WebSphere. WebSphere Message Broker

Version 6 Release 0

Replay Server
SupportPac IO03

WebSphere® WebSphere Message Broker

IBM

Version 6 Release 0

Replay Server
SupportPac IO03

Second Edition (January 2006)

This edition applies to IBM WebSphere Business Integration Message Broker Version 5.0 and to all subsequent releases and modifications until otherwise indicated in new editions.

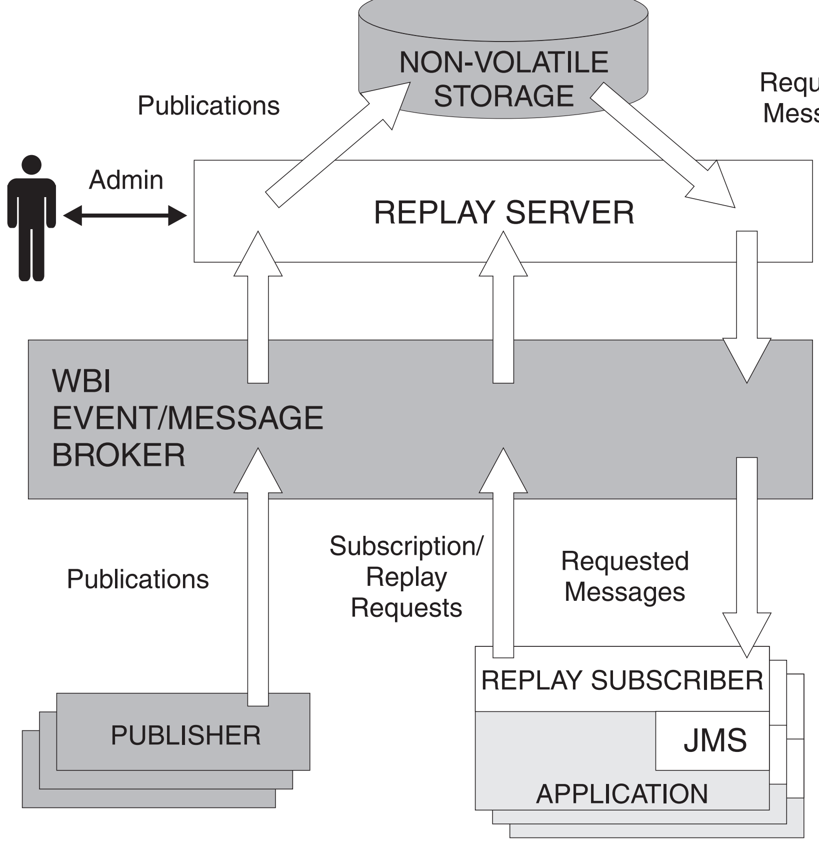
Contents

Introduction

Note: Throughout this document, references to WebSphere Message Broker should be assumed to apply also to WebSphere Event Broker.

The Replay Server provides an extension to publish/subscribe applications that use the IBM® WebSphere® Message Broker by allowing application programs to receive published messages, not just when they are published, but whenever they want.

The following diagram shows how the Replay Server interacts with the WebSphere Message Broker and its publishers and replay subscribers. It also shows the flow of publications, subscriptions, replay requests and requested messages between the components of a Replay system.



The Replay Subscriber subscribes to a range of topics on the broker. When a message is published on one of these topics, the message is saved by the Replay Server in non-volatile storage (which might be a database such as DB/2). Each message has a time stamp and a sequence number added to it when it is saved. The sequence number is unique for each message that is published on a specific topic.

An Application Programming Interface (API) has been defined for the Replay Subscriber. This API allows Java Message Service (JMS) applications to subscribe to the published messages (publications) that the Replay Server has stored. The API allows an application to request a specified topic or range of topics.

Publications that satisfy one or more of the following criteria can be requested:
- Publications that have been published since a specific time.
- Publications that have sequence numbers in a specified range.
- Publications that have not yet been published.

Messages (publications) are then sent (or replayed) to Replay Subscribers when required.

Use the Pruner program to remove from the non-volatile storage (or database) messages that are no longer required. Operations that are performed by the Pruner do not affect the normal operations of the Replay Server.

Detailed information about the Replay Server is found in the following parts of this document:
- "Getting started" on page 3 describes publish/subscribe applications and discusses what the message replay server does, how it does it, and why you might want to use it.
- "Installing Replay" on page 11 describes the hardware and software that you need. It also describes how to install a Replay Server and how to change a Java Message Service (JMS) client so that it can replay messages.
- "Uninstalling Replay" on page 15 describes how to uninstall a Replay Server and remove replay capability from a JMS client.
- "Configuring Replay" on page 17 describes how to configure the Replay environment.
- "Administering Replay" on page 23 describes how to operate and administer a Replay Server.
- "Developing applications" on page 29 describes how to develop new Replay Client applications that use the extensions that are provided to the JMS API for Replay.
- "Problem diagnosis" on page 35 describes how to diagnose problems in a Replay configuration.
- "Command reference" on page 37 provides reference information for the commands that you use to operate and administer a Replay Server.
- "Programming reference" on page 51 provides reference information for the extensions that are provided to the JMS API for Replay.

Related information

- See the WebSphere Message Broker Information Center for information about publish and subscribe.
- See the manual *WebSphere MQ Using Java SC34-6066* for information about the JMS programming interface for publish and subscribe.

Getting started

The following sections describe the Replay Server, and how you can use it:

- “Publish/subscribe” describes the publish/subscribe style of messaging.
- “Replay” on page 5 describes what the Replay Server does and gives some examples of why you might use it.
- “How Replay works” on page 6 describes how the Replay Server works, and how you can use it.
- “Configuration options” on page 9 describes some configuration restrictions.

Publish/subscribe

Publish/subscribe is a style of messaging application in which the applications that provide information are de-coupled from the applications that might use that information.

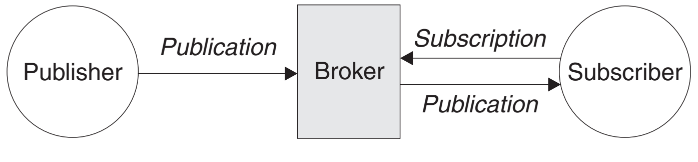
- A *publisher* is an application that provides information in a publish/subscribe system.
- A *subscriber* is an application that uses the information in a publish/subscribe system.
- The information that a publisher provides in a publish/subscribe system is known as a *publication*.
- The request that a subscriber makes for information in a publish/subscribe system is known as a *subscription*.

In a publish/subscribe system, a publisher does not need to know who uses the information that it provides, and a subscriber does not need to know who provides the information that it uses.

Message brokers make sure that messages arrive at the correct destinations, and are transformed to the format required at each destination.

The simplest form of a publish/subscribe system has one message broker, one application that publishes messages, and one application that subscribes to messages.

The following figure shows a simple publish/subscribe application. There is one publisher, one broker, and one subscriber. A publication is sent from the publisher to the broker, a subscription is sent from the subscriber to the broker, and a publication is sent from the broker to the subscriber.



However, a typical publish/subscribe system has more than one publisher and more than one subscriber, and often more than one broker too.

Note that an application can be both a publisher and a subscriber.

The publisher generates a message that it wants to publish on a topic and sends it to an input node on the broker. A message flow running in the broker retrieves the message from its input node and passes the message to a Publication node for distribution to a subscriber.

The input node might be one of the following built-in nodes:
- MQInput which represents a WebSphere MQ queue
- Real-timeInput which receives messages from a JMS application using WebSphere MQ Real-time Transport
- MQeInput which represents a WebSphere MQ Everyplace queue
- SCADAInput which represents a SCADA input port

Real-timeInput using WebSphere MQ Real-time Transport achieves much better performance than input using MQ queues, but has significant restrictions as described in "Subscriptions."

A subscriber registers a request for a publication by specifying the topic, or topics, of the published messages that it is interested in.

Topics

A *topic* is a character string that describes the nature of the data that is being published in a publish/subscribe system.

Topics are key to the successful delivery of messages in a publish/subscribe system. Instead of including a specific destination address in each message, a publisher assigns a topic to the message. The message broker matches the topic with a list of clients who have subscribed to that topic, and delivers the message to each of those clients.

A topic string can include any character from the Unicode character set, including the space character. However, there are three characters that have special meanings. These characters are described in "Special characters in topics" on page 38. A subscriber can specify a topic string containing these special "wild card" characters to match publications on a range of topics.

Restriction: Do not use '?' in topic names that are used by the Replay Server.

Topic strings that start with the characters "$Sys" are reserved for IBM use. The message replay function exploits such reserved topics for communication through WebSphere Business Integration Message Broker.

Subscriptions

There are two types of subscriber: durable subscribers and nondurable subscribers. If a subscriber needs to receive all the messages that are published on a topic, including the ones that are published while the subscriber is inactive, it uses a durable subscription. The message broker retains a record of this durable subscription and makes sure that all messages from the topic's publishers are retained until they are acknowledged by this durable subscriber or have expired. A nondurable subscriber only receives publications while the subscriber is active.

The high performance WebSphere MQ Real-time Transport only supports nondurable subscribers. However, you can use the Replay Server to achieve most of the value of durable subscriptions while still using the WebSphere MQ

Real-time Transport, but more advanced coding than basic JMS is required. Refer to “Java programming interfaces” on page 51 for more information.

Replay

Note: Throughout this section, when we refer to a broker, assume that the broker belongs to the WebSphere Message Broker product.

Function

The Replay Server captures selected published messages from a broker, stores them in non-volatile storage, and provides mechanisms for publish/subscribe clients to subsequently receive some or all of these stored messages, whenever and as frequently as required.

For message capture, the Replay Server subscribes to a broker using an administratively defined topic list. Any of the topic strings that are specified in this topic list can contain one or more wildcard characters so that published messages on a range of specific topics can be selected.

Wildcard characters in topic strings are described in “Special characters in topics” on page 38.

For each message that it captures, the Replay Server appends both a time stamp and a sequence number before storing the message. The time stamp represents the date and time that the message is captured. The sequence number relates to the topic that is associated with the message that is captured; each time that the Replay Server captures a message, it increments the current message sequence number for the topic that is associated with that message. These time stamps and sequence numbers can be used by a subscriber client to specify which messages it wants to receive.

The Replay Server stores the captured messages in a DB2 database. The non-volatile storage that is provided by this DB2 database is known as the persistent store.

To prevent the persistent store from becoming full, the Replay Server provides an administrative interface that can be used to remove, from the persistent store, messages that meet specified criteria. This pruning of messages from the persistent store can start immediately, or at specified times, or after specified time intervals.

Subscriber client applications can initiate replay from the Replay Server. They specify timestamp values or message sequence numbers to select the start and end points for the replay; this selection can be for messages that have already been captured, or for messages that will be received and captured in the future, either up to some specified time or sequence number, or indefinitely. They can also specify the topics of messages of interest, and request, for example, the replay of every Nth message that meets the other criteria for message selection. These subscriber applications can be written in Java, using proprietary extensions to the Java Message Service (JMS) programming interface.

In this release, existing unchanged JMS applications cannot usefully use Replay.

Examples that show the use of the Replay Server

Here are some examples that show how and why you might use the Replay Server:

- Subscriber catch up.

  You might have a trading application that uses publish/subscribe to receive stock market data. If this application is not always available, or if the trader who uses the application is not always present, then the application can use Replay to receive relevant messages which were published while the application was unavailable, and to then continue to receive new messages as they are captured by the Replay Server.

  This is somewhat like a durable subscription but with the benefit that it works with the high performance MQ Real-time protocol and with the disadvantage that messages might be lost if the Replay Server fails before the message has been asynchronously written to the DB2 database.
- Sampling with replay of interesting data.

  You might have an application that analyzes sensor data related to geological events. To minimize data volumes you might want to sample this sensor data sufficiently often to spot an interesting event, such as an earthquake, but not often enough to fully analyze that event. Having spotted the interesting event, your application can use Replay to receive all the detailed messages that are published shortly before and shortly after that event.
- Application testing.

  You might want to test and improve a new application using real world data. You can use Replay to capture a typical sample of this real world data, which your test applications can then repeatedly receive. Your test applications can choose to receive the captured messages either at a fixed arrival rate unrelated to their real world arrival times, or with built-in delays so that they arrive with the same timing pattern as in the real world.
- Problem diagnosis.

  You can use a Replay Server to capture all messages during normal operation. If a problem arises, you can start your problem diagnosis tools and then replay the relevant captured messages to obtain the appropriate problem diagnosis data.

How Replay works

This section contains short descriptions of the major components of a Replay system, and also describes how the Replay Server works.

- "The persistent store" on page 7 describes how captured messages and other data are stored in non-volatile storage.
- "Capturing messages" on page 7 describes how you specify which messages are captured and how they are tagged for selection for replay.
- "Removing messages" on page 7 describes why messages need to be periodically removed from the persistent store; it also describes how you can specify which messages are removed, and when removal takes place.
- "Subscribing to a Replay Server" on page 8 describes how publish/subscribe clients can specify criteria for the messages that they want to receive from the Replay Server, and how they receive these messages.
- "Support for existing applications" on page 8 describes how existing, unchanged JMS publish/subscribe client applications can use Replay.
- "Administering a Replay Server" on page 9 describes how you can administer the Replay Server.

- "Security" on page 9 describes how the security of messages is safeguarded.

The persistent store

The persistent store retains captured messages, together with associated control data, in non-volatile storage so that the messages can be replayed at a future time. After a message is replayed, it is not removed from the persistent store. Therefore, you can replay a message as often as you want. The only way to remove a message from the persistent store is to use the pruner process that is described in "Removing messages."

The Replay Server asynchronously writes captured messages into non-volatile storage. Messages are held in memory during periods of high activity and moved into non-volatile storage during periods of lower activity. This maximizes performance at the risk of message loss if the Replay Server fails before that message has been written to non-volatile storage.

The Replay Server can also be configured to discard messages from its internal memory store if the period of high activity causes an unacceptable buildup of messages for storing.

Capturing messages

As described in "The changeproperties and reportproperties commands" on page 40, you can use the **changeproperties** command to set the **collectorstring** property to capture messages whose topics match a specified list of topics. This list of topics can include a single topic or multiple, perhaps unrelated, topics. Each topic in the list can include wildcard characters so that it can match a range of topics. Any change to the **collectorstring** property replaces the previous list of topics with the new list, and causes the Replay Server to unsubscribe from the topics in the previous list and subscribe to the topics in the new list.

The Replay Server appends a time stamp and a sequence number to each message that it captures. The time stamp records the local date and time that the message was captured. The sequence number records the absolute sequence number of this message within the set of messages which this Replay Server has captured on the topic associated with this publication. The Replay Server has a separate sequence number for each topic, and increments this sequence number each time it captures a message that is associated with that topic. It then stores this sequence number with the message. It stores each sequence number as a 64-bit integer, which means that a new sequence number can be allocated every microsecond for several thousand years before the sequence number wraps.

Each time a sequence number is incremented, it is saved in the persistent store. If the Replay Server fails, when the persistent store is restarted, the last assigned sequence number for each topic is recovered from the persistent store.

As described in "Replay security" on page 25, you can use the **reportproperties** command to display the current value of the **collectorstring** property to see the list of topics to which the Replay Server is currently subscribing.

Removing messages

When Replay is being used, there is a constant flow of captured messages into the persistent store. This growth of the number of messages in the persistent store will gradually degrade Replay performance and ultimately stop Replay when all available data sets are full. Therefore, you must periodically remove messages from the persistent store.

Use the **startpruner** command, described in "The pruner commands" on page 44, to initiate the asynchronous removal of messages from the persistent store. You can specify a threshold number of messages in the persistent store so that, when this number is exceeded, messages are removed.

You can choose which messages are to be removed. For example, you can remove a specified number, or a specified percentage, of the oldest messages; you can also choose to remove only those messages that are associated with specific topics. You can request an immediate test of the message removal threshold or a periodic test of the message removal threshold which can be repeated at a specified time interval, or at specified times of the day. If a test shows that the message threshold is exceeded, messages are removed from the persistent store using the specified message removal strategy.

Note: The last message on any topic is never removed.

Subscribing to a Replay Server

A replay subscriber client connects to a broker in the same way as any other publish/subscribe client. The Replay Server also connects to a broker. The Replay component of the Replay Server and the Replay subscriber client communicate with each other through the broker.

A Replay subscriber client application creates a ReplaySignature to specify the messages that it wants the Replay Server to replay for that client.

You can set the following data in the ReplaySignature:

- The start point of the required replay; you can specify either a time stamp or a message sequence number, or you can specify that the replay starts now.
- The end point of the required replay; you can specify either a time stamp or a message sequence number, or you can specify that the replay continues indefinitely.
- The time interval between the replay of successive messages; you can specify either the original interval or a specified minimum interval ('throttled'), or you can specify that messages are replayed as quickly as possible.
- A filter that specifies, for example, that every Nth message is replayed, or that messages that are separated by a specified time interval are replayed.

A replay subscriber application passes the above ReplaySignature, together with a topic string and an optional JMS message selector, to a method that creates a new replay subscriber request.

Support for existing applications

You can use new ConnectionFactory options to cause an existing, unchanged, JMS subscriber application to receive messages from the near real time feed of your Replay Server. However this uses a default ReplaySignature that causes the replay to start now, continue indefinitely, replay with the original timing, and with no filtering of messages. The only benefit you gain from this, over and above direct use of the message broker, is that you can be sure that your application only sees messages which have been captured by a Replay Server. Such messages can thus be replayed by a new client application if the need arises.

Administering a Replay Server

As described in "Starting and stopping Replay Server administration" on page 23, you can locally administer your Replay Server from a local administration window, or you can remotely administer it from any machine that can run a JMS client and that is connected to your broker network. The administrative client application uses JMS publish/subscribe to communicate with your Replay Server.

The administrative interface provides the following commands:

- **start** to start all Replay Server components.
- **stop** to stop all Replay Server components except the pruner and the administration server.
- **changeproperties** to change the value of one or more specified properties in the Replay Server configuration file.
- **reportproperties** to report the current configuration, or of one or more specified properties, in the Replay Server configuration file.
- **help** to display a list of available commands with details of their syntax.
- **ping** to verify that the replay administration client can communicate with the Replay Server.
- **startpruner** to set the properties for the pruner and to start the pruner component of the Replay Server.
- **stoppruner** to stop the pruner component of the Replay Server.
- **prunerstatus** to display statistics for the pruner component of the Replay Server.

The configuration file contains properties that control the behavior of the Replay Server. A default configuration file is provided for each newly created Replay Server. You can use the **changeproperties** command to change any of these properties; these changes persist across any shutdown and restart of the Replay Server.

Security

The Replay Server uses the services of WebSphere Message Broker to enforce topic security.

For message capture the WebSphere Message Broker checks that the UserId associated with the Replay Server collector component is authorized to subscribe to the specified topics.

For Replay, the replay subscriber client code temporarily subscribes to the original topic to cause the WebSphere Message Broker to check that this subscriber is authorized to subscribe to that topic.

Configuration options

This release has the following restrictions on how you can configure your Replay Server:

- Your Replay Server must run in the same operating system image as the WebSphere Message Broker. This means that your Replay Server uses the DB2 database that is already installed on this operating system image as a prerequisite for your WebSphere Message Broker.
- You can have only one Replay Server in a network of connected brokers such as a broker collective. Install your Replay Server on the most powerful platform within this broker collective.

Replay clients can attach to any broker within the collective to access the Replay Server. Within the Replay Server, the Collector and Replayer components do not have to use the same broker in the collective; for example, separate brokers can be used for inbound and outbound operations. See “Configuring Replay Server communication” on page 20 for more information about how to do this.

- You can have only one Replay Server on any machine.

Installing Replay

The following sections describe how to install message replay:

- "Prerequisite hardware and software" describes the platforms that support the Replay Server and their prerequisite software products.
- "Installing a Replay Server" on page 12 describes how to install an instance of a Replay Server.
- "Installing a Replay Client on a JMS client" on page 13 describes how to upgrade a JMS client to include message replay support.

Prerequisite hardware and software

General

The Replay function requires Version 5.0, Version 5.1, or Version 6.0 of WebSphere Message Broker.

Replay server

You can install a Replay Server on any of the following operating systems:

Operating system	Software	Hardware
AIX	AIX Version 5.1 with Maintenance Level 3 plus PTF U476951 or Maintenance Level 4 AIX Version 5.2 with Maintenance Level 1 plus PTF for APAR IY44 922	IBM e(logo)server™ pSeries™ IBM RS/6000® processor machines
Linux® on Intel®	Linux Intel (IA32) RedHat Advanced Server Version 2.1 Linux Intel (IA32) SuSE Linux Enterprise Server (SLES) Version 8 (8.1 GA or Service Pack 2a) Linux Intel (IA32) United Linux 1.0 plus Service Pack 2a	IBM e(logo)server xSeries® or equivalent Intel based systems
Solaris	Solaris 8 or 9 with SunSolve recommended patch level	Sun Microsystems SPARC processor machines
Windows	Windows 2000 Professional, Server, and Advanced Server with Service Pack 3 or Service Pack 4 Windows XP Pro with Service Pack 1 Windows 2003	IBM e(logo)server xSeries or equivalent Intel based system IBM e(logo)server iSeries™ Server using the IBM Integrated xSeries Server

A Replay Server needs the following levels, or higher levels, of the following prerequisite software products.

Because your Replay Server must be installed on the same node as your WebSphere Message Broker, you will already have these products because they are also required by your WebSphere Message Broker:

- IBM DB2® Universal Database Enterprise Server Edition 7.2 FixPak 9 or IBM DB2 Universal Database™ Enterprise Server Edition 8.1 FixPak 2.
- IBM WebSphere MQ Version 5.3.0.4 (V5.3 with Fix Pack 04) or Version 5.3.0.5 (V5.3 with Fix Pack 05).

  Note: If you are running publish/subscribe applications that use WebSphere MQ Real-time Transport , you might experience problems in a heavily loaded system. Many of these problems can be solved by installing WebSphere MQ Version 5.3 Fix Pack 5. For more details, contact your IBM Support center.
- Java Runtime Environment 1.4.1 (service release 3 for AIX® only).

  On AIX, ensure that you have installed Java 1.4.1 Service Release 3 in directory /usr/java141 . If you have installed Java1.4.1 in another directory, create a symbolic link in /usr/java141 to the directory in which you installed Java 1.4.1.
- Microsoft Data Access Components (MDAC) Version 2.7 SP1, Version 2.7 SP1a, or Version 2.8.

Replay client

You need the following software to run a JMS Replay client:

- Java Development Kit, Java Runtime Environment (JRE), or Java-enabled Web browser for the client operating system
- For z/OS and OS/390, OS/390 Version 2 Release 9 or higher, or z/OS, with UNIX System Services (USS)
- For OS/400, the iSeries(TM) Developer Kit for Java, 5769-JV1, and the Qshell Interpreter, OS/400 (5769-SS1) Option 3

The following list shows the supported Java 2 Software Development Kits and Java Runtime Environments:

- IBM Developer Kit for AIX, Java Technology Edition, Version 1.4.1
- IBM Developer Kit for Linux, Java Technology Edition, Version 1.4.1
- IBM Developer Kit for OS/390, Java Technology Edition, Version 1.4.1
- IBM Developer Kit for Windows, Java Technology Edition, Version 1.4.1
- IBM iSeries Developer Kit for Java, Version 1.4
- Java 2 Standard Edition, for the Solaris Operating Environment, SDK 1.4.1

To fully support Secure Socket Layer (SSL) authentication, you need a Java Runtime Environment at Version 1.4.1 for your platform. SSL support enables your WebSphere MQ Java Message Service (JMS) applications to benefit from secure connection to the broker, providing authentication, message integrity, and data encryption.

Installing a Replay Server

You must install your Replay Server on the same operating system image as a WebSphere Message Broker. This means that your Replay Server can use the same DB2 database as your WebSphere Message Broker. However, the Java runtime environment must be Version 1.4 or higher.

Extract the io03.zip file, or untar the io03.tar.Z file, into your chosen installation directory and add the following files at the beginning of the system CLASSPATH:

```
<install-dir>/replayServer.jar
<install-dir>/aspectjrt.jar
```

where <install-dir> is the name of your installation directory.

Installing a Replay Client on a JMS client

This describes the tasks to add the Replay Client code to an existing WebSphere MQ Java Message Service (JMS) client. You must have previously installed the JMS client code as described in the manual *WebSphere MQ Using Java SC34-6066*.

Included in the io03.zip or the io03.tar.Z file is a file io03_client.zip or io03_client.tar. Copy this file onto the client machine, extract or untar into your chosen installation directory, and add the following files at the beginning of the system CLASSPATH:

```
<install-dir>/replayClient.jar
<install-dir>/aspectjrt.jar
```

where <install-dir> is the name of your installation directory.

Uninstalling Replay

The following sections describe how to uninstall Replay:

- "Uninstalling a Replay Server" describes how to uninstall an instance of a Replay Server.
- "Uninstalling a Replay Client from a JMS client" describes how to remove Replay Support from a JMS client.

Uninstalling a Replay Server

Delete the install directory and remove the files from the CLASSPATH.

Uninstalling a Replay Client from a JMS client

Delete the install directory and remove the files from the CLASSPATH.

Configuring Replay

The following sections describe how to configure your Replay environment:

- "Configuring DB2" describes how to configure the DB2 database that is used to store captured messages.
- "Configuring Replay Server resources" on page 19 describes how to configure a Replay Server for optimum performance.
- "Configuring Replay Server communication" on page 20 describes how to configure the communication link between your Replay Server and the broker to which the Replay Server connects.
- "Configuring Replay Client communication" on page 20 describes how to configure communication links between Replay clients and the broker to which they connect.

Configuring DB2

Your Replay Server uses DB2 to store captured messages for possible replay. This section describes how to configure and secure that DB2 database, including its security requirements. It assumes that you are familiar with how to perform basic DB2 administration.

The default size of the message content is 32K, and no topic name can be longer than 500 characters. The message is not stored in the database if these values are exceeded.

1. (Optional) You can change the appropriate DDL file to allow larger values for the size of the message, or the length of the topic name, but you must do this before you configure your database. The following steps are required:
   a. Open the appropriate DDL file in a text editor and locate the lines `"CONTENT" BLOB(32768) NOT LOGGED NOT COMPACT` for the BLOB size, and `"TOPIC_NAME" VARCHAR(500) NOT NULL` for the length of the topic name.
   b. Change BLOB(32768) and VARCHAR(500) to the values that you want.
   c. Save and close the DDL file.
2. Now run the IBM programs, specific to your operating system, to create the necessary DB2 tables and other resources for your Replay Server. Depending on your security policy, you can either use the userid and password of the DB2 instance owner, or you can create either a single user account for both the Replay Server storage and pruner components to use, or a separate account for each of these components, and set up the necessary DB2 privileges for these accounts. The instance owner already has all the required privileges to run the Replay Server storage and pruner components.
3. (Optional) For Windows, create the DB2 tables that are used to store captured messages:
   a. Save the supplied replay_windows.ddl file to a temporary location; you can find this file in the DDL/db2 folder.
   b. Open a DB2 Command Window and navigate to the folder that contains the replay_windows.ddl file.
   c. Run the command db2 -tf replay_windows.dll.
4. (Optional) For Linux, AIX, or Solaris, create the DB2 tables that are used to store captured messages:

   a. Save the supplied replay_unix.ddl file to a temporary location; you can find this file in the DDL/db2 folder.
   b. Log on as the DB2 instance owner.
   c. Open a console window and navigate to the temporary location that contains the replay_unix.ddl file.
   d. Run the command db2 -tvf replay_unix.ddl.
5. Bypass this step if you plan to run the Replay Server storage and pruner components under the Userid and password of the DB2 instance owner. Otherwise, either create a single DB2 account for both the storage and pruner components, or create separate DB2 accounts for the storage and pruner components.
6. Bypass this step if you plan to run the Replay Server storage and pruner components under the Userid and password of the DB2 instance owner. Otherwise, authorize the above user accounts to access the appropriate DB2 tables:
   a. For Windows, open a DB2 command line processor window. For Linux, AIX, or Solaris, open a console window.
   b. Enter the command CONNECT TO REPLAY.
   c. Enter the command GRANT SELECT,DELETE ON REPLAY.PRUNERVIEW TO USER <PrunerUserName>.
   d. Enter the command GRANT SELECT ON REPLAY.MESSAGE_COUNT TO USER <PrunerUserName>.
   e. Enter the command GRANT SELECT, INSERT ON REPLAY.TOPIC_LIST TO USER <ReplayUserName>.
   f. Enter the command GRANT SELECT ON REPLAY.HIGHESTSEQUENCENUMBER TO USER <ReplayUserName>.
   g. Enter the command GRANT INSERT ON REPLAY.MESSAGE_DATA TO USER <ReplayUserName>.
7. To connect to the database, make sure that the db2jcc.jar and db2jcc_license_cu.jar files, which are usually located in the <db2install>/SQLLIB/java directory, are on the CLASSPATH. These files contain the JDCB drivers and the appropriate license information.
8. If the database is running on a different machine from the Replay Server, you must configure the database instance to accept incoming TCP/IP connections:
   a. For Windows, open a DB2 command window.
   b. For Linux, AIX, or Solaris, open a console window.
   c. Enter the command db2set DB2COMM=tcpip.

The following properties are associated with the database:
- storage_url
- storage_portnumber
- storage_password
- storage_userid

Refer to "Setting and displaying Replay Server properties" on page 24 for details of setting these parameters.

Configuring JMS

Configure the Real-time transport of WebSphere Message Broker.

You can change the appropriate DDL file to allow larger values for high performance by entering the following commands (the user must havre the appropriate mqbrkrs group membership) from the command line:

```
mqsichangeproperties <Broker Name> -o Dynamic Subscription -n statsInterval -v 1000
mqsichangeproperties <Broker Name> -o Dynamic Subscription
                                  -n enableClientDiscOnQueueOverflow -v false
mqsichangeproperties <Broker Name> -o Dynamic Subscription -n maxClientQueueSize -v 0
mqsichangeproperties <Broker Name> -o Dynamic Subscription -n maxBrokerQueueSize -v 0
mqsichangeproperties <Broker Name> -o Dynamic Subscription -n clientPingInterval -v 0
mqsichangeproperties <Broker Name> -o Dynamic Subscription
                                  -n jvmMaxHeapSize -v 536870912
```

Using the JMSAdmin tool that is supplied with the WebSphere MQ JMS implementation, configure the following TopicConnectionFactory objects.

These are the defaults that are used by the Replay Server, and are needed for initial startup, although they can be changed later:

```
DEFINE TCF(replay) BROKERVER(v2) TRANSPORT (direct) HOSTNAME (localhost) PORT(606)
DEFINE TCF(replayAdmin) BROKERVER(v2) TRANSPORT (direct) HOSTNAME (localhost) PORT(606)
```

These objects assume that the underlying WebSphere Message Broker has a JMSIPOptimizedFlow node running on port 606 to provide publish/subscribe.

Configuring Replay Server resources

You must have completed the installation of the relevant instance of the Replay Server before you can configure its resources.

1. Enter the following command at the command prompt of the machine on which you have installed the Replay Server code: .

   `java com.ibm.broker.replay.server.ReplayServer -CREATE server_name` This creates an instance of the Replay Server with the specified server_name. When the default configuration file has been created, it must be manually edited with the correct JNDI parameters.

   Using the values from your JMSAdmin.config file (used by the JMSAdmin tool that is mentioned in the previous section), update the COMMSICF and ICF parameters with the correct InitialContextFactory class, and update the COMMSURL and URL parameters with the correct URL for your JNDI server.
2. Enter the following command at the command prompt of the machine on which you have just created a default Replay Server: `java com.ibm.broker.replay.server.ReplayServer server_name` This starts that server's administration function and provides the Replay Server Console,a command prompt at which you can enter replay administration commands.
3. Enter the command **reportproperties** at the command prompt of the above Replay Server administration window to display the current configuration of the server. Review the displayed property values and decide which, if any, you need to change. See "The changeproperties and reportproperties commands" on page 40 for information about the contents of the configuration file.
4. Enter the following command to modify the default configuration file: `changeproperties -propertyName1 propertyValue1 -propertyName2 propertyValue2 ... -propertyNameX propertyValueX` . This command is described in more detail in "Setting and displaying Replay Server properties" on page 24.

Configuring Replay Server communication

You must know the name of the WebSphere Message Broker to which your Replay Server should connect for message capture and replay.

1. Use the JMS administration tool described in the manual *WebSphere MQ Using Java SC34-6066* to define three named ConnectionFactory objects that describe how the Replay Server collects incoming messages (Collector component) from the WebSphere Message Broker, replays outgoing messages (Replayer component) through the WebSphere Message Broker, and communicates with the Replay Administration Client (Comms component) through the WebSphere Message Broker.

   These connection factories specify the TCP/IP address and port of the target WebSphere Message Broker and the required communication protocol, such as MQ transport or MQ Real-time transport.

   The JMSAdmin command

   ```
   def tcf(REPLAY_CONNECTOR_TCF) brokerver(v2) transport(direct)
       hostname(localhost) port(808)
   ```

   creates a topic connection factory for use by the Collector, using the WebSphere MQ Real-time transport for a broker on the local machine that is running a Real-time flow listening to port 808.
2. Update the Replay Server configuration to use the correct connection factories.

   Use the **changeproperties -collectortcf**, **changeproperties -replayertcf**, and **changeproperties -commstcf** commands to specify the names of the connection factories that you defined in step 1 above. See "The changeproperties and reportproperties commands" on page 40 for more details obout the **changeproperties** command.

Configuring Replay Client communication

The Replay Client API is based upon JMS. Therefore, any underlying transport properties are masked inside ConnectionFactory and TopicConnectionFactory implementations. To allow Replay Client applications to connect to, and communicate with, a Replay Server, the applications must be supplied with an appropriate Replay-based ConnectionFactory. This is looked up from the Java Naming and Directory Interface (JNDI) in the usual JMS way, and is created in the following way.

1. Start the extended REplay JMSAdmin tool with the supplied replayjmsadmin.bat file or replayjmsadmin script.

   This takes an optional parameter that references the config file; the default file is JMSAdmin.config.

   This program wrappers the standard WebSphere MQ JMSAdmin JNDI tool; therefore, the connection details that are held in the JMSAdmin.config file do not need to be changed from the normal use of JMSAdmin.

   JMSAdmin.config can either be copied into the same directory as the Replay installation, or referenced directly from the replayjmsadmin command with an absolute file path.

   The JMSAdmin tool allows access to any commands that the JMSAdmin tool supports, as well as the three commands that are described below.
2. To define a new REplay ConnectionFactory or TopicConnectionFactory, use the following commands:

   ```
   DEFINE RCF(new_name) CF(old_name) REP(replayer)
   ```

or `DEFINE RTCF(new_name) TCF(old_name) REP(replayer)`

where new_name is the lookup key for the new Replay ConnectionFactory or TopicConnectionFactory, old_name is the lookup key for an existing Replay ConnectionFactory or TopicConnectionFactory from which the new object inherits connection properties, and replayer is the name of the Replay Server to connect to, as defined by the ReplayerServer CREATE command.

3. To delete a Replay ConnectionFactory or TopicConnection Factory, use either of the following commands:

   `DELETE RCF(name)`

   or `DELETE RTCF(name)`

   where name is the lookup key for the ConnectionFactory or TopicConnectionFactory that is to be deleted.
4. To display a Replay ConnectionFactory or TopicConnection Factory, use either of the following commands:

   `DISPLAY RCF(name)`

   or `DISPLAY RTCF(name)`

   where name is the lookup key for the ConnectionFactory or TopicConnectionFactory that is to be displayed.

If JMS 1.0.2 is being used, use a TopicConnectionFactory; if JMS 1.1 is being used, use a ConnectionFactory.

In either case, the application needs one of these objects so that it knows how to communicate with the Replay Server.

If a ReplayConnectionFactory needs to be changed, it must be deleted and re-defined. Note also that changing the characteristics of the inherited ConnectionFactory does not affect the Replay ConnectionFactory; it must be deleted and then re-defined.

Administering Replay

The following sections describe how to administer message replay:

- "Starting and stopping Replay Server administration" describes how to start and stop message replay administration.
- "Setting and displaying Replay Server properties" on page 24 describes how to set and display Replay Server properties.
- "Starting and stopping the Replay Server" on page 25 describes how to start and stop a message replay server.
- "Replay security" on page 25 describes how to control message replay administration security.
- "Capturing messages" on page 26 describes how to initiate message capture.
- "Removing messages" on page 27 describes how to initiate background message removal

Starting and stopping Replay Server administration

The Replay Server can be administered in two ways. Use either the Replay Server console or the Replay Admin Client:

Replay Server Console
When a Replay Server is started from a console window, you are provided with a command prompt when startup completes. All server commands can be entered at this command prompt. Entering 'quit' shuts down the Replay Server.

Replay Admin Client
The Replay Admin Client can be run locally, or remotely from the Replay Server. Entering 'quit' shuts down only the REplay Admin Client; the Replay Server remains active.

When you use the Replay Admin Client, the machine on which the client runs requires a Java Runtime Environment, and access to a JNDI namespace.

If the client runs on the same machine as the Replay Server, both are probably available because they are needed for the Replay Server. But, if the client runs remotely, you must make sure that they are available on that machine.

A file admin.cfg that is one of the files that are provided for the installation of the client, provides the JNDI configuration that is used by the REplay Admin Client. This file must be located on the same path that is used to start the Replay Admin Client.

The file contains the three variables that are listed below with their default values:

```
COMMSICF=com.sun.jndi.fscontext.RefFSContextFactory
COMMSURL=file@/C:/jndi
COMMSTCF=replayAdmin
```

COMMSICF provides the initial Context Factory for JNDI lookup.

COMMSURL is the location of the JNDI namespace.

1. Edit this file as required for your system configuration.

2. Enter the following command at the command prompt of the machine at which you will enter administration commands:

   ```
   java com.ibm.broker.replay.server.admin.ReplayAdmin server_name
   ```

   where server_name is the name that was specified on the create command for the Replay Server that you wish to administer (see "Configuring Replay Server resources" on page 19).
3. Enter the command **help** to verify that the administration client is working, and to display help information about the administration commands that are available.
4. Enter the command **ping nn** at the command prompt of the Replay Administration Client to verify that the Replay Admin Client can communicate with the required Replay Server. Here nn is a timeout, in seconds, that the client will wait for a response from the Replay Server.

The newly started administration client offers a command prompt at which you can enter replay administration commands such as **start**, **changeproperties**, **reportproperties**, or **startpruner**.

Setting and displaying Replay Server properties

Make sure that you have started the administration component of the target Replay Server, and, optionally, if administering remotely, that you have started a replay administration client for that target Replay Server, as described in "Starting and stopping Replay Server administration" on page 23, before you can enter any administration commands, including those that set and display Replay Server properties.

The behavior of the Replay Server is driven by a configuration file that persists across the shutdown and startup of the Replay Server. When you first create the Replay Server, you get a configuration file with default values for each of its constituent properties. You can change one or more of these properties using the **changeproperties** command, or you can display the current values of one or more of these properties using the **reportproperties** command.

Note: You can use the **changeproperties** command to modify the default property file before you start the Replay Server replay functions.

If you change the configuration file while the Replay Server is running, you should stop and re-start the Replay Server for the change to take effect. See "The changeproperties and reportproperties commands" on page 40 for full details.

1. Enter the command **changeproperties -propertyName1 propertyValue1 -propertyName2 propertyValue2 ...** at the command prompt in the administration window of the target Replay Server, or at the command prompt of a running replay administration client that is connected to the target Replay Server. This command changes the values of propertyName1 and propertyName2. You can change one or more properties with a single **changeproperties** command. All valid propertyName keywords, together with their default and valid values, are described in "The changeproperties and reportproperties commands" on page 40.
2. Enter the command **reportproperties -propertyName1 -propertyName2 ...** at the command prompt in the administration window of the target Replay Server, or at the command prompt of a running replay administration client that is connected to the target Replay Server. This command displays the current values of propertyName1 and PropertyName2. You can display the

values of one or more properties with a single **reportproperties** command. If you specify no parameters on the **reportproperties** command, the values of all the Replay Server properties are displayed.

Here are some examples that show the use of the **changeproperties** and **reportproperties** commands:

- The command **changeproperties -collectorstring prices/coffee** causes the Replay Server to subscribe to, and capture, messages on the topic prices/coffee.
- The command **changeproperties -storageuserid db2admin28 -collectorstring #** specifies that the Replay Server uses the UserId DB2admin28 to save captured messages on DB2 and captures messages on any topic (the wildcard character # in the root position matches all topics).
- The command **reportproperties** displays the value of every property in the Replay Server configuration file.
- The command **reportproperties -collectorstring** displays the current value of the collectorstring property.
- The command **reportproperties -collectorstring -storageuserid** displays the current values of both the collectorstring and storageuserid properties.

Starting and stopping the Replay Server

You must have already started the administration component of the target Replay Server (and optionally started a replay administration client for that target Replay Server, as described in "Starting and stopping Replay Server administration" on page 23), before you can enter any administration commands.

1. Enter the command **start** at the command prompt in the administration window of the target Replay Server, or at the command prompt of a running replay administration client that is connected to the target Replay Server. This command starts all functions of the replay server.
2. Enter the command **stop** at the command prompt in the administration window of the target Replay Server, or at the command prompt of a running replay administration client that is connected to the target Replay Server. This command stops all functions of the replay server except the pruner and administration. You can restart the replay server by entering a new **start** command.

The Replay Server maintains active connections to both WebSphere Message BrokerWebSphere Event Broker and the database. Therefore, the Replay Server must be stopped before you can stop WebSphere Message BrokerWebSphere Event Broker or the database.

Replay security

This section describes the tasks to set up the appropriate security environment for message capture and replay.

1. Ensure that the Replay Server collector and replayer functions are authorized to access the message storage on DB2:
   a. Set up a DB2 account for the Replay Server collector and replayer functions as described in "Configuring DB2" on page 17.
   b. Use the command **changeproperties -storageuserid db2userid -storagepassword db2password**, as described in "The changeproperties and

reportproperties commands” on page 40, to tell the Replay Server the userid and password for the DB2 account that you set up.

2. Ensure that the Replay Server pruner function is authorized to access the message storage on DB2:
   a. Set up a DB2 account for the Replay Server pruner function as described in “Configuring DB2” on page 17.
   b. Use the command **changeproperties -prunerusername db2userid -prunerpassword db2password**, as described in “The changeproperties and reportproperties commands” on page 40, to tell the Replay Server the userid and password for the DB2 account that you set up.
3. Ensure that the WebSphere Message Broker topic tree authorizes the above userid that was used to start the Replay Server to subscribe to the topics that the Replay Server is capturing. See the WebSphere Message Broker Information Center for more information about topic access security.

Capturing messages

Before you can capture any messages, you must configure your Replay Server, as described in “Configuring Replay” on page 17, and connect it to a broker, as described in “Configuring Replay Server communication” on page 20.

1. Start a replay administration session as described in “Starting and stopping Replay Server administration” on page 23.
2. Enter the command **changeproperties -collectorstring topic_definition**, where topic_definition is a single topic or multiple topics separated by commas. A comma is the default separator; you can change the separator. This instructs your Replay Server to subscribe to each topic within the topic_definition and to capture publications on these topics. Each topic_definition can include wildcard characters (described in “Special characters in topics” on page 38) to capture publications on a range of topics. Note that a subsequent **changeproperties topic_definition** command unsubscribes from all topics in the previous topic_definition and subscribes to all topics in the new topic_definition. The new topic_definition replaces the previous topic_definition; it does not extend it.
3. Ensure that your Replay Server is authorized to subscribe to publications on the above topics.
4. Ensure that your Replay Server is running as described in “Starting and stopping the Replay Server” on page 25.
5. Enter the **reportproperties -collectorstring** command to verify that your Replay Server is capturing the expected publications.
6. If you do not like the comma as a separator in a topic_definition, enter the command **changeproperties -topicseparator separator** to change it. Here, separator is a single character that specifies the required separator.

Note: If the Replay Server is active (the **start** command has been issued at the Replay Server console, or at the Replay Admin Client), changes to the **collectorstring** variable have immediate affect; you do not have to restart the Replay Server.

Displaying the active topics

Enter the **reportproperties -collectorstring** command to display the currently active topic definition for your Replay Server.

Removing messages

Before you can initiate asynchronous message removal, you must first configure your Replay Server as described in “Configuring Replay” on page 17.

1. Review the information in “The pruner commands” on page 44 to determine the strategy that you require for asynchronous message removal. This identifies the criteria for starting message removal (the -highWaterMark keyword) and the criteria for how many messages should be removed (the -pruningMethod keyword).
2. Decide the timing of the test to determine whether asynchronous message removal should be started or not. You can request that this test be done immediately (the -now keyword), or repeatedly at a specified interval (the -delay keyword), or repeatedly at a specified time of day (the -timeOfDay keyword).
3. Decide whether you only want messages with specified topics to be removed (the -topicNames keyword), or whether you want messages with any topic to be removed. If you specify the -topicNames keyword, the pruning operation is applied separately to each specified topic. If you omit the -topicNames keyword, the pruning operation ignores the topic of the candidate messages for message removal.
4. Start a replay administration session as described in “Starting and stopping Replay Server administration” on page 23.
5. Enter an appropriate **startpruner** command to achieve the desired result. Review the examples shown in “The pruner commands” on page 44 to help you choose the required command.
6. Enter the **stoppruner** command to finish removing messages.

Developing applications

The following sections describe how to write message replay applications:

- "Understanding the Java Message Service (JMS)" describes how a subscriber client application can already use the Java Message Service (JMS) to subscribe to a specified set of topics and receive messages about these topics.
- "Understanding replay" on page 31 describes how a subscriber client application can use proprietary JMS extensions to receive messages that have been captured by the Replay Server.
- "Selecting the required messages" on page 32 describes what a subscriber client application must do to specify the messages that it wants to receive from a Replay Server.

Understanding the Java Message Service (JMS)

The Java Message Service support in JMS 1.0.2 is based around the following interfaces:

Connection
This provides access to the underlying transport, and is used to create Sessions. It holds the parameters that control how to connect to WebSphere Message Broker.

Session
This provides a context, such as transaction state, for message producers (publishers) and message consumers (subscribers).

Message producer
This is used to send messages.

Message consumer
This is used to receive messages.

A key idea in JMS is that application programs should be written using only standard JMS interfaces. All vendor-specific information is encapsulated in the following offline defined objects:

- QueueConnectionFactory
- TopicConnectionFactory
- Queue
- Topic

These are known as ″administered objects″; that is, they are objects that can be built using a vendor-supplied administration tool and can be stored in a JNDI (Java Naming and Directory Interface) namespace. A JMS application can retrieve these objects from the namespace and use them without needing to know which vendor provided the implementation.

The generic JMS interfaces are divided into subclasses specifically for ″Point-to-Point″ and ″Publish/Subscribe″ behavior.

The publish/subscribe versions are:

TopicConnection
Use the CreateTopicConnection method to create a TopicConnection using an offline-defined TopicConnectionFactory that you access from the JNDI namespace. The TopicConnectionFactory contains WebSphere MQ specific attributes. These specify whether the connection uses MQ transport to a queue manager with a specified queue manager name and host name from which a WebSphere Message Broker reads its input messages, or the connection uses MQ real time transport directly to a WebSphere Message Broker with a specified host name and port number.

TopicSession
Use the CreateSession method of the TopicConnection object to create a TopicSession. This takes parameters that specify whether the TopicSession is transacted or non-transacted, and also the TopicSession's acknowledge mode.

Topic
JMS does not tightly define how a topic name is structured. The WebSphere Message Broker uses a hierarchical name like "Sports/Football/Spurs/Signings". You must publish to a specific topic, but you can subscribe to multiple topics through a topic name that includes wild cards as described in "Special characters in topics" on page 38. You can define topics offline and access them via the JNDI namespace, or you can use the CreateTopic or CreateTemporaryTopic methods of the TopicSession object to dynamically create a topic.

TopicPublisher
Use the CreatePublisher method of the TopicSession to create a TopicPublisher for a specified topic. The TopicPublisher object includes the Publish method to publish a message on the specified topic.

TopicSubscriber
Use the CreateSubscriber method of the TopicSession object to create a TopicSubscriber to a specified topic. The TopicSubscriber object includes the Receive and ReceiveNoWait methods to receive a message published to the specified topic. The subscriber can only receive messages that are published while that subscriber is active. When the TopicSubscriber object is closed, the corresponding subscription is deleted.

The CreateSubscriber method can also accept a second parameter (the first parameter being the topic name) to specify a message selector that states the required values of a specified set of message properties. The methods also accept a third parameter that specifies whether messages that are published on the same connection as the subscriber are to be suppressed, but this parameter is irrelevant for replay.

A JMS Message includes a standard message header that contains standard message properties, and an additional message header that contains extra application-defined message properties. A JMS client application can reference these message properties in a message selector that it can pass to the JMS provider to filter messages that are received. Message selectors cannot reference values in the message body. A message selector uses a subset of the Standard Query Language (SQL) to specify the filter expression.

JMS also supports asynchronous message delivery. It implements the MessageListener interface, which contains one method, onMessage. In the onMessage method, you define the actions that are taken when a message arrives. You register the message listener with a specific TopicSubscriber by using the setMessageListener method. After you register the message listener, you call the startmethod on the TopicConnection to begin message delivery. When message delivery begins, JMS automatically calls the message listener's onMessage method whenever a message arrives that matches the topic and filter expression in the TopicSubscriber. The user-written onMessage method takes one argument of type Message, which the method should cast to the message type of the received message.

The JMS 1.1 specification continues to support JMS 1.0.2 compliant applications. It also supports and recommends a simpler and more generic programming interface that is common to both ″point to point″ messaging and publish/subscribe messaging. The following table shows how the JMS 1.0.2 messaging style dependent interfaces map to the JMS 1.1 common interfaces. Replay supports both.

Publish/subscribe interfaces in JMS 1.0.2 and JMS 1.1	Common interfaces in JMS 1.1
TopicConnectionFactory	ConnectionFactory
TopicConnection	Connection
TopicSession	Session
Topic	Destination
TopicPublisher	MessageProducer
TopicSubscriber	MessageConsumer

Understanding replay

You can modify an existing JMS subscriber application or write a new JMS subscriber application to exploit message replay by using the proprietary JMS extensions that are described in "Java programming interfaces" on page 51.

The addition of a Replay Server to your JMS publish/subscribe network has no affect at all on the interfaces for publishing messages. An unchanged JMS application can publish messages using standard JMS publish/subscribe interfaces and can have these messages captured by your Replay Server for possible later replay. The publishing application is completely unaware that some or all of its published messages have been captured by your Replay Server.

An existing unchanged JMS publish/subscribe application cannot subscribe to or receive messages from a Replay Server. You must replace some of your JMS methods with equivalent proprietary JMS replay extensions that are described in this section.

The following table shows how the JMS replay extensions relate to the normal JMS publish/subscribe interfaces, for both JMS 1.0.2 users and JMS 1.1 users:

JMS 1.0.2 publish/subscribe	JMS 1.0.2 style replay	JMS 1.1 publish/subscribe	JMS 1.1 style replay
-	ReplaySignature	-	ReplaySignature
TopicConnectionFactory	TopicConnectionFactory	ConnectionFactory	ConnectionFactory
TopicConnection	TopicConnection	Connection	Connection
Topic	Topic	Destination	Destination
TopicSession	ReplayTopicSession	Session	ReplaySession
TopicSubscriber	ReplayTopicSubscriber	MessageConsumer	ReplayMessageConsumer

-	ReplayJMSMessage	-	ReplayJMSMessage

You code a JMS message replay subscriber application exactly as you would code a standard JMS subscriber application, except for the following:
- ReplayJMSManager.
- ReplaySignature.
- ReplaySession..
- ReplayMessageConsumer.
- ReplayTopicSession.
- ReplayTopicSubscriber.

Refer to “Java programming interfaces” on page 51 for more details.

These JMS extensions are in the `com.ibm.broker.replay.api` package that is part of the Replay jar file.

Selecting the required messages

The ReplaySignature object specifies the required start and end points for your replay, together with information on the required time interval between replay of successive messages, and information on whether you want to replay every message or a sample of messages.

You then pass a TopicString (which might include a wildcard character), a message selector expression, and a ReplaySignature object as parameters on a createReplayConsumer or createReplayTopicSubscriber method to create a new ReplayMessageConsumer object (or on a reset method to reset an already active ReplayMessageConsumer object).

1. You can instantiate a ReplaySignature object at any time by using the newReplaySignature method. You can then use the following methods to define or discover the properties of the ReplaySignature.
   a. Decide the required replay start point.

      You have the choice of setStartNow, setStartAsTimestamp, and setStartAsSequenceNumber.

      The setStartNow does not imply a timestamp, but is rather a request to connect the ReplayMessageConsumer to the near real time feed as soon as possible, provided that the time interval is not throttled.

      The setAsSequenceNumber specifies an absolute message sequence number for the target topic, rather than a relative sequence number. You must thus have previously extracted a valid sequence number from a previously retrieved message using the getReplaySequenceNumber method of the ReplayJMSMessage object.

      You can determine the current value of the replay start point using methods like isStartNow and isStartSequenceNumber to determine the type of start point and then the method getStart to retrieve the actual start value.
   b. Decide the required replay end point.

      You have the choice of setEndNever, setEndAsTimestamp, and setEndAsSequenceNumber.

    The setEndNever method connects the replay session to the near real time feed as soon as possible after all relevant captured messages have been replayed, provided that the time interval is not throttled.

    You can query the specified end point by methods like isEndNever, isEndSequenceNumber, and getEnd.

   c. Decide the required time interval between the replay of successive messages.

      You have the choice between setTimingToAsap which replays successive messages as quickly as possible, setTimingToOriginal which replays successive messages with the same time intervals as the messages were originally published, and setTimingToThrottled which replays successive messages with a specified minimum time interval.

      You can use methods like isTimingOriginal to discover the current replay timing style, and the method getThrottlingInterMessageTime if you have previously specified setTimingByThrottle.

   d. Decide whether you want to replay all messages or a subset of messages.

      You have a choice between setSamplingToByNumberOfMessages which replays every Nth message (where N can be 1) and setSamplingToByTime which replays messages separated by a specified time interval ignoring intervening messages.

      You can use methods like isSampled and isSampledByNumberOfMessages to determine the current sampling style, and the method getSamplingInterval to discover the current sampling interval.

   e. Decide how you want any replay sampling to relate to message topics.

      This is only relevant if you enable replay sampling and you specify a wild card in the topic string for your replay request.

      You have a choice between setSamplingOnAggregatedTopics which applies the sampling rules to the totality of messages ignoring topics, and setSampligByTopic which separately applies the sampling rules to each distinct topic.

      You can use methods like isSampledOnTopic to discover the current sampling style.

2. Use the createReplayConsumer or createReplayTopicSubscriber method to create a new ReplayMessageConsumer object passing the following parameters:
   a. A Destination that specifies the topic, or topics, of the messages that you want to be replayed. This Destination can include a wildcard character as described in “Special characters in topics” on page 38, and can thereby encompass multiple topics.
   b. An optional message selector. This is a character string that contains an SQL like expression that specifies acceptable values of one or more fields in the message header. A null string indicates that all messages are selected.
   c. A ReplaySignature. If this is omitted, a default ReplaySignature of setStartNow, setEndNever, setTimingASAP, and setSamplingToByNumberOfMessages to 1 (i.e. replay every message) is assumed.
3. If you want to change the message selection criteria for an already active ReplayMessageConsumer then use the reset method passing the same parameters (Destination, MessageSelector, ReplaySignature) as for the createReplayConsumer method.

The following code snippet creates a ReplaySignature to replay every 20th captured message as quickly as possible starting 100 seconds ago and continuing indefinitely. It then uses that signature to create a ReplayMessageConsumer to replay these messages across all captured topics.

```
ReplaySignature sig = new ReplaySignature();
sig.setStartAsTimestamp(System.currentTimeMillis() - 100000);
sig.setEndToNever();
sig.setTimingToAsap();
sig.setSamplingOnAggregatedTopics();
sig.setSamplingToAsNumberOfMessages(20);

conn = cf.createConnection();
sesh = conn.createSession(false, Session.AUTO_ACKNOWLEDGE);
cons = ((ReplaySession)sesh).createReplayConsumer("#", null, sig);
```

Problem diagnosis

The following section provides information about how to use diagnostic trace.

Diagnostic trace

Diagnostic trace files are written in the root directory of the Replay Server.

You control whether anything is written to a diagnostic trace file by setting the **tracelevel** property of the **changeproperties** command.

- Set the value of the **tracelevel** property to `none` if you do not want any trace output.
- Set the value of the **tracelevel** property to `event` if you want a record to be written to the diagnostic trace file whenever a system event (for example, the pruner starting or stopping) occurs on the Replay Server.
- Set the value of the **tracelevel** property to `summary` if you want a record to be written to the diagnostic trace file whenever, for example, subscriptions occur on the Replay Server.
- Set the value of the **tracelevel** property to `method` if you want a record to be written to the diagnostic trace file whenever a method is entered or exited from on the Replay Server.
- Set the value of the **tracelevel** property to `detail` if you want a record to be written to the diagnostic trace file whenever, for example, publications occur on the Replay Server.
- Set the value of the **tracelevel** property to `full` if you want a record to be written to the diagnostic trace file whenever, for example, a subscription is received from, or a message is sent to, the Replay Server.

These levels are cumulative; for example, if you specify `method`, you also get `summary` and `event` trace records.

You can control where the diagnostic trace records are written by setting the **tracedestination** property of the **changeproperties** command.

- Set the value of the **tracedestination** property to `combined` if you want all trace records to be written to a single file.
- Set the value of the **tracedestination** property to `component` if you want trace records for separate components of the Replay Server to be written to separate files.

Note: You do not have to stop the Replay Server when you change your trace options.

The names of the trace files that are produced when tracing the Replay Server are dependent upon the setting of the **tracedestination** property.

If the **tracedestination** property is set to `combined`, one trace file is produced with a name similar to replayServerName_replay.log, where replayServerName is the name of the Replay Server that produced the file.

Server FFDC

FFDC = First Failure Data Capture

Any exceptions that are thrown by the Replay Server are caught and added to a file called `ffdc_replay.log`. This file is located in the root directory of the Replay Server.

Publication tracking

You can trace a publication on its path through the Replay Server. The following code fragment shows how to set this up:

```
TextMessage msg:
//.........
boolean isTracking;
// Set isTracking to true or false
msg.setBooleanProperty("com.ibm.broker.replay.track_publication", isTracking);
...........
```

The **tracelevel** property must be **event**, or above, for Publication Tracking entries to appear in the trace file.

Client Tracing

To use tracing on the Replay Client, the application must be started with property -DTRACE=ON in the Java command: `java -DTRACE=ON myReplaySubscriber`

This creates a file with a name similar to REPLAY_1103117519538_696.TRC.

This trace can be used in addition to the standard WebSphere MQ JMS trace.

Error publication

The Replay Server also publishes error and warning messages on the topic $SYS/REPLAY/<replay server name>/ERRORS.

These messages can be used to monitor the status of the Replay Server.

Command reference

This section describes the commands that are provided by the Replay Server:

- Use the “The changeproperties and reportproperties commands” on page 40 to connect your Replay Server to the required WebSphere Message Broker, and to specify the topics that are to be captured by the Replay Server.
- Use the “The pruner commands” on page 44 to initiate immediate or periodic removal of messages from the persistent store.

This section also provides the following reference information that is applicable to all Replay Server administration commands:

- “Characters allowed in commands” for information about the characters that you can use in your commands.
- “Special characters in topics” on page 38 for information about how to structure topic strings and include wildcard characters so that a single subscriber topic string can match a range of publications.
- “Responses to commands” on page 40 for information about command response messages.

Characters allowed in commands

There are a few rules that you must adhere to when you provide names or identifiers for the objects that you reference in your Replay Server administration commands.

The Replay Server resources are:

- Brokers
- Topics

The character set that you can use to name brokers is:

- Uppercase A-Z
- Lowercase a-z
- Numerics 0-9
- Any special characters supported by the underlying file system:

Character	Windows	UNIX	Character	Windows	UNIX
$	Yes	No	%	Yes	Yes
′ (apostrophe)	Yes	No	′ (quote)	Yes	No
- (dash)	Yes	Yes	_(underscore)	Yes	Yes
@	Yes	Yes	~ (tilde)	Yes	Yes
!	Yes	Yes	&	Yes	Yes
(	Yes	No	)	Yes	No
{	Yes	Yes	}	Yes	Yes
[	Yes	Yes	]	Yes	Yes
#	Yes	Yes	+	Yes	No
, (comma)	Yes	Yes	;	Yes	No

Character	Windows	UNIX	Character	Windows	UNIX
=	Yes	Yes	(space)	Yes	Yes

In general, you can use characters **A** through **Z**, **a** through **z**, and **0** through **9**, plus any Unicode character with a decimal value greater than 127 (hexadecimal X'7F'), provided that your operating system can recognize the characters chosen.

For all other resources, any characters that are supported by the database configuration are supported.

Broker names and fixed names (`ConfigMgr` and `UserNameServer`) are not case-sensitive on the Windows platforms. For example, broker names `Broker1` and `BROKER1` refer to the same broker.

On UNIX platforms, broker names are case-sensitive and `Broker1` and `BROKER1` refer to different brokers. You must use `UserNameServer` as shown. (In this context, `ConfigMgr` is not relevant.)

Special characters in topics

A topic can contain any character in the Unicode character set. However, the following three characters have a special meaning:
The topic level separator "/".
The multilevel wildcard "#".
The single-level wildcard "+".

Restriction: You must avoid '?' in topic names that are used by the Replay Server.

The topic level separator is used to introduce structure into the topic, and can therefore be specified within the topic for that purpose.

The multilevel wildcard and single-level wildcard can be used for subscriptions, but they cannot be used within a topic by the publisher of a message.

The topic level separator

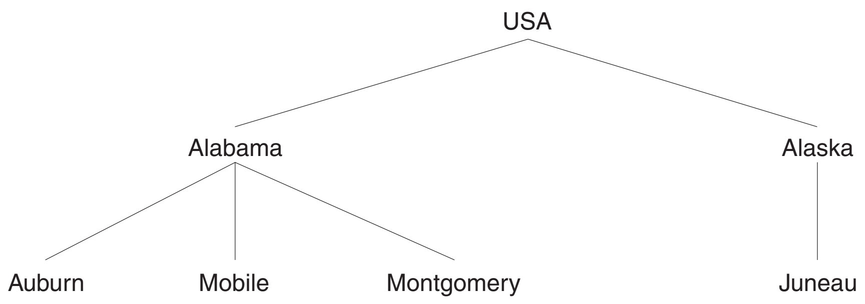
The topic level separator character "/" is used to provide a hierarchical structure to the topic space. It must be used by applications to separate levels within a topic tree. The use of the topic level separator is significant when the two wildcard characters are encountered in topics specified by subscribers.

Topic hierarchy is important in the administration of access control.

The multilevel wildcard

The multilevel wildcard character "#" is used to match any number of levels within a topic.

The following figure shows an example of a topic tree with one root topic:



For example, using the example topic tree shown above, if you subscribe to ″USA/Alaska/#″, you receive messages on topics ″USA/Alaska″ and ″USA/Alaska/Juneau″.

The multilevel wildcard can represent zero or more levels. Therefore, ″USA/#″ can also match the singular ″USA″, where # represents zero levels. The topic level separator is meaningless in this context, because there are no levels to separate.

The multilevel wildcard can be specified only on its own or next to the topic level separator character. Therefore, ″#″ and ″USA/#″ are both valid, but ″USA#″ is not valid.

The single-level wildcard

The single-level wildcard character ″+″ matches one, and only one, topic level. For example, ″USA/+″ matches ″USA/Alabama″, but not ″USA/Alabama/Auburn″. Also, because the single-level wildcard matches only a single level, ″USA/+″ does not match ″USA″.

The single-level wildcard can be used at any level in the topic tree, and in conjunction with the multilevel wildcard. The single-level wildcard must be specified next to the topic level separator, except when it is specified on its own. Therefore, ″+″ and ″USA/+″ are both valid, but ″USA+″ is not valid

Rules for using commands

Observe the following rules when using the Replay Server commands:

- Each command must be issued at the command prompt of the administration window of the Replay Server whose behaviour you want to modify, or at the command prompt of an administration client directed to the Replay Server whose behaviour you want to change.
- Each command starts with a primary keyword (the executable command name) followed by one or more blanks.
- Following the primary keyword, parameters can occur in any order.
- Parameters are shown in the form `-t`, for example. In all cases, the character `/` can be substituted for the `-` character.
- If a parameter has a corresponding value, its value must follow the parameter to which it relates. A parameter can be followed directly by its value, or it can be separated by any number of blanks.
- Repetition of parameters is not allowed.
- Strings that contain blanks or special characters must be enclosed in double quotation marks. For example:

```
changeproperties -collectorstring "hall of fame"
```

You can specify a null, or empty, string with a pair of double quotes with nothing between.

- The case sensitivity of primary keywords and parameters depends on the underlying operating system. On Windows platforms, keywords are not case sensitive; for example: `changeproperties`, `CHANGEPROPERTIES` and `ChangeProperties` are all acceptable. On UNIX platforms, you must use lower case; only `changeproperties` is acceptable.

Responses to commands

Responses are issued to the commands as messages that are returned to the console from which they were issued.

The changeproperties and reportproperties commands

Supported operating systems

- Windows 2000, Windows XP
- UNIX

Purpose

Use the **changeproperties** command to change one or more properties of the Replay Server.

Use the **reportproperties** command to display the current value of one or more or all of the properties of the Replay Server

Syntax

```
►►──changeproperties──┬─-propertyName1──propertyValue1──┬──────────────────►◄
                      └──-propertyName2──propertyValue2─┘
```

```
►►──reportproperties──┬─-all──────────┬────────────────────────────────────►◄
                      ├─-propertyName1─┤
                      └──-propertyName2─┘
```

Parameters

The following is a list of the available property names and their associated values:

-replayServerName characterString
Specifies the name of the Replay Server , as provided on the java

com.ibm.broker.replay.server.ReplayServer -CREATE server_name command. You cannot change this name. Instead you should create a new Replay Server with the requiired name.

-collectorString topicList
Sets the character string identifying the topic(s) of the messages which the Replay Servercaptures. This topicList can contain a single topicString or multiple topicStrings separated by a specified topicSeparator. Each topicString can contain a wild card character to match multiple topics as described in "Special characters in topics" on page 38. A subsequent changeproperties -collectorString command will replace the current topicList rather than add to it. Changes to -collectorString take immediate effect. The default -collectorString is the wild card character ″#″. This matches all topics which means that the Replay Server captures messages on any topic.

-collectortcf connectionFactoryName
Specifies the name of a JMS ConnectionFactory. This in turn specifies the WebSphere Business Integration Message Broker that the Replay Server collector function subscribes to, and which transport protocol , such as MQ real time transport or MQ transport, is used to receive matching publications. You must stop and restart the replay server for this change to take effect. The default value is *replay*.

-commstcf connectionFactoryName
Specifies the name of a JMS ConnectionFactory. This in turn specifies the WebSphere Business Integration Message Broker to which the Replay Server administration function connects, and the transport protocol , such as MQ real time transport or MQ transport, used to carry administration requests. You must close and restart the replay server for this change to take effect. The default value is *replayAdmin*.

-commsicf contextFactoryName
Specifies the initial context factory which the replay server administration function uses to look up named objects, such as a named ConnectionFactory, using the Java Naming and Directory Interface (JNDI). The possible values depend on the -connsurl parameter, because -commsicf and -commsurl together describe how to access the JNDI. You must close and restart the replay server for this change to take effect. The default value is *com.sun.jndi.fscontext.RefFSContextFactory*. ??? We need a better explanation here with details of the semantics of the allowed combinations of the -connsicf and -connsurl keywords ???

-commsurl urlValue
Specifies the location of the Java Naming and Directory service which the replay service administration function uses to look up named JMS administered objects, such as a named ConnectionFactory. You must close and restart the replay server for changes to this parameter to take effect. The default value is *file:/C:/jndi*.

-icf contextFactoryName
Specifies the initial context factory which the replay server uses to look up named objects, such as a named ConnectionFactory, using the Java Naming and Directory Interface (JNDI). The possible values depend on the -url parameter, because -icf and -url together describe how to access the JNDI. You must close and restart the replay server for this change to take effect. The default value is *com.sun.jndi.fscontext.RefFSContextFactory*.

-replayertcf connectionFactoryName
Specifies the name of a JMS ConnectionFactory. This in turn specifies which

WebSphere Business Integration Message Broker the Replay Server replayer function connects to, and which transport protocol , such as MQ real time transport or MQ transport, is used to publish replayed messages. You must stop and restart the Replay Server for this change to take effect. The default value is *replay*.

-storagepassword password
Specifies the password which the Replay Server uses to store messages on DB2. This should match the password associated with the Replay Server storage component DB2 account name which you established as described in "Configuring DB2" on page 17. You are unlikely to change this value, but if you do then you must close and restart the Replay Server for the change to take effect. There is no default value.

-storageuserid userIdentifier
Specifies the userid which the Replay Server uses to store messages on DB2. This should match the userid associated with the account name used by the Replay Server storage component as described in "Configuring DB2" on page 17. You are unlikely to change this value, but if you do, you must close and restart the Replay Server for the change to take effect. There is no default value.

-storage_url url
Specifies the URL of the machine that hosts the replay database. A URL is required only if the replay database is on a different machine from the pruner. If the parameter is blank, or is omitted, it is assumed that the replay database is hosted on the same machine as the pruner.

-storage_portnumber portnumber
Specifies the port number that the Replay Server listens on for incoming connections. The port number is needed only when connecting to a remote database. If left blank, the default value of portnumber is 50,000.

-topicSeparator separatorCharacter
Specifies the single character topic separator which separates multiple topicStrings in a single topicList. The default separatorCharacter is ",". Any change takes immediate effect, but you must change collectorString at the same time.

-url urlValue
Specifies the location of the Java Naming and Directory service which the replay server uses to look up named objects, such as a named ConnectionFactory. You must stop and restart the Replay Server for any change to take effect. The default value is `file:/C:/jndi`.

-tracelevel level
Specifies the amount of trace information that is written by the Replay Server. Choose one of the following values: `none`, `event`, `summary`, `method`, `detail`, or `full`. See "Diagnostic trace" on page 35 for an explanation of these values. The default value is `event`.

-tracedestination destination
Specifies the form that the written trace files take. Choose one of the following values: `combined` or `component`. See "Diagnostic trace" on page 35 for an explanation of these values. The default value is `combined`.

-managermaxoverflow overflow
Specifies the maximum number of messages that can be held in memory pending being written to non-volatile storage by the Replay Server. If the in-memory message store grows to more than this value, all messages that are

held in the in-memory message store are discarded. Setting this value too high can cause the Replay Server to run out of resources. The default value is 25000.

-managermaxoverflowcycles cycles
Specifies the maximum number of consecutive ″overflowing″ storage cycles that the Replay Server undertakes before all messages in the in-memory message store are deleted. An ″overflowing″ storage cycle occurs when the number of messages in the in-memory store greater than the value specified in **storagemaxrecords**, and has increased since the last storage cycle. The default value is 2.

-storagethreads number
Specifies the number of threads that the Storage component of the Replay Server uses to concurrently write messages to non-volatile storage. The default value is 10.

-storagemaxrecords number
Specifies the maximum number of messages that one thread of the Storage component of the Replay Server attempts to write to non-volatile storage at any one time. The default value is 1000.

-storagebatchsize number
Specifies the size of the batch that is used to store messages by the threads of the Storage component of the Replay Server. For example, if **storagemaxrecords** is 1000 and **storagebatchsize** is 100, when a thread writes 1000 messages, it presents them to non-volatile storage in 10 batches of 100 messages. The default value is 100.

-storagebuffersize number
Specifies the maximum size of the messages that can be written to non-volatile storage. This number must match the value of the BLOB size in the DDL file that is used to create the Replay database; for more information, see "Configuring DB2" on page 17. The default value is 32768.

-replaynlpoolsize number
Specifies the number of threads that are used by the Replayer component of the Replay Server to publish messages on the near-live stream. The default value is 10.

-nearlive value
Specifies whether the near-live stream is to be used. Choose a value of either `on` or `off`. The default value is `on`.

Authorization

On Windows platforms, the user ID that is used to invoke this command must have **Administrator** authority on the local system.

On UNIX platforms, the user ID that is used to invoke this command must either be **root** or the same as that specified in the `-i` parameter of the **mqsicreatebroker** command. It must also be a member of the **mqbrkrs** group.

Examples

Here are some example commands:

- The following command captures messages for the topic prices/copy:

```
changeproperties -collectorstring  prices/coffee
```

- The following command captures messages for the topics prices/copy and sales/coffee:

  ```
  changeproperties -collectorstring prices/coffee,sales/coffee
  ```
- The following command captures messages for topic prices for any commodity and messages for topic sales/tea:

  ```
  changeproperties -collectorstring  prices/#,sales/tea
  ```

The pruner commands

Supported operating systems

- Windows 2000, Windows XP
- UNIX

Purpose

Use the **startpruner** command to start the pruner. Asynchronous message removal from the persistent store is necessary to prevent the persistent store becoming full.

Use the **stoppruner** command to stop the pruner.

Use the **prunerstatus** command to display statistics about the pruner.

Syntax

```
►►──startpruner──-strategyfile──file──-strategy──strategy──────────────────────►

►──pruner_userid──username──-pruner_password──password──────────────────────────►

►─pruningvariable──variable──-pruningmethod──method──-delay──delay──────────────►

►─highwatermark──highwatermark──topicstoprune──topics──timestoprune──times──────►

►─prunetimestart──timestamp──prunetimeend──timestamp──storage_url──url──────────►

►─storage_portnumber──portnumber───────────────────────────────────────────────►◄
```

```
►►──stoppruner─────────────────────────────────────────────────────────────────►◄
```

```
►►──prunerstatus───────────────────────────────────────────────────────────────►◄
```

Parameters

-strategyfile file
: Specifies the name of a file that contains the parameters that are to be used by the pruner.

-strategy strategy
: Specifies the pruning strategy to use. Possible options are:

'StandardPrune' (or 'sp')

'StandardPruneOnTopics' (or 'spot')

'immediate' (or 'i')

'immediateOnTopics' (or 'iot')

'pruneAtTimes' (or 'pat')

'pruneAtTimesOnTopics' (or 'patot')

'pruneBetweenTimes' (or 'pbt')

'pruneBetweenTimesOnTopics' (or 'pbtot')

-pruner_userid username
Specifies the username for connecting to the database.

-prunerpassword password
Specifies the password of the username for connecting to the database.

-pruningvariable variable
Specifies the pruning variable to use.

-pruningmethod method
Specifies the pruning method to use. Possible options are:

'Oldest' (or 'op' or 0)
Removes the oldest messages until it has removed the number of messages that is specified in **pruningvariable**.

'Percentage' (or 'p' or 1)
Removes the oldest messages until it has removed the percentage (%) that is specified in **pruningvariable**.

'TimeBased' (or 't' or 2)
Removes all messages that are older than the time that is specified in **pruningvariable**.

The following formats of **pruningvariable** are valid:

ss - removes all messages that are more than ss seconds old.

mm:ss - removes all messages that are more than mm minutes and ss seconds old.

hh:mm:ss - removes all messages that are more than hh hours, mm minutes, and ss seconds old.

dd:hh:mm:ss - removes all messages that are more than dd days, hh hours, mm minutes, and ss seconds old.

'Watermark' (or 'w' or 3)
Removes the oldest messages until only the number of messages that is specified in **pruningvariable** remain.

For example, if **highwatermark** is 100,000 and **pruningvariable** is 75,000, the oldest 25,000 messages are removed to bring the message count down to 75,000.

-delay delay
Specifies the time interval between database checks. Possible formats are:

ss - time in seconds

mm:ss - time in minutes and seconds

hh:mm:ss - time in hours, minutes, and seconds

dd:hh:mm:ss - time in days, hours, minutes, and seconds

-highwatermark highwatermark
Specifies the minimum number of messages that must be in the database before pruning is started.

topicstoprune topics
If using a topic-based pruning strategy, this specifies a comma-separated list of the topics to prune.

Note: You can specify wildcard topics, but note that the wildcard character ('*') that you use in pruner commands is different from the wildcard characters ('#' and '+') that you can use in subscriptions. For example, if you specify 'computer/sales/*', topics 'computers/sales/desktops', 'computers/sales/laptops', 'computers/sales/PDA', and 'computers/sales/servers' are all pruned.

timestoprune topics
If using a time-based pruning strategy, this specifies the time to start pruning. The format for this time can be:

hh:mm - for example, 14:00 specifies 2pm.

ddd-hh:mm - for example, Mon-14:00 specifies Monday at 2pm.

d-hh:mm - for example, 6-14:00 specifies the 6th day of the month at 2pm.

-prunetimestart timestamp
Specifies the timestamp of the first message to be removed.

-prunetimeend timstamp
Specifies the timestamp of the last message to be removed.

-storageurl url
Specifies the URL of the machine that hosts the replay database. A URL is only required if the replay database is on a different machine from the pruner. If this parameter is blank or omitted, it is assumed that the replay database is hosted on the same machine as the pruner.

-storage_portnumber portnumber
Specifies the port number that the replay database is listening on for incoming connections. The port number is required only when connecting to a remote replay database. If this parameter is left blank, the default value of the port number is 50,000.

Authorization

On Windows platforms, the user ID used to invoke this command must have **Administrator** authority on the local system.

On UNIX platforms, the user ID used to invoke this command must either be **root** or must be the same as that specified in the `-i` parameter. It must also be a member of the **mqbrkrs** group.

Examples

Here are some example commands:

Pruning all topics

```
startpruner -delay 1000 -highwatermark 10000 -pruningmethod Oldest
-pruningvariable 2500
```

The pruner checks the database once every 1000 seconds. If the number of messages in the database is greater than 10,000, the oldest 2500 messages are removed.

```
startpruner -delay 5:00 -highwatermark 20000 -pruningmethod p
-pruningvariable 20
```

The pruner checks the database once every 5 minutes. If the number of messages in the database is greater than 20,000, the oldest 20% of the messages is removed.

```
startpruner -delay 1:00:00 -highwatermark 30000 -pruningmethod 2
-pruningvariable 30:00
```

The pruner checks the database once every hour. If the number of messages in the database is greater than 30,000, messages that are older than 30 minutes are removed.

```
startpruner -delay 3:00:00:00 -highwatermark 40000 -pruningmethod 3
-pruningvariable 25000
```

The pruner checks the database once every 3 days. If the number of messages in the database is greater than 40,000, the oldest messages are removed, until only 25,000 remain.

Immediate pruning

```
startpruner -strategy immediate -highwatermark 20000 -pruningmethod p
-pruningvariable 20
```

The pruner checks the database immediately. If the number of messages in the database is greater than 20,000, the oldest 20% of the messages is removed.

Pruning at specific times

```
startpruner -strategy pruneAtTimes -timestoprune 12:00,18:00,00:00
-highwatermark 10000 -pruningmethod TimeBased -pruningvariable 2:00:00:00
```

The pruner checks the database at noon, 6 p.m., and midnight every day. If the number of messages in the database is greater than 10,000, all messages that are more than 2 days old are removed.

```
startpruner -strategy pruneAtTimes -timestoprune Mon-06:00,Wed-18:00,Fri-21:00
-highwatermark 100000 -pruningmethod Oldest -pruningvariable 10000
```

The pruner checks the database at 6 a.m. every Monday, 6 p.m. every Wednesday, and 9 p.m. every Friday. If the number of messages in the database is greater than 100,000, the oldest 10,000 messages are removed.

```
startpruner -strategy pruneAtTimes -timestoprune 1-00:00,15-18:00
-highwatermark 500000 -pruningmethod p -pruningvariable 50
```

The pruner checks the database at midnight on the first day of every month, and at 6 p.m. on the 15th day of every month. If the number of messages in the database is greater than 500,000, the oldest 50% of the messages is removed.

Pruning selected topics

```
startpruner -strategy StandardPruneOnTopics -topicstoprune
prices/FTSE/IBM,prices/DOW/AnotherCompany -delay 30:00 -highwatermark 1000
-pruningmethod Oldest -pruningvariable 250
```

The pruner checks the database once every 30 minutes. If the number of messages on the topics 'prices/FTSE/IBM' or 'prices/DOW/AnotherCompany' is greater than 1000, the oldest 250 messages on those topics are removed.

```
startpruner -strategy StandardPruneOnTopics -topicstoprune
prices/FTSE/MyCompany -delay 30:00 -highwatermark 10000
-pruningmethod Percentage -pruningvariable 25
```

The pruner checks the database once every 30 minutes. If the number of messages on the topic ’prices/FTSE/MyCompany’ is greater than 10,000, the oldest 25% of messages is removed.

```
startpruner -strategy StandardPruneOnTopics -topicstoprune
computer/sales/*,prices/shares/PruneJuice -delay 30:00 -highwatermark 10000
-pruningmethod Percentage -pruningvariable 25
```

The pruner checks the database once every 30 minutes. If the number of messages on the topic ’prices/shares/PruneJuice’, or the cumulative number of messages on all topics described by ’computer/sales/*’ (for example, ’computer/sales/desktops’, ’computer/sales/laptops’, ’computer/sales/PDA’, and ’computer/sales/servers’) is greater than 10,000, the oldest 25% of messages on that topic, or set of topics, is removed.

For example, assume that the number of messages in the database for each topic is as shown in the following table:

Topic name	Number of messages
computer/sales/desktops	100
computer/sales/laptops	8000
computer/sales/PDA	1000
computer/sales/servers	1000
prices/shares/PruneJuice	8000

Messages are removed for topics ’computer/sales/desktops’, ’computer/sales/laptops’, ’computer/sales/PDA’, and ’computer/sales/servers’ because the total number of messages is greater than the highwater mark. But messages for the topic ’prices/shares/PruneJuice’ are not removed because the number of messages on this topic is less than the highwater mark.

Tip: When you use wildcard characters in topic names, be aware of the following conventions for their use:

- -topicsToPrune computer/sales/* prunes all topics whose names start ’computer/sales/’. For example, topics ’computer/sales/desktops’ and ’computer/sales/laptops’ are pruned, but topics ’tests/computer/linux’ and ’windows/drivers/computer’ are not pruned.
- -topicsToPrune *computer* prunes all topics whose names contain ’computer’, regardless of where in the name it occurs. For example, topics ’computer/sales/desktops’, ’computer/sales/laptops’, ’tests/computer/linux’ and ’windows/drivers/computer’ are all pruned.
- -topicsToPrune *computer prunes all topics whose names end ’computer’. For example, topics ’windows/drivers/’ and ’tests/unix/computer’ are pruned, but ’computer/sales/desktops’, ’computer/sales/laptops’, and ’tests/computer/linux’ are not pruned.

Note: Topic names are case-sensitive; for example, the names ’IBM’, ’Ibm’, and ’ibm’ identify different topics.

Pruning selected topics at specific times

```
startpruner -strategy pruneAtTimesOnTopics -topicstoprune
prices*,messages/Football/MyTeam -timestoprune 18:00
-highwatermark 10000 -pruningmethod Percentage
-pruningvariable 25
```

The pruner checks the database at 6 p.m. every day. If the number of messages on the topic ′messages/Football/MyTeam′, or the total number of messages on all the topics whose names begin with ′prices′ is greater than 10,000, the oldest 25% of messages on that topic, or set of topics, is removed.

Pruning messages that were received between specific times

```
startpruner -strategy pruneBetweenTimes
-prunetimestart '2004-10-20 00:00:00'
-prunetimeend '2004-11-05 18:00:00'
```

The pruner removes all messages, on all topics, that were received between midnight on 20th October 2004 and 6 p.m. on 5th November 2004. Pruning occurs immediately.

Pruning messages on selected topics that were received between specific times

```
startpruner -strategy pruneBetweenTimesOnTopics
-prunetimestart '2004-11-19 18:00:00'
-prunetimeend '2004-11-22 09:00:00'
-topicsToPrune computer/sales/*
```

The pruner removes all messages on all topics that have names that begin ′computer/sales/′ and that were received between 6 p.m. on 19th November 2004 and 9 a.m. on 22nd November 2004. Pruning occurs immediately.

Note: The highwatermark parameter is not used.

Programming reference

This section provides reference information about:

- “Java programming interfaces” for reference information about the proprietary Java Message Service (JMS) extensions for message replay.

Java programming interfaces

Purpose

You can use the following proprietary extensions to the Java Message Service (JMS) programming interface to enable your JMS client applications to request message replay from a Replay Server, and to subsequently receive the replayed messages.

This is documented in Javadoc under the doc subdirectory of the Replay installation.

Appendix. 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 information 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 information. The furnishing of this information does not give 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:

- IBM World Trade Asia Corporation
- Licensing
- 2-31 Roppongi 3-chome, Minato-ku
- Tokyo 106, 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 information. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this information 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.

Trademarks

The following terms are trademarks of International Business Machines Corporation in the United States, other countries, or both:

AIX	CICS	DB2
DB2 Universal Database	developerWorks	@server
Everyplace	FFST	First Failure Support Technology
IBM	IMS	IMS/ESA
iSeries	Language Environment	MQSeries
MVS	NetView	OS/400
OS/390	pSeries	RACF
RETAIN	RS/6000	SupportPac
Tivoli	VisualAge	WebSphere
xSeries	z/OS	zSeries

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Pentium is a registered trademark of Intel.

UNIX is a registered trademark in the United States and other countries licensed exclusively through The Open Group.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, and service names may be trademarks or service marks of others.

Printed in USA