hostName>
                <userID>mqm</userID>
            </delete>
        </originator>
        <ID>1</ID>
        <reply QMGR="US2.BINDINGS">WMQFTE.4D400F8B20003902</reply>
    </deleteScheduledTransfer>
</request>
```

Related reference:

"File transfer request message format" on page 803
File transfers are initiated by XML messages arriving at an agent command queue, typically as a result of a user issuing a file transfer command or by using the WebSphere MQ Explorer plug-in. The transfer request XML must conform to the FileTransfer.xsd schema and have the <request> element as the root element. The FileTransfer.xsd schema document is located in the *MQ_INSTALLATION_PATH*/mqft/samples/ schema directory. The FileTransfer.xsd schema imports fteutils.xsd, which is in the same directory.

Call request message examples:

Examples of the messages that you can put on the agent command queue to request that the agent creates a managed call or creates a transfer that calls programs.

Managed call request example

```
<?xml version="1.0" encoding="UTF-8"?>
<request xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         version="1.00"
         xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
    <managedCall>
        <originator>
            <hostName>example.com.</hostName>
            <userID>mqm</userID>
        </originator>
        <agent agent="DNWE" QMgr="QM1"/>
        <transferSet>
            <call>
                <command name="echo" successRC="0">
                    <argument>call</argument>
                    <argument>test</argument>
                </command>
            </call>
        </transferSet>
        <job>
            <name>managedCallCalls.xml</name>
        </job>
    </managedCall>
</request>
```

Managed transfer request example with calls

```
<?xml version="1.0" encoding="UTF-8"?>
<request xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         version="1.00"
         xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
    <managedTransfer>
        <originator>
            <hostName>example.com.</hostName>
            <userID>mqm</userID>
        </originator>
        <sourceAgent agent="DNWE" QMgr="QM1"/>
        <destinationAgent agent="DNWE" QMgr="QM1"/>
        <transferSet>
            <preSourceCall>
                <command name="echo" successRC="0">
                    <argument>preSourceCall</argument>
                    <argument>test</argument>
                </command>
            </preSourceCall>
            <postSourceCall>
                <command name="echo" successRC="0">
                    <argument>postSourceCall</argument>
                    <argument>test</argument>
                </command>
            </postSourceCall>
```

```
            <preDestinationCall>
                <command name="echo" successRC="0">
                    <argument>preDestinationCall</argument>
                    <argument>test</argument>
                </command>
            </preDestinationCall>
            <postDestinationCall>
                <command name="echo" successRC="0">
                    <argument>postDestinationCall</argument>
                    <argument>test</argument>
                </command>
            </postDestinationCall>
        </transferSet>
        <job>
            <name>managedTransferCalls.xml</name>
        </job>
    </managedTransfer>
</request>
```

Related concepts:

"Specifying programs to run" on page 277
You can run programs on a system where a IBM WebSphere MQ Managed File Transfer agent is running.
As part of a file transfer request, you can specify a program to run either before a transfer starts, or after
it finishes. Additionally, you can start a program that is not part of a file transfer request by submitting a
managed call request.

Related reference:

"File transfer request message format" on page 803
File transfers are initiated by XML messages arriving at an agent command queue, typically as a result of
a user issuing a file transfer command or by using the WebSphere MQ Explorer plug-in. The transfer
request XML must conform to the FileTransfer.xsd schema and have the <request> element as the root
element. The FileTransfer.xsd schema document is located in the *MQ_INSTALLATION_PATH*/mqft/samples/
schema directory. The FileTransfer.xsd schema imports fteutils.xsd, which is in the same directory.

Monitor request message formats

Resource monitors are created when a suitable XML message arrives at an agent's command queue,
typically as a result of a user issuing the fteCreateMonitor command or using the WebSphere MQ
Explorer interface.

The monitor XML must conform to the Monitor.xsd schema using the <monitor> element as the root
element.

Monitor messages can have one of the following root elements:
- <monitor> - for creating and starting a new resource monitor
- <deleteMonitor> - for stopping and deleting an existing monitor

There is no command message for the fteListMonitors command because the command directly retrieves
matching monitor definitions from the SYSTEM.FTE topic.

Schema

The following schema describes which elements are valid in a monitor request XML message.

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        targetNamespace="http://www.ibm.com/xmlns/wmqfte/7.0.1/MonitorDefinition"
        xmlns="http://www.ibm.com/xmlns/wmqfte/7.0.1/MonitorDefinition">

<xsd:include schemaLocation="FileTransfer.xsd" />

    <xsd:element name="monitor">
        <xsd:complexType>
            <xsd:sequence>
```

```
                <xsd:element name="name"            type="monitorNameType"
                            minOccurs="1"            maxOccurs="1" />
                <xsd:element name="description"      type="xsd:string"
                            minOccurs="0"            maxOccurs="1" />
                <xsd:element name="pollInterval"     type="pollIntervalType"
                            minOccurs="1"            maxOccurs="1" default="10" />
                <xsd:element name="batch"            type="batchType"
                            minOccurs="0"            maxOccurs="1" />
                <xsd:element name="agent"            type="agentNameType"
                            minOccurs="1"            maxOccurs="1" />
                <xsd:element name="resources"        type="monitorResourcesType"
                            minOccurs="0"            maxOccurs="1" />
                <xsd:element name="triggerMatch"     type="triggerMatchType"
                            maxOccurs="1"            minOccurs="1" />
                <xsd:element name="reply"            type="replyType"
                            maxOccurs="1"            minOccurs="0" />
                <xsd:element name="tasks"            type="monitorTasksType"
                            maxOccurs="1"            minOccurs="1" />
                <xsd:element name="originator"       type="origRequestType"
                            maxOccurs="1"            minOccurs="1"/>
                <xsd:element name="job"              type="jobType"
                            maxOccurs="1"            minOccurs="0"/>
                <xsd:element name="defaultVariables" type="defaultVariablesType"
                            maxOccurs="1"            minOccurs="0"/>
            </xsd:sequence>
            <xsd:attribute name="version" type="versionType" use="required" />
        </xsd:complexType>
    </xsd:element>

    <xsd:element name="deleteMonitor">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="name"       type="monitorNameType"
                            minOccurs="1"       maxOccurs="1" />
                <xsd:element name="originator" type="origRequestType"
                            maxOccurs="1"       minOccurs="1"/>
                <xsd:element name="reply"      type="replyType"
                            maxOccurs="1"       minOccurs="0" />
            </xsd:sequence>
            <xsd:attribute name="version" type="versionType" use="required" />
        </xsd:complexType>
    </xsd:element>

    <xsd:complexType name="transferRequestType">
        <xsd:choice>
            <xsd:element name="managedTransfer" type="managedTransferType" />
            <xsd:element name="managedCall"     type="managedCallType" />
        </xsd:choice>
        <xsd:attribute name="version" type="versionType" />
    </xsd:complexType>

    <xsd:complexType name="monitorResourcesType">
        <xsd:choice>
            <xsd:sequence>
                <xsd:element name="directory" type="monitoredDirectoryType"
                            minOccurs="1"    maxOccurs="unbounded" />
            </xsd:sequence>
            <xsd:element name="queue" type="monitoredQueueType"/>
        </xsd:choice>
     </xsd:complexType>

    <xsd:complexType name="monitoredDirectoryType">
        <xsd:simpleContent>
            <xsd:extension base="xsd:string">
                <xsd:attribute name="recursionLevel" type="xsd:nonNegativeInteger" />
                <xsd:attribute name="id" type="resourceIdAttrType" />
            </xsd:extension>
```

```
        </xsd:simpleContent>
    </xsd:complexType>

    <xsd:complexType name="monitoredQueueType">
        <xsd:simpleContent>
            <xsd:extension base="xsd:string">
                <xsd:attribute name="id" type="resourceIdAttrType" />
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>

    <xsd:complexType name="triggerMatchType">
        <xsd:sequence>
            <xsd:element name="conditions" type="conditionsType"
                        minOccurs="1"    maxOccurs="1" />
        </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="conditionsType">
        <xsd:choice minOccurs="1">
            <xsd:element name="allOf"     type="listPredicateType"
                        minOccurs="1"    maxOccurs="1" />
            <xsd:element name="anyOf"     type="listPredicateType"
                        minOccurs="1"    maxOccurs="1" />
            <xsd:element name="condition" type="conditionType"
                        minOccurs="1"    maxOccurs="1" />
        </xsd:choice>
    </xsd:complexType>

    <xsd:complexType name="listPredicateType">
        <xsd:choice>
            <xsd:element name="condition" type="conditionType"
                        minOccurs="1"    maxOccurs="unbounded" />
        </xsd:choice>
    </xsd:complexType>

    <xsd:complexType name="conditionType">
        <xsd:sequence>
            <xsd:element name="name"     type="conditionNameType"
                        minOccurs="0"   maxOccurs="1" />
            <xsd:element name="resource" type="resourceIdType"
                        minOccurs="0"   maxOccurs="1" />
            <xsd:choice minOccurs="1">
                <xsd:element name="fileMatch"        type="fileMatchConditionType"
                            minOccurs="1"         maxOccurs="1" />
                <xsd:element name="fileNoMatch"      type="fileNoMatchConditionType"
                            minOccurs="1"         maxOccurs="1" />
                <xsd:element name="fileSize"         type="fileSizeConditionType"
                            minOccurs="1"         maxOccurs="1" />
                <xsd:element name="queueNotEmpty"    type="queueNotEmptyConditionType"
                            minOccurs="1"         maxOccurs="1" />
                <xsd:element name="completeGroups"   type="completeGroupsConditionType"
                            minOccurs="1"         maxOccurs="1" />
                <xsd:element name="fileSizeSame"     type="fileSizeSameType"
                            minOccurs="1"         maxOccurs="1"/>
            </xsd:choice>
        </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="fileMatchConditionType">
        <xsd:sequence>
            <xsd:element name="pattern" type="conditionPatternType"
                        minOccurs="0"  default="*.*" />
            <xsd:element name="exclude" type="conditionPatternType"
                        minOccurs="0"  maxOccurs="1"/>
        </xsd:sequence>
    </xsd:complexType>
```

```xsd
<xsd:complexType name="fileNoMatchConditionType">
    <xsd:sequence>
        <xsd:element name="pattern" type="conditionPatternType"
                     minOccurs="0"  default="*.*" />
        <xsd:element name="exclude" type="conditionPatternType"
                     minOccurs="0"  maxOccurs="1"/>
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="fileSizeConditionType">
    <xsd:sequence>
        <xsd:element name="compare" type="sizeCompareType"
                     minOccurs="1"  default="0" />
        <xsd:element name="pattern" type="conditionPatternType"
                     minOccurs="0"  default="*.*" />
        <xsd:element name="exclude" type="conditionPatternType"
                     minOccurs="0"  maxOccurs="1"/>
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="sizeCompareType">
    <xsd:simpleContent>
        <xsd:extension base="xsd:int">
            <xsd:attribute name="operator" type="sizeOperatorType"  use="required" />
            <xsd:attribute name="units"    type="fileSizeUnitsType" use="required" />
        </xsd:extension>
    </xsd:simpleContent>
</xsd:complexType>

<xsd:simpleType name="sizeOperatorType">
    <xsd:restriction base="xsd:string">
        <xsd:enumeration value=">=" />
    </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="fileSizeUnitsType">
    <xsd:restriction base="xsd:string">
        <xsd:pattern value="[bB]|[kK][bB]|[mM][bB]|[gG][bB]" />
    </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="conditionPatternType">
    <xsd:simpleContent>
        <xsd:extension base="xsd:string">
            <xsd:attribute name="type"    type="patternTypeAttributeType"
                           use="optional" default="wildcard"/>
        </xsd:extension>
    </xsd:simpleContent>
</xsd:complexType>

<xsd:simpleType name="patternTypeAttributeType">
    <xsd:restriction base="xsd:token">
        <xsd:enumeration value="regex" />
        <xsd:enumeration value="wildcard" />
    </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="conditionNameType">
    <xsd:simpleContent>
        <xsd:extension base="xsd:string" />
    </xsd:simpleContent>
</xsd:complexType>

<xsd:complexType name="queueNotEmptyConditionType"/>

<xsd:complexType name="completeGroupsConditionType"/>
```

```
<xsd:complexType name="fileSizeSameType">
    <xsd:sequence>
        <xsd:element name="pattern" type="conditionPatternType"
                    minOccurs="1"  maxOccurs="1"/>
        <xsd:element name="exclude" type="conditionPatternType"
                    minOccurs="0"  maxOccurs="1"/>
    </xsd:sequence>
    <xsd:attribute name="polls" type="positiveIntegerType" use="required" />
</xsd:complexType>

<xsd:complexType name="pollIntervalType">
    <xsd:simpleContent>
        <xsd:extension base="xsd:int">
            <xsd:attribute name="units"    type="timeUnitsType"
                            use="optional"  default="minutes" />
        </xsd:extension>
    </xsd:simpleContent>
</xsd:complexType>

<xsd:complexType name="batchType">
    <xsd:attribute name="maxSize" type="positiveIntegerType" use="required"/>
</xsd:complexType>

<xsd:simpleType name="timeUnitsType">
    <xsd:restriction base="xsd:token">
        <xsd:enumeration value="seconds" />
        <xsd:enumeration value="minutes" />
        <xsd:enumeration value="hours" />
        <xsd:enumeration value="days" />
        <xsd:enumeration value="weeks" />
        <xsd:enumeration value="months" />
        <xsd:enumeration value="years" />
    </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="monitorTasksType">
    <xsd:sequence>
        <xsd:element name="task"    type="monitorTaskType"
                    minOccurs="1" maxOccurs="1" />
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="monitorTaskType">
    <xsd:sequence>
        <xsd:element name="name"        type="monitorTaskNameType"
                    minOccurs="1"      maxOccurs="1" />
        <xsd:element name="description" type="xsd:string"
                    minOccurs="0"      maxOccurs="1" />
        <xsd:element name="transfer"    type="transferTaskType"
                    minOccurs="0"      maxOccurs="1" />
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="transferTaskType">
    <xsd:sequence>
        <xsd:element name="request"  type="transferRequestType"
                    minOccurs="1"   maxOccurs="1" />
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="resourceIdType">
    <xsd:attribute name="id" type="xsd:string" use="optional" />
</xsd:complexType>

<xsd:simpleType name="resourceIdAttrType">
    <xsd:restriction base="xsd:string"></xsd:restriction>
```

```
        </xsd:simpleType>

        <xsd:simpleType name="monitorNameType">
            <xsd:restriction base="xsd:string">
                <xsd:pattern value="[^%\*]+" />
            </xsd:restriction>
        </xsd:simpleType>

        <xsd:simpleType name="agentNameType">
            <xsd:restriction base="xsd:string">
                <xsd:pattern value="[.%_0-9A-Z]*" />
            </xsd:restriction>
        </xsd:simpleType>

        <xsd:simpleType name="monitorTaskNameType">
            <xsd:restriction base="xsd:string">
                <xsd:pattern value=".*" />
            </xsd:restriction>
        </xsd:simpleType>

        <xsd:complexType name="defaultVariablesType">
            <xsd:sequence>
                <xsd:element name="variable"        type="variableType"
                            maxOccurs="unbounded" minOccurs="1" />
            </xsd:sequence>
        </xsd:complexType>

        <xsd:complexType name="variableType">
            <xsd:simpleContent>
                <xsd:extension base="xsd:string">
                    <xsd:attribute name="key" type="xsd:string" use="required" />
                </xsd:extension>
            </xsd:simpleContent>
        </xsd:complexType>

</xsd:schema>
```

Understanding the create monitor message

The elements and attributes used in create monitor messages are described:

Element descriptions

`<monitor>`
  Group element containing all the elements required to cancel a file transfer in progress.

Attribute	Description
version	Specifies the version of this element as supplied by WebSphere MQ Managed File Transfer.

`<name>`
  The name of the monitor, unique within the monitor's agent.

`<description>`
  Description of the monitor (not currently used).

`<pollInterval>`
  The time interval between each check of the resource against the trigger condition.

Attribute	Description
units	Specifies the time units for the poll interval. Valid values are: • seconds • minutes • hours • days • weeks • months • years

`<agent>`
Name of the agent the monitor is associated with.

`<resources>`
Group element that contains the elements specifying the resources to monitor.

`<directory>`
Fully qualified path specifying the directory on the monitor's agent machine to monitor.

Attribute	Description
recursionLevel	The number of subdirectories to monitor in addition to the specified directory.
id	Unique identifier for the resource.

`<queue>`
Queue name specifying the queue to monitor on the monitoring agent's queue manager.

`<triggerMatch>`
Group element that contains the elements specifying the trigger conditions to compare with the monitored resource.

`<conditions>`
Group element that contains the elements specifying the type of condition to compare with the monitored resource.

`<allOf>`
Predicate that specifies that all contained conditions must be satisfied.

`<anyOf>`
Predicate that specifies that any contained conditions must be satisfied.

`<condition>`
Defines a comparison condition that will contribute to the overall monitor trigger condition.

`<name>`
Name of the condition.

`<resource>`
Identifies the resource definition to compare the condition against.

Attribute	Description
id	Unique identifier for the resource.

If the resource that is being monitored is a directory, one of the following three elements must be specified in the condition:

- fileMatch
- fileNoMatch
- fileSize

If the resource that is being monitored is a queue, one of the following two elements must be specified in the condition:
- queueNotEmpty
- completeGroups

`<fileMatch>`
    Group element for a file name match condition.

`<pattern>`
    Specifies a file name match pattern. Files on the resource must match the pattern in order to satisfy the condition. The default pattern is * (any file will match).

`<fileNoMatch>`
    Group element for an inverse file name match condition.

`<pattern>`
    Specifies an inverse file name match pattern. If no files on the monitored resource match, the condition is satisfied. The default pattern is * (the absence of any file will match).

`<fileSize>`
    Group element for a file size comparison.

`<compare>`
    Specifies a file size comparison. The value must be a non-negative integer.

Attribute	Description
operator	Comparison operator to use. Only >=' is supported.
units	Specifies file size units, which can be one of: • B - bytes • KB - kilobytes • MB - megabytes • GB - gigabytes The units value is case insensitive, so mb' works as well as MB'.

`<pattern>`
    File name pattern to match. Default is * (any file will match).

`<queueNotEmpty>`
    This can only be specified if the resource is a queue. Specifies that there must be a message on the queue for the monitor to be triggered.

`<completeGroups>`
    This can only be specified if the resource is a queue. Specifies that there must be a complete group of messages present on the queue for the monitor to be triggered. A single transfer task is executed for each complete group on the queue.

`<reply>`
    Optional element that is used to specify reply queue for asynchronous requests.

Attribute	Description
QMGR	Queue manager name.

\<tasks\>

Group element to contain elements which specify the tasks to invoke when the monitor trigger conditions are satisfied.

\<task\>

Group element which defines an individual task that the monitor will invoke when the trigger conditions are satisfied. Currently only one task can be specified.

\<name\>

Name of the task. Accepts any alphanumeric characters.

\<description\>

Description of the task. Any text value is allowed.

\<transfer\>

Group element that defines a transfer task.

\<request\>

Group element that defines the type of task. This must contain one of the following elements which are inherited from the `FileTransfer.xsd` schema definition:

- managedTransfer
- managedCall

Attribute	Description
version	Version of the request as provided by WebSphere MQ Managed File Transfer. This is in the form n.mm where n is the major release version and mm is the minor version. For example 1.00.

\<originator\>

Group element that contains the elements specifying the originator of the request.

\<hostName\>

The host name of the system where the source file is located.

\<userID\>

The user ID that originated the file transfer.

\<mqmdUserID\>

Optional. The WebSphere MQ user ID that was supplied in the message descriptor (MQMD).

\<job\>

Group element containing job information.

\<jobName\>

Specifies logical job identifier.

\<defaultVariables\>

Group element containing one or more variable elements. These variables are used in variable substitution when monitoring a queue. For more information about variable substitution, see "Customizing tasks with variable substitution" on page 225.

\<variable\>

Element containing the value associated with the key given by the key attribute.

Attribute	Description
key	The name of the default variable.

Understanding the delete monitor message

The elements and attributes used in delete monitor messages are described:

Element descriptions

`<deleteMonitor>`
   Group element containing all the elements required to stop and delete a monitor.

Attribute	Description
version	Specifies the version of this element as supplied by WebSphere MQ Managed File Transfer.

`<name>`
   Name of monitor to delete.

`<originator>`
   Group element that contains the elements specifying the originator of the request.

`<hostName>`
   The host name of the system where the source file is located.

`<userID>`
   The user ID that originated the file transfer.

`<mqmdUserID>`
   Optional. The WebSphere MQ user ID that was supplied in the message descriptor (MQMD).

`<reply>`
   Specifies the name of the temporary reply queue generated for the request. The name of the queue is as defined by the key `dynamicQueuePrefix` in the `command.properties` configuration file. If this is not specified, the queue name has a default value of WMQFTE.

Attribute	Description
QMGR	The name of the command queue manager on which the temporary dynamic queue is generated to receive replies.

Examples

Examples of XML messages that conform to this schema are provided for each of the following monitor requests:
- Create a monitor
- Delete a monitor

Related concepts:

"Resource monitoring" on page 215
You can monitor WebSphere MQ Managed File Transfer resources; for example, a queue or a directory. When a condition on this resource is satisfied, the resource monitor starts a task, such as a file transfer. You can create a resource monitor by using the **fteCreateMonitor** command or the Monitors view in the WebSphere MQ Managed File Transfer plug-in for WebSphere MQ Explorer.

Monitor request message examples:

Examples of the messages that you can put on the agent command queue to request that the agent create or delete a monitor.

Create monitor request

```
<?xml version="1.0" encoding="UTF-8"?>
<monitor:monitor xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                 xmlns:monitor="http://www.ibm.com/xmlns/wmqfte/7.0.1/MonitorDefinition"
                 version="4.00"
                 xsi:schemaLocation="http://www.ibm.com/xmlns/wmqfte/7.0.1/MonitorDefinition ./Monitor.xsd">
    <name>EXAMPLEMONITOR</name>
    <pollInterval>1</pollInterval>
    <agent>US2.BINDINGS.FILE</agent>
    <resources>
        <directory recursionLevel="0">/srv/nfs/incoming</directory>
    </resources>
    <triggerMatch>
        <conditions>
            <allOf>
                <condition>
                    <fileMatch>
                        <pattern>*.completed</pattern>
                    </fileMatch>
                </condition>
            </allOf>
        </conditions>
    </triggerMatch>
    <reply QMGR="US2.BINDINGS">WMQFTE.4D400F8B20003702</reply>
    <tasks>
        <task>
            <name/>
            <transfer>
                <request xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                         version="4.00"
                         xsi:noNamespaceSchemaLocation="FileTransfer.xsd">
                    <managedTransfer>
                        <originator>
                            <hostName>example.com.</hostName>
                            <userID>mqm</userID>
                        </originator>
                        <sourceAgent QMgr="US2.BINDINGS" agent="US2.BINDINGS.FILE"/>
                        <destinationAgent QMgr="US2.BINDINGS" agent="US2.BINDINGS.FILE"/>
                        <transferSet>
                            <item checksumMethod="MD5" mode="binary">
                                <source disposition="leave" recursive="false">
                                    <file>/srv/nfs/incoming/*.txt</file>
                                </source>
                                <destination exist="error" type="directory">
                                    <file>/srv/backup</file>
                                </destination>
                            </item>
                        </transferSet>
                    </managedTransfer>
                </request>
            </transfer>
        </task>
    </tasks>
    <originator>
        <hostName>example.com.</hostName>
        <userID>mqm</userID>
    </originator>
</monitor:monitor>
```

Delete monitor request

```
<?xml version="1.0" encoding="UTF-8"?>
<monitor:deleteMonitor xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                       xmlns:monitor="http://www.ibm.com/xmlns/wmqfte/7.0.1/MonitorDefinition"
                       version="4.00"
                       xsi:schemaLocation="http://www.ibm.com/xmlns/wmqfte/7.0.1/MonitorDefinition ./Monitor.xsd">
    <name>EXAMPLEMONITOR</name>
    <originator>
        <hostName>example.com.</hostName>
        <userID>mqm</userID>
    </originator>
    <reply QMGR="QM_JUPITER">WMQFTE.4D400F8B20003705</reply>
</monitor:deleteMonitor>
```

Resource monitors are created when a suitable XML message arrives at an agent's command queue, typically as a result of a user issuing the fteCreateMonitor command or using the WebSphere MQ Explorer interface.

Ping agent request message format

You can ping an agent by issuing an **ftePingAgent** command or by putting an XML message on the agent command queue. The ping agent request XML must conform to the PingAgent.xsd schema. After you have installed WebSphere MQ Managed File Transfer, you can find the PingAgent.xsd schema file in the following directory: *MQ_INSTALLATION_PATH*/mqft/samples/schema. The PingAgent.xsd schema imports fteutils.xsd, which is in the same directory.

When the agent receives a ping agent request message on its command queue, if the agent is active, it returns an XML response message to the command or application that put the ping agent request message on the command queue. The response message from the agent is in the format defined by Reply.xsd. For more information about this format, see "Reply message format" on page 832.

Schema

The following schema describes which elements are valid in an ping agent request XML message.

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
          xmlns="http://www.ibm.com/xmlns/wmqfte/7.0.1/PingAgent"
          targetNamespace="http://www.ibm.com/xmlns/wmqfte/7.0.1/PingAgent">

    <xsd:include schemaLocation="fteutils.xsd"/>

    <xsd:element name="pingAgent">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="originator" type="origRequestType" maxOccurs="1" minOccurs="1"/>
                <xsd:element name="agent"      type="agentType"       maxOccurs="1" minOccurs="1"/>
                <xsd:element name="reply"      type="replyType"       maxOccurs="1" minOccurs="0" />
            </xsd:sequence>
            <xsd:attribute name="version" type="versionType" use="required" />
        </xsd:complexType>
    </xsd:element>

</xsd:schema>
```

Understanding the ping agent request message

The elements and attributes used in the ping agent request messages are described in the following list:

<pingAgent>
    Group element containing all the elements required to specify a ping agent request.

<originator>
    Group element containing all the elements required to specify the originator of the ping request.

    <hostName>
        The host name of the machine where the request originated.

    <userID>
        The user name of the originator of the request.

    <mqmdUserID>
        The MQMD user name of the originator of the request.

<agent>
    The agent to ping.

Attribute	Description
agent	Required. The name of the agent.
QMgr	Optional. The queue manager that the agent connects to.

<reply>

The name of the queue for the agent to send the reply message to.

Attribute	Description
QMGR	Required. The name of the queue manager where the reply queue is located.

Example

This example shows a ping agent message sent to the agent AGENT_JUPITER. If AGENT_JUPITER is active and able to process agent requests, it sends a response message to the queue WMQFTE.4D400F8B20003708 on QM_JUPITER.

```
<?xml version="1.0" encoding="UTF-8"?>
<ping:pingAgent xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                xmlns:ping="http://www.ibm.com/xmlns/wmqfte/7.0.1/PingAgent"
                version="4.00">
    <originator>
        <hostName>example.com.</hostName>
        <userID>mqm</userID>
    </originator>
    <agent agent="AGENT_JUPITER" QMgr="QM_JUPITER"/>
    <reply QMGR="QM_JUPITER">WMQFTE.4D400F8B20003708</reply>
</ping:pingAgent>
```

Reply message format

When an agent receives an XML message on its agent command queue, if a response is required, the agent will send an XML reply message to the reply queue defined in the original message. The reply XML conforms to the Reply.xsd schema. The Reply.xsd schema document is located in the *MQ_INSTALLATION_PATH*/mqft/samples/schema directory. The Reply.xsd schema imports fteutils.xsd, which is in the same directory.

Schema

The following schema describes which elements are valid in a reply XML message.

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

<xsd:include schemaLocation="TransferLog.xsd"/>

    <xsd:element name="reply">
     <xsd:complexType>
      <xsd:sequence>
         <xsd:element name="transferSet" type="transferSetType" minOccurs="0" maxOccurs="1" />
       <xsd:element name="status" type="statusType" minOccurs="1" maxOccurs="1" />
      </xsd:sequence>
         <xsd:attribute name="version"   type="versionType"       use="required"/>
         <xsd:attribute name="ID"        type="IDType"            use="required"/>

     </xsd:complexType>
    </xsd:element>

</xsd:schema>
```

Understanding the reply message

The elements and attributes used in the reply messages are described in the following list:

`<reply>`
Element containing the elements that specify the reply information.

Attribute	Description
ID	The ID of the reply.
version	The version of the reply message format.

`<status>`
The status of the action that the agent was requested to perform.

Attribute	Description
resultCode	The result code returned from the action that the agent performed.

`<supplement>`
Additional response information about the action that the agent was requested to perform.

Example

In the following section is an example reply message:

```
<reply version="1.00"          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                               xsi:noNamespaceSchemaLocation="Reply.xsd"
                               ID="010202030000000000000000000000000000000000000000">
    <status resultCode="65">
        <supplement>Additional reply information</supplement>
    </status>
</reply>
```

WebSphere MQ Managed File Transfer diagnostic messages

Diagnostic messages are available here in numerical order, grouped according to the part of Managed File Transfer from which they originate.

For details of these messages, see IBM Knowledge Center: https://www.ibm.com/support/knowledgecenter/SSFKSJ_7.5.0/com.ibm.wmqfte.doc/messages_main.htm

Index

X

XML message format
  audit   626, 656, 767, 796
  command   803, 968
  monitor log   656, 796
  transfer log   626, 767
  transfer request   803, 968

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Programming interface information

Programming interface information, if provided, is intended to help you create application software for use with this program.

This book contains information on intended programming interfaces that allow the customer to write programs to obtain the services of WebSphere MQ.

However, this information may also contain diagnosis, modification, and tuning information. Diagnosis, modification and tuning information is provided to help you debug your application software.

Important: Do not use this diagnosis, modification, and tuning information as a programming interface because it is subject to change.

Trademarks

IBM, the IBM logo, ibm.com®, are trademarks of IBM Corporation, registered in many jurisdictions worldwide. A current list of IBM trademarks is available on the Web at "Copyright and trademark information"www.ibm.com/legal/copytrade.shtml. Other product and service names might be trademarks of IBM or other companies.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

This product includes software developed by the Eclipse Project (http://www.eclipse.org/).

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Sending your comments to IBM

We appreciate your input on this publication. Feel free to comment on the clarity, accuracy, and completeness of the information or give us any other feedback that you might have.

Use one of the following methods to send us your comments:
- Send an email to ibmkc@us.ibm.com
- Use the form on the web here: www.ibm.com/software/data/rcf/

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate, without incurring any obligation to you.

Include the following information:
- Your name and address
- Your email address
- Your telephone or fax number
- The publication title and order number
- The topic and page number related to your comment
- The text of your comment

IBM or any other organizations will only use the personal information that you supply to contact you about the issues that you submit.

Thank you for your participation.