

IBM Integration Bus 9.0.0
Version 1 Release 0

*IBM Integration Bus 9.0.0
Administering*

IBM

Note

Before using this information and the product it supports, read the information in "Notices" on page 895.

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

© Copyright IBM Corporation 1999, 2014.

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




PDF books and IBM Knowledge Center

PDF books are provided as a convenience for printing and offline reading. For the latest information, see the online product documentation in IBM® Knowledge Center.

The PDF documentation is updated less frequently than the online product documentation in IBM Knowledge Center.

Links to product documentation topics outside this PDF book go to the online product documentation in IBM Knowledge Center. Links to targets outside a PDF book are marked by icons that indicate whether the target is a PDF book or a web page.

Table 1. Icons that prefix links to topics outside this book

Icon	Description
	A link to a page in the online product documentation in IBM Knowledge Center.
<i>Figure 1.</i>	
	A link to a web page.
<i>Figure 2.</i>	
	A link to a PDF book.
<i>Figure 3.</i>	

Contents

PDF books and IBM Knowledge Center **iii**

Figures **ix**

Chapter 1. Administering brokers and broker resources **1**

Managing brokers	2
Connecting to a local broker	3
Connecting to a remote broker	4
Connecting to a remote integration node on z/OS	6
Importing integration node definitions into the IBM Integration Explorer	8
Exporting integration node connection details from the IBM Integration Explorer	10
Automatically reconnecting to a broker	11
Disconnecting from an integration node in the IBM Integration Explorer	12
Using the Administration Queue	12
Grouping integration nodes by using integration sets	13
Starting and stopping a broker	19
Viewing broker properties	26
Starting a WebSphere MQ queue manager as a Windows service	27
Stopping a WebSphere MQ queue manager when you stop a broker	28
Deleting a broker	29
Managing integration servers	34
Creating an integration server	35
Renaming an integration server using the IBM Integration Explorer	40
Starting an integration server using the IBM Integration Toolkit or IBM Integration Explorer	41
Stopping an integration server using the IBM Integration Toolkit or IBM Integration Explorer	43
Deleting an integration server	45
Managing message flows	50
Setting the start mode of flows and applications at run time	50
Starting a message flow by using the IBM Integration Toolkit or IBM Integration Explorer	53
Stopping a message flow using the IBM Integration Toolkit or IBM Integration Explorer	54
Deleting a message flow using the IBM Integration Toolkit or IBM Integration Explorer	55
Setting user-defined properties dynamically at run time using the IBM Integration Explorer	56
Developing applications that use the IBM Integration API	58
The IBM Integration API	59
The IBM Integration API samples	60
Configuring an environment for developing and running CMP applications	72
Connecting to a broker from a CMP application	79

Navigating brokers and broker resources in a CMP application.	80
Deploying resources to a broker from a CMP application	87
Setting message flow user-defined properties at run time in a CMP application	90
Working with properties of a configurable service of type UserDefined at run time in a JavaCompute node	92
Managing brokers in a CMP application.	94
Managing brokers from JavaCompute nodes	102
Working with resource statistics in a CMP application	103
Working with Activity logs in a CMP application	105
Recording and replaying data with a CMP application	107
Submitting batch requests from a CMP application	109
Examples of IBM Integration API code	110
Administering brokers using the web user interface	127
Configuring the web user interface server	128
Enabling and disabling the web user interface	131
Accessing the web user interface	133
Customizing the data viewer	135
Administering integration nodes from WebSphere Application Server	136
Managing resources used by brokers	137
Listing database connections that the broker holds	137
Quiescing a database	138
Using a JDBC connection pool to manage database resources used by an integration server	139
Managing data caching	141
Administering Java applications	186
Tuning JVM parameters	187
Configuring classloaders for Java user-defined nodes	187
Configuring classloaders for JavaCompute nodes	187
Configuring classloaders for ESQL routines	188
Managing JMS resources	189
Understanding behavior using Activity log	189
Using configurable services.	189
Monitoring JMS resource statistics	189
Accessing Administration log information	190
Viewing Administration log information	190
Saving Administration log information	192
Clearing Administration log information	193
Changing Administration Log view preferences	194
Changing the location of the work path	194
Changing the location of the work path on Windows systems	195
Changing the location of the work path on Linux and UNIX systems	195
Backing up resources	196

Backing up the broker	197
Restoring the broker	198
Backing up the IBM Integration Explorer and IBM Integration Toolkit workspace	200
Chapter 2. Security	203
Security overview	203
Authorization for configuration tasks	205
Security exits	205
Public key cryptography	206
Digital certificates	208
Digital signatures	212
Administration security	213
Administration security overview	214
Setting up administration security	221
Activating broker administration security for WebSphere MQ Version 7.1, or later	241
Message flow security	242
Message flow security overview	243
Setting up message flow security	292
Integration Bus server security	396
Creating user IDs	397
Security for the IBM Integration Toolkit and IBM Integration Explorer	398
Considering security for a broker	407
Configuring broker user IDs on z/OS	409
Integration server user IDs on z/OS	411
Specifying an alternative user ID to run an integration server on z/OS	413
Implementing SSL authentication	415
Chapter 3. Performance, monitoring, and workload management	449
Performance	450
Performance planning	451
Message flow design and performance	453
Code design and performance	455
Analyzing message flow performance	469
Tuning message flow performance	510
Analyzing resource performance	529
Subscribing to statistics reports	542
Tuning the broker	543
Tuning the SAP adapter for scalability and performance	558
Tuning SOAP processing for scalability and performance	559
Troubleshooting performance problems	560
Business-level monitoring	565
Monitoring overview	567
Configuring monitoring event sources using monitoring properties	575
Configuring monitoring event sources using a monitoring profile	580
Activating monitoring	583
Enabling and disabling event sources	585
Monitoring flows by using WebSphere Business Monitor	587
Reporting monitoring settings	593
Recording, viewing, and replaying data	595
Record and replay	597

Enabling security for record and replay	598
Recording data	601
Viewing recorded data	616
Replaying data	618
Using multiple brokers for record and replay	620
Using Activity logs	623
Activity log overview	624
Working with Activity logs in IBM Integration Explorer	625
Viewing and setting runtime properties for Activity logs	630
Writing Activity logs to files	632
Workload management	633
Configure a message flow to cause notification threshold messages to be published	634
Setting the maximum rate for a message flow	637
Unresponsive message flows	639
Configure a workload management policy within Integration Registry	644
Chapter 4. Troubleshooting and support	647
Troubleshooting overview	647
Recording the symptoms of the problem	648
Re-creating the problem	648
Eliminating possible causes	648
Making initial checks	650
Has IBM Integration Bus run successfully before?	650
Did you log off Windows while IBM Integration Bus components were active?	651
Are the Linux and UNIX environment variables set correctly?	652
Are there any error messages or return codes that explain the problem?	653
Can you see all of your files and folders?	654
Can you reproduce the problem?	654
Has the message flow run successfully before?	656
Have you made any changes since the last successful run?	657
Is there a problem with descriptive text for a command?	658
Is there a problem with a database?	659
Is there a problem with the network?	660
Does the problem affect all users?	661
Have you recently changed a password?	661
Have you applied any service updates?	662
Do you have a component that is running slowly?	663
Additional checks for z/OS users	664
Dealing with problems	666
Resolving problems when installing	667
Resolving problems when uninstalling	675
Resolving problems that occur when migrating or importing resources	676
Resolving problems when running commands	679
Resolving problems when running samples	681
Resolving problems when creating resources	685
Resolving problems when renaming resources	687
Resolving problems that occur when you start resources	688

Resolving problems when stopping resources	705	Formatting trace	856
Resolving problems when deleting resources	707	Interpreting trace	859
Resolving problems when developing message flows	708	Clearing old information from trace files	861
Resolving problems when deploying message flows or message sets	752	Changing trace settings from the IBM Integration Explorer	862
Resolving problems that occur when debugging message flows	768	ODBC trace	864
Resolving problems when developing message models	775	IBM Integration API (CMP) trace	867
Resolving problems when using messages	782	Switching Trace nodes on and off	868
Resolving problems when you are writing business rules	796	Using dumps and abend files	871
Resolving problems when you use the IBM Integration Toolkit	797	Checking for dumps	872
Resolving problems when using the IBM Integration Explorer	806	Using the DUMP command on z/OS	873
Resolving problems when using Data Analysis	808	Checking for abend files	875
Resolving problems when using databases	809	Contacting your IBM Support Center	876
Resolving problems when using publish/subscribe	819	IBM Support Assistant Data Collector	879
Resolving problems with performance	823	Collecting data in console mode with IBM Support Assistant Data Collector	880
Resolving problems when you monitor message flows	827	Selecting a problem collector for IBM Support Assistant Data Collector	881
Resolving problems when developing CMP applications	828	Searching knowledge bases	882
Resolving problems with user-defined extensions	829	Getting product fixes	883
Resolving problems when you convert WebSphere Enterprise Service Bus resources	835	Contacting IBM Software Support	885
Using logs	839	Determine the effect of the problem on your business	885
Windows: Viewing the local error log	840	Describe your problem and gather background information	886
Linux and UNIX systems: Configuring the syslog daemon	841	Submit your problem to IBM Software Support	886
z/OS: Viewing broker job logs	843	Recovering after failure	887
Viewing the Eclipse error log	844	Recovering after the broker fails	888
Using trace	845	Recovering after an integration server fails	889
Starting service trace	847	Recovering after the broker queue manager fails	890
Checking service trace options	849	Diagnostic messages	891
Changing service trace options	851	Diagnostic messages: Runtime components	891
Stopping service trace	852	Diagnostic messages: IBM Integration Toolkit	891
Retrieving service trace	854	Diagnostic messages: WebSphere Adapters	892
		Notices	895
		Programming interface information	897
		Trademarks	897
		Sending your comments to IBM	899

Figures

1.	iii
2.	iii
3.	iii
4.	Graphic showing the placement of the global cache in a multi-broker environment.	144
5.	Graphic showing the placement of the global cache in a multi-broker environment.	145
6.	Graphic showing the placement of the global cache in a multi-broker environment.	146
7.	Graphic showing the placement of the global cache in a multiple client scenario.	147
8.	Graphic showing the placement of the global cache in a multiple client scenario.	148
9.	Graphic showing the placement of the global cache in a multiple client scenario.	148
10.	Graphic showing the placement of the global cache in a multiple client scenario.	149
11.	Diagram showing how integration server 1 hosts a catalog server and a container server, and integration servers 2, 3, and 4 host container servers only. Integration servers 5 and 6 do not host any cache components, but their message flows can still communicate with the cache.	156
12.	Diagram showing two brokers that are participating in an embedded cache. Integration server 1 of broker 1 contains a catalog server and a container server. Integration server 1 of broker 2 also contains a catalog server and container server. Integration servers 2, 3, and 4 of broker 1 contain container server. Integration server 2 of broker 2 also contains a container server.	157
13.	Diagram showing how IBM Integration Bus can connect to an embedded cache and a WebSphere eXtreme Scale grid at the same time. Integration server 1 in the broker contains a catalog server and a container server. Integration servers 2, 3, and 4 each host a container server. Double-ended arrows link the message flows in each integration server to the embedded cache and to a remote WebSphere eXtreme Scale grid. Between the message flows and remote grid is a box that represents the configurable service that is used to connect to the external grid.	158
14.	This diagram shows how a client authenticates a server, and is described in the preceding text.	211
15.	This diagram illustrates mutual authentication, and is described in the preceding text.	211
16.	Diagram showing the eight identity properties.	251
17.	Diagram showing the flow of identity authentication.	258
18.	Diagram showing identity authorization.	261
19.	Diagram showing identity mapping.	264
20.	Diagram showing the sequence of events that occur when a message arrives at a security enabled input node in a message flow.	267
21.	Diagram showing the sequence of events that occur when a message arrives at a SecurityPEP node in a message flow.	273
22.	The image is described in the text.	278
23.	Diagram showing the flow of interaction between Message Broker, TFIM V6.2, and TAM.	283
24.	The diagram is described in the surrounding text.	436
25.	The diagram is described in the surrounding text.	445
26.	Java compatibility logo	897

Chapter 1. Administering brokers and broker resources

Administering brokers and associated integration node resources includes the tasks that you perform frequently to activate and manage those resources. Choose the method you prefer to administer your brokers and associated resources.

About this task

Administration of brokers includes the following tasks:

- “Managing brokers” on page 2
- “Managing integration servers” on page 34
- “Managing message flows” on page 50
- “Developing applications that use the IBM Integration API” on page 58
- “Administering brokers using the web user interface” on page 127
- “Administering integration nodes from WebSphere Application Server” on page 136
- “Managing resources used by brokers” on page 137
- “Administering Java applications” on page 186
- “Managing JMS resources” on page 189
- “Accessing Administration log information” on page 190
- “Changing the location of the work path” on page 194
- “Backing up resources” on page 196

These tasks can be performed by using one, or more, of the administrative techniques supported by IBM Integration Bus:

- The IBM Integration Toolkit
- The IBM Integration Explorer
- The IBM Integration Bus commands
- The IBM Integration API (also known as the CMP)
- The Representational State Transfer (REST) API
- The IBM Integration Bus web user interface
- IBM Integration Bus administration for WebSphere® Application Server

For each task, the administrative techniques that you can use are identified.

Related concepts:



The IBM Integration Bus environment

An integration node (broker) is a set of execution processes that hosts one or more message flows to route, transform, and enrich in flight messages.



IBM Integration Toolkit

The IBM Integration Toolkit is an integrated development environment and graphical user interface based on the Eclipse platform.



IBM Integration Explorer

The IBM Integration Explorer is a graphical user interface based on the Eclipse platform for administering your integration nodes (brokers).



The IBM Integration API

The IBM Integration API is a programming interface that your applications can use to control brokers and their resources through a remote interface.

Related tasks:



Deploying a user exit

Deploy your user exit to the broker.

“Analyzing message flow performance” on page 469

You can configure your broker to collect statistics about the operation and behavior of your message flows. Use this information to assess the performance of your message flows.



Configuring brokers for test and production environments

Create one or more brokers on one or more computers, and configure them on your test and production systems to process messages that contain your business data.



Representational State Transfer (REST) API

IBM Integration Bus supports the REST management API, and you can use HTTP clients to administer broker resources.

Related reference:



Administration in z/OS®

In the z/OS environment, commands are issued through the console and others in batch jobs.

Managing brokers

Work with your existing brokers to manage their connections and their active status by using the IBM Integration Toolkit or IBM Integration Explorer. You can use the IBM Integration API (also known as the CMP) to complete some of these actions.

About this task

- “Connecting to a local broker” on page 3
- “Connecting to a remote broker” on page 4
- “Connecting to a remote integration node on z/OS” on page 6
- “Disconnecting from an integration node in the IBM Integration Explorer” on page 12
- “Starting and stopping a broker” on page 19
- Modifying a broker
- “Viewing broker properties” on page 26
- Preparing the environment for WebSphere Adapters nodes
- Preparing the environment for IMS™ nodes
- Preparing the environment for the CICSRequest node
- Changing the operation mode of your broker
- Checking the operation mode of your broker
- Moving from IBM Integration Bus on a distributed system to z/OS
- “Starting a WebSphere MQ queue manager as a Windows service” on page 27
- “Stopping a WebSphere MQ queue manager when you stop a broker” on page 28

- “Deleting a broker” on page 29

Related concepts:



The IBM Integration Bus environment

An integration node (broker) is a set of execution processes that hosts one or more message flows to route, transform, and enrich in flight messages.

Related tasks:



Configuring brokers for development environments

Set up application development environments on Linux on x86 or Windows to create, test, and deploy message flows and associated resources.



Configuring brokers for test and production environments

Create one or more brokers on one or more computers, and configure them on your test and production systems to process messages that contain your business data.

“Managing integration servers” on page 34

Work with your existing integration servers to manage the message flows that you have deployed to them by using the IBM Integration Toolkit or IBM Integration Explorer. You can use the command line, and the IBM Integration API (also known as the CMP) to complete some of these actions.

“Managing message flows” on page 50

Work with your existing messages that you have deployed to integration servers by using the IBM Integration Toolkit or IBM Integration Explorer. You can use the IBM Integration API (also known as the CMP) to complete some of these actions.



Configuring a broker as a WebSphere MQ service

Use these topics to make changes when your broker is operating as a WebSphere MQ service.

Related information:



IBM Integration API (CMP)

Connecting to a local broker

To administer a local broker by using the IBM Integration Toolkit or IBM Integration Explorer, you must connect to the broker.

Before you begin

Before you start:

- Create a broker

About this task

In the IBM Integration Toolkit, local brokers are automatically connected. You must manually connect to remote brokers. In the IBM Integration Explorer you can choose to disconnect from both local and remote brokers. You can also choose to automatically reconnect to a broker when you start the IBM Integration Explorer.

To connect to a broker in the IBM Integration Explorer:

Procedure

1. In the Navigator view, expand the Integration Nodes folder.
2. Right-click the broker you want to connect to, and click **Connect**.

Results

You can now view properties and configure your broker by using the IBM Integration Toolkit or the IBM Integration Explorer.

Related concepts:

IBM Integration Explorer

The IBM Integration Explorer is a graphical user interface based on the Eclipse platform for administering your integration nodes (brokers).

Related tasks:

“Connecting to a remote broker”

To administer a remote broker by using the IBM Integration Toolkit or IBM Integration Explorer, you must connect to the broker.

“Automatically reconnecting to a broker” on page 11

You can configure each broker so that the IBM Integration Explorer automatically reconnects to it if the connection is lost; for example, if the network connection to a remote queue manager fails.

Creating an integration node using the IBM Integration Explorer

On Linux on x86 or Windows, you can create integration nodes (brokers) by using the IBM Integration Explorer.

“Creating an integration server using the IBM Integration Toolkit or IBM Integration Explorer” on page 36

Use the IBM Integration Toolkit or IBM Integration Explorer to create integration servers on your broker.

Chapter 1, “Administering brokers and broker resources,” on page 1

Administering brokers and associated integration node resources includes the tasks that you perform frequently to activate and manage those resources. Choose the method you prefer to administer your brokers and associated resources.

Connecting to a remote broker

To administer a remote broker by using the IBM Integration Toolkit or IBM Integration Explorer, you must connect to the broker.

Before you begin

Before you start:

- Create a broker

Before you can connect to a remote broker, the broker and its queue manager must be running. You must also complete the following steps:

- Ensure that a command server is running on the queue manager.
- Ensure a server-connection channel has been defined on the broker queue manager. When you create a broker, a default SVRCONN channel, called SYSTEM.BKR.CONFIG, is created. This channel supports connections from one or more remote clients to the broker.
- Create a TCP/IP listener on the broker queue manager.
- Ensure that the listener is running.
- Create a group and a user ID on the computer where the broker is running by using the local system facilities; for example, the **groupadd** command on Linux on x86. Add the user ID to the group.

- Run the **setmqaut** command on the broker queue manager to grant authority to the group:

```
setmqaut -m queueManager -t qmgr -g group +connect
```

About this task

You can create a connection to a remote broker, or you can create a connection to a remote broker by using settings from a `.broker` file. Right-click the Integration Nodes folder, and click **Connect to a Remote Broker Using *.broker File**. Select the `.broker` file, and click **Open**. The connection to the remote broker is created in your workspace.

To create a connection to a remote broker:

Procedure

1. Open the IBM Integration Explorer, or open the Integration Nodes view in the IBM Integration Toolkit.
2. To create a connection to a remote broker, right-click the Integration Nodes folder, and click **Connect to a Remote Broker**.
3. In the Connect to a broker wizard, enter the following values:
 - a. The value for the **Queue Manager name** that your remote broker is using.
 - b. The **Host** name or IP address of the computer on which your broker is running.
 - c. The **TCP Port** on which the WebSphere MQ queue manager is listening (the default is 1414). This property must be a valid positive number.
 - d. Optional: The name of the server-connection channel in the **SVRCONN Channel Name** field. The channel has a default name of `SYSTEM.BKR.CONFIG`.

You can create more than one server-connection channel and define a different SSL certificate on each channel to enforce; for example, users with view access on to one channel and users with deploy access on to a different channel.

You can then create WebSphere MQ exits on each channel to provide additional authentication of the WebSphere MQ message sent to the broker.

You must create the server-connection channel manually on the broker queue manager by using one of the following options:

- The WebSphere MQ **runmqsc** command to create a channel with options `CHLTYPE(SVRCONN)` and `TRPTYPE(TCP)`.
- The WebSphere MQ Explorer to create a server-connection channel with the transmission protocol set to TCP.

For more information see your WebSphere MQ documentation.

If you do not change the name, or attempt to delete it, the default name of `SYSTEM.BKR.CONFIG` is assumed. The name of the server-connection channel is changed only if you enter another name in place of `SYSTEM.BKR.CONFIG`.

- e. Optional: The **Class** of the Security Exit required to connect to the WebSphere MQ queue manager. This property must be a valid Java™ class name, but you can leave this field blank if it does not apply to your domain connection.
- f. Optional: The **JAR File Location** for the Security Exit required to connect to the WebSphere MQ queue manager. Click **Browse** to find the file location. If this field does not apply to your domain connection, you can leave it blank. If you enter a Security Exit **Class**, you must provide a **JAR File Location**.

- g. Optional: The **Cipher Suite**, **Distinguished Names**, **CRL Name List**, **Key Store**, and **Trust Store** parameters are required to enable SSL. For more information, see “Implementing SSL authentication” on page 415. The **Cipher Suite** field lists available cipher suites. Click **More** to configure Custom SSL Cipher Suites in the Broker Administration Preferences window. If a **Cipher Suite** is not specified, all of the other fields in the SSL section are unavailable.

If you specify a keystore or truststore when you define the connection information, you are prompted for the keystore or truststore when you connect to the remote broker.

4. Click **Finish** to connect to the remote broker.

Results

You can now view properties and configure your broker using the IBM Integration Toolkit or IBM Integration Explorer.

Related concepts:



IBM Integration Explorer

The IBM Integration Explorer is a graphical user interface based on the Eclipse platform for administering your integration nodes (brokers).

“Security exits” on page 205

Use security exit programs to verify that the partner at the other end of a connection is genuine.

“Security for the IBM Integration Toolkit and IBM Integration Explorer” on page 398

What to consider when you set up security for the IBM Integration Toolkit and IBM Integration Explorer.

Related tasks:



Creating an integration node using the IBM Integration Explorer

On Linux on x86 or Windows, you can create integration nodes (brokers) by using the IBM Integration Explorer.

“Creating an integration server using the IBM Integration Toolkit or IBM Integration Explorer” on page 36

Use the IBM Integration Toolkit or IBM Integration Explorer to create integration servers on your broker.

Chapter 1, “Administering brokers and broker resources,” on page 1

Administering brokers and associated integration node resources includes the tasks that you perform frequently to activate and manage those resources. Choose the method you prefer to administer your brokers and associated resources.

Connecting to a remote integration node on z/OS

To administer a remote integration node (broker) that is deployed on z/OS by using the IBM Integration Toolkit or IBM Integration Explorer, you must connect to the integration node.

Before you begin

Before you start:

- Create a broker

About this task

To connect to a remote integration node on z/OS:

Procedure

1. In the Navigator view, or Integration Nodes view, right-click the Integration Nodes folder, and click **Connect to a Remote Integration Node**.
2. In the Connect to an integration node wizard, enter the following values:
 - a. The value for the **Queue Manager name** that your remote integration node is using.
 - b. The **Host** name or IP address of the machine on which your integration node is running.
 - c. The TCP **Port** on which the WebSphere MQ queue manager is listening (the default is 1414). This property must be a valid positive number.
 - d. Optional: The name of the server-connection channel in the **SVRCONN Channel Name** field. The channel has a default name of SYSTEM.BKR.CONFIG.

You can create more than one server-connection channel and define a different SSL certificate on each channel to enforce; for example, users with view access on to one channel and users with deploy access on to a different channel.

You can then create WebSphere MQ exits on each channel to provide additional authentication of the WebSphere MQ message sent to the integration node.

You must create the server-connection channel manually on the queue manager for the integration node by using one of the following options:

- The WebSphere MQ **runmqsc** command to create a channel with options CHLTYPE(SVRCONN) and TRPTYPE(TCP).
- The WebSphere MQ Explorer to create a server-connection channel with the transmission protocol set to TCP.

For more information see your WebSphere MQ documentation.

The default name of SYSTEM.BKR.CONFIG is assumed if you do not change the name, or attempt to delete it. The name of the server-connection channel is changed only if you enter another name in place of SYSTEM.BKR.CONFIG.

- e. Optional: The **Class** of the Security Exit required to connect to the WebSphere MQ queue manager. This property must be a valid Java class name, but you can leave this field blank if it does not apply to your domain connection.
 - f. Optional: The **JAR File Location** for the Security Exit required to connect to the WebSphere MQ queue manager. Click **Browse** to find the file location. You can leave this field blank if it does not apply to your domain connection. You must provide a **JAR File Location** if you enter a Security Exit **Class**.
 - g. Optional: The **Cipher Suite, Distinguished Names, CRL Name List, Key Store, and Trust Store** parameters are required to enable SSL. For more information, see "Implementing SSL authentication" on page 415. The **Cipher Suite** field displays available cipher suites. Click **More** to configure Custom SSL Cipher Suites in the IBM Integration Preferences window. If a **Cipher Suite** is not specified, all of the other fields in the SSL section are unavailable.
3. Click **Finish** to connect to the remote integration node.

Results

You can now view properties and configure your integration node by using the IBM Integration Toolkit or IBM Integration Explorer.

Related concepts:

IBM Integration Explorer

The IBM Integration Explorer is a graphical user interface based on the Eclipse platform for administering your integration nodes (brokers).

“Security exits” on page 205

Use security exit programs to verify that the partner at the other end of a connection is genuine.

“Security for the IBM Integration Toolkit and IBM Integration Explorer” on page 398

What to consider when you set up security for the IBM Integration Toolkit and IBM Integration Explorer.

Related tasks:

Creating an integration node using the IBM Integration Explorer

On Linux on x86 or Windows, you can create integration nodes (brokers) by using the IBM Integration Explorer.

“Creating an integration server using the IBM Integration Toolkit or IBM Integration Explorer” on page 36

Use the IBM Integration Toolkit or IBM Integration Explorer to create integration servers on your broker.

Chapter 1, “Administering brokers and broker resources,” on page 1

Administering brokers and associated integration node resources includes the tasks that you perform frequently to activate and manage those resources. Choose the method you prefer to administer your brokers and associated resources.

“Connecting to a local broker” on page 3

To administer a local broker by using the IBM Integration Toolkit or IBM Integration Explorer, you must connect to the broker.

“Connecting to a remote broker” on page 4

To administer a remote broker by using the IBM Integration Toolkit or IBM Integration Explorer, you must connect to the broker.

Importing integration node definitions into the IBM Integration Explorer

Import integration node connection details that have been created by another user into your session of the IBM Integration Explorer or IBM Integration Toolkit, to use the integration node.

About this task

The properties of an integration node connection can be exported from one instance of the IBM Integration Explorer, and imported into a different instance. You can also import an integration node connection into an instance of the IBM Integration Toolkit. The properties of the integration node connection are stored in a `.broker` file.

Importing an integration node definition into the IBM Integration Toolkit

About this task

To import integration node connection properties from a `.broker` file into the IBM Integration Toolkit:

Procedure

1. In the Integration Nodes view, right-click the Integration Nodes folder and click **Connect to a Remote Integration Node Using *.broker File**.
2. Navigate to the integration node connection files that you want to import, and click **Open**.

Results

The `.broker` file is imported, and the integration node is displayed in the Integration Nodes folder. You can now connect to the integration node.

Importing an integration node definition into the IBM Integration Explorer

About this task

To import integration node connection properties from a `.broker` file into the IBM Integration Explorer:

Procedure

1. In the Navigator view, right-click the Integration Nodes folder and click **Connect to a remote integration node using *.broker file**.
2. Navigate to the integration node connection files that you want to import, and click **OK**.

Results

The `.broker` file is imported, and the integration node is displayed in the Integration Nodes folder. You can now connect to the integration node.

Related concepts:



IBM Integration Explorer

The IBM Integration Explorer is a graphical user interface based on the Eclipse platform for administering your integration nodes (brokers).

Related tasks:



Configuring integration nodes using the IBM Integration Explorer

Configure your local and remote integration nodes (brokers) by using the IBM Integration Explorer.

“Exporting integration node connection details from the IBM Integration Explorer” on page 10

Use the IBM Integration Explorer to export integration node (broker) connection details for another user.

“Connecting to a remote broker” on page 4

To administer a remote broker by using the IBM Integration Toolkit or IBM Integration Explorer, you must connect to the broker.

“Connecting to a remote integration node on z/OS” on page 6

To administer a remote integration node (broker) that is deployed on z/OS by using the IBM Integration Toolkit or IBM Integration Explorer, you must connect to the integration node.

Exporting integration node connection details from the IBM Integration Explorer

Use the IBM Integration Explorer to export integration node (broker) connection details for another user.

About this task

The properties of an integration node connection can be exported from one instance of the IBM Integration Explorer and imported into a different instance. The properties of the integration node connection are stored in a `.broker` file.

To export integration node (broker) connection properties to a `.broker` file, complete the following steps.

Procedure

1. In the Navigator view, expand the Integration Nodes folder.
2. Select the integration node (broker) for which you want to export the connection details, and click **Export for remote connection**.
3. Select a location to save the `.broker` file, and click **Save**.

Results

The `.broker` file is exported. Another user can now import the `.broker` file to connect to the integration node (broker).

Related concepts:



IBM Integration Explorer

The IBM Integration Explorer is a graphical user interface based on the Eclipse platform for administering your integration nodes (brokers).

Related tasks:



Configuring integration nodes using the IBM Integration Explorer

Configure your local and remote integration nodes (brokers) by using the IBM Integration Explorer.

“Importing integration node definitions into the IBM Integration Explorer” on page 8

Import integration node connection details that have been created by another user into your session of the IBM Integration Explorer or IBM Integration Toolkit, to use the integration node.

“Connecting to a remote broker” on page 4

To administer a remote broker by using the IBM Integration Toolkit or IBM Integration Explorer, you must connect to the broker.

“Connecting to a remote integration node on z/OS” on page 6

To administer a remote integration node (broker) that is deployed on z/OS by using the IBM Integration Toolkit or IBM Integration Explorer, you must connect to the integration node.

Automatically reconnecting to a broker

You can configure each broker so that the IBM Integration Explorer automatically reconnects to it if the connection is lost; for example, if the network connection to a remote queue manager fails.

About this task

If you manually disconnect an integration node (broker), the integration node is not automatically reconnected until the next time that you close and restart the IBM Integration Explorer. To configure an integration node to automatically reconnect when you start the IBM Integration Explorer:

Procedure

1. In the Navigator view, expand the Integration Nodes folder.
2. Right-click your integration node, and select **Properties**.
3. Ensure **Autoreconnect** is selected in the General tab.
4. Click **OK**.

Results

The integration node is automatically reconnected when you next start the IBM Integration Explorer.

What to do next

If you want to configure the integration node so that it is not automatically reconnected when you start the IBM Integration Explorer, clear **Autoreconnect** in the integration node properties window.

Related concepts:



IBM Integration Explorer

The IBM Integration Explorer is a graphical user interface based on the Eclipse platform for administering your integration nodes (brokers).

Related tasks:



Creating an integration node using the IBM Integration Explorer

On Linux on x86 or Windows, you can create integration nodes (brokers) by using the IBM Integration Explorer.

“Connecting to a local broker” on page 3

To administer a local broker by using the IBM Integration Toolkit or IBM Integration Explorer, you must connect to the broker.

“Connecting to a remote broker” on page 4

To administer a remote broker by using the IBM Integration Toolkit or IBM Integration Explorer, you must connect to the broker.

“Connecting to a remote integration node on z/OS” on page 6

To administer a remote integration node (broker) that is deployed on z/OS by using the IBM Integration Toolkit or IBM Integration Explorer, you must connect to the integration node.

Disconnecting from an integration node in the IBM Integration Explorer

You can disconnect from all integration nodes (brokers) that you are not currently configuring to improve performance in the IBM Integration Explorer.

About this task

To disconnect from an integration node in IBM Integration Explorer:

Procedure

1. In the Navigator view, expand the Integration Nodes folder.
2. Right-click the integration node you want to disconnect from, and click **Disconnect**.

Results

You have disconnected from the selected integration node.

Related concepts:



IBM Integration Explorer

The IBM Integration Explorer is a graphical user interface based on the Eclipse platform for administering your integration nodes (brokers).

Related tasks:



Creating an integration node using the IBM Integration Explorer

On Linux on x86 or Windows, you can create integration nodes (brokers) by using the IBM Integration Explorer.

“Connecting to a local broker” on page 3

To administer a local broker by using the IBM Integration Toolkit or IBM Integration Explorer, you must connect to the broker.

“Creating an integration server using the IBM Integration Toolkit or IBM Integration Explorer” on page 36

Use the IBM Integration Toolkit or IBM Integration Explorer to create integration servers on your broker.

Chapter 1, “Administering brokers and broker resources,” on page 1

Administering brokers and associated integration node resources includes the tasks that you perform frequently to activate and manage those resources. Choose the method you prefer to administer your brokers and associated resources.

Using the Administration Queue

The Administration Queue shows pending and submitted tasks that have been sent to the broker, and are waiting to be processed. Use the IBM Integration Explorer to view details of the tasks, and to cancel tasks in the queue.

About this task

Each broker that is connected in the IBM Integration Explorer displays an icon labeled **Administration Queue**. When you click the Administration Queue icon, tasks submitted to the broker are displayed in the Administration Queue QuickView in the Content pane. The Administration Queue icon changes to indicate the number of messages on the Administration Queue.

The following properties are displayed for each task on the Administration Queue:

- Order
- Status
- Username
- Operation Type
- Object Name
- Object Type
- Creation Time
- Elapsed Time
- Identifier

You can remove tasks from the Administration Queue if you want to cancel requests that have been sent to a broker. To remove tasks from the Administration Queue:

Procedure

1. In the Navigator view, expand the Integration Nodes folder.
2. Expand the broker with which you want to work, and right-click **Administration Queue**.
3. Click **Cancel Work Items** The Administration Queue window is displayed.
4. Select the task or tasks that you want to remove from the Administration Queue, and click **Cancel Work Items**.

Results

The selected tasks are removed from the Administration Queue.

Related concepts:



IBM Integration Explorer

The IBM Integration Explorer is a graphical user interface based on the Eclipse platform for administering your integration nodes (brokers).

Related tasks:



Configuring integration nodes using the IBM Integration Explorer

Configure your local and remote integration nodes (brokers) by using the IBM Integration Explorer.

Chapter 1, “Administering brokers and broker resources,” on page 1

Administering brokers and associated integration node resources includes the tasks that you perform frequently to activate and manage those resources. Choose the method you prefer to administer your brokers and associated resources.

Chapter 3, “Performance, monitoring, and workload management,” on page 449

You can change various aspects of your broker configuration to tune brokers and message flows, and monitor message flows and publish/subscribe applications.

Grouping integration nodes by using integration sets

You can create a manual or an automatic integration set to visually group your integration nodes (brokers) in the IBM Integration Explorer.

About this task

If you have a large number of integration nodes displayed in the IBM Integration Explorer, you might find it helpful to group the integration nodes by using integration sets. You can create manual integration sets, automatic integration sets, or a combination of both types. When you create an integration set for the first time, two sets are created; a set called All and a set with a name that you provided

when you created it. The All integration set contains all your local integration nodes, and any remote integration nodes definitions in your workspace. If you delete all of the integration sets that you have created, the All integration set is automatically removed. You cannot rename, modify, or delete the All integration set manually.

A manual integration set is empty until you add integration nodes to the integration set from the All integration set or another integration set. You can add or remove integration nodes from the manual integration set at any time.

An automatic integration set uses integration node tags to dynamically add and remove integration nodes from a set based on values you provide or the current state of the integration nodes.

See the following topics for more information about how to create and use integration sets in the IBM Integration Explorer:

- “Creating a manual integration set in the IBM Integration Explorer”
- “Adding and modifying integration node tags in the IBM Integration Explorer” on page 15
- “Creating an automatic integration set in the IBM Integration Explorer” on page 17
- “Modifying integration sets in the IBM Integration Explorer” on page 18

Creating a manual integration set in the IBM Integration Explorer

You can create a manual integration set, and add integration nodes to visually group your integration nodes in the IBM Integration Explorer.

Before you begin

Before you start:

- Create an integration node (broker)

About this task

A manual integration set is empty until you add integration nodes to the integration set from the All integration set, or from another integration set. You can add or remove integration nodes from the manual integration set at any time.

To create a manual integration set in the IBM Integration Explorer:

Procedure

1. In the Navigator view right-click the Integration Nodes folder, and click **Integration Sets > New Integration Set**. The Create New Integration Set wizard is displayed.
2. Enter a name for the integration set.
3. Ensure **Manual** is selected as the set type.
4. Click **Finish**.

Results

The integration set is added to the Integration Nodes folder. If this set was the first integration set that you have created, the All integration set is also added to the Integration Nodes folder. The All integration set contains all your local integration nodes, and all remote integration node definitions in your workspace.

What to do next

You can now move integration nodes from your All integration set, or other integration sets, into the manual integration set that you have created.

Related concepts:



IBM Integration Explorer

The IBM Integration Explorer is a graphical user interface based on the Eclipse platform for administering your integration nodes (brokers).

Related tasks:

“Managing brokers” on page 2

Work with your existing brokers to manage their connections and their active status by using the IBM Integration Toolkit or IBM Integration Explorer. You can use the IBM Integration API (also known as the CMP) to complete some of these actions.



Configuring integration nodes using the IBM Integration Explorer

Configure your local and remote integration nodes (brokers) by using the IBM Integration Explorer.

“Grouping integration nodes by using integration sets” on page 13

You can create a manual or an automatic integration set to visually group your integration nodes (brokers) in the IBM Integration Explorer.

“Creating an automatic integration set in the IBM Integration Explorer” on page 17

Create an automatic integration set in the IBM Integration Explorer to group integration nodes (brokers) dynamically based on values you provide, or their current state.

Adding and modifying integration node tags in the IBM Integration Explorer

Add or modify the integration node tags on your integration nodes (brokers) to change the integration nodes that are grouped in an automatic set in the IBM Integration Explorer.

Before you begin

Before you start:

- Create a broker

About this task

An automatic integration set uses integration node tags to dynamically add and remove integration nodes from a set based on values you provide or the current state of the integration nodes. You must add the appropriate tags to your integration nodes before they can be dynamically added to an automatic integration set. You can select from the following included integration node tags, or you can configure your own custom integration node tags:

- brokerEnvironment:Development
- brokerEnvironment:Test
- brokerEnvironment:QA
- brokerEnvironment:Production

You can also remove tags that you have previously added to an integration node to stop the integration node being added to an automatic set.

To add or modify integration node tags in the IBM Integration Explorer:

Procedure

1. In the Navigator view, expand the Integration Nodes folder.
2. Right-click the integration node for which you want to add or modify integration node tags, and click **Modify Integration Node Tags**.
3. Select the tags to add to the integration node:
 - Select integration node tags from the available list.
 - Add a custom integration node tag:
 - a. Enter a name for the tag in the **Tags** field.
 - b. Click **Add**.
You can also select a tag and click **Remove** to remove it from the list of available integration node tags.
 - Clear tags that you want to remove from the integration node.
4. Click **Finish**.

Results

The selected integration node tags are added to the integration node. If the integration node tags are matched in an automatic integration set, the integration node appears in the integration set. If the integration node tags no longer match in an automatic integration set, the integration node no longer appears in the integration set.

What to do next

You can create or modify an automatic integration set to use one or more of the integration node tags that you have created.

Related concepts:



IBM Integration Explorer

The IBM Integration Explorer is a graphical user interface based on the Eclipse platform for administering your integration nodes (brokers).

Related tasks:

“Grouping integration nodes by using integration sets” on page 13

You can create a manual or an automatic integration set to visually group your integration nodes (brokers) in the IBM Integration Explorer.

“Creating a manual integration set in the IBM Integration Explorer” on page 14

You can create a manual integration set, and add integration nodes to visually group your integration nodes in the IBM Integration Explorer.

“Creating an automatic integration set in the IBM Integration Explorer” on page 17

Create an automatic integration set in the IBM Integration Explorer to group integration nodes (brokers) dynamically based on values you provide, or their current state.

“Modifying integration sets in the IBM Integration Explorer” on page 18

Modify an automatic integration set in the IBM Integration Explorer to change or add integration node tags used to filter integration nodes (brokers) in the integration set.

Creating an automatic integration set in the IBM Integration Explorer

Create an automatic integration set in the IBM Integration Explorer to group integration nodes (brokers) dynamically based on values you provide, or their current state.

Before you begin

Before you start:

- Create a broker
- Add broker tags to your brokers

About this task

An automatic integration set uses integration node tags to dynamically add and remove integration nodes from a set based on values you provide or the current state of the integration nodes. You must add the appropriate tags to your integration nodes before they can be dynamically added to an automatic integration set. You can select a single tag, or multiple tags, for an automatic integration set.

You can select from the following included integration set, or any custom tags you have added:

- brokerEnvironment:Development
- brokerEnvironment:Test
- brokerEnvironment:QA
- brokerEnvironment:Production
- brokerStatus:Started
- brokerStatus:Stopped
- brokerStatus:Connected
- brokerStatus:Disconnected

To create an automatic integration set:

Procedure

1. In the Navigator view right-click the Integration Nodes folder, and click **Integration Sets > New Integration Set**. The Create New Integration Set wizard is displayed.
2. Enter a name for the integration set.
3. Ensure **Automatic** is selected as the set type.
4. Click **Next**.
5. Select the conditions under which you want the integration node to be a part of the integration set from the list of available filters. Click the **Add** button to move them into the selected filters pane.
6. Decide if you want the integration node to match all the selected conditions, or one or more of them:
 - If you want the integration node to match all the selected conditions ensure **matches ALL of the selected filters** is selected.
 - If you want the integration node to match one or more of the selected conditions ensure **matches ANY of the selected filters** is selected.
7. Click **Finish**.

Results

The integration set is added to the Integration Nodes folder, and all integration nodes that match the selected filters are added to the integration set. If this set is the first integration set that you have created, the All integration nodes set is also added to the Integration Nodes folder. The All integration nodes set contains all your local integration nodes, and all remote integration node definitions in your workspace.

Related concepts:



IBM Integration Explorer

The IBM Integration Explorer is a graphical user interface based on the Eclipse platform for administering your integration nodes (brokers).

Related tasks:

“Grouping integration nodes by using integration sets” on page 13

You can create a manual or an automatic integration set to visually group your integration nodes (brokers) in the IBM Integration Explorer.

“Creating a manual integration set in the IBM Integration Explorer” on page 14

You can create a manual integration set, and add integration nodes to visually group your integration nodes in the IBM Integration Explorer.

“Adding and modifying integration node tags in the IBM Integration Explorer” on page 15

Add or modify the integration node tags on your integration nodes (brokers) to change the integration nodes that are grouped in an automatic set in the IBM Integration Explorer.

“Modifying integration sets in the IBM Integration Explorer”

Modify an automatic integration set in the IBM Integration Explorer to change or add integration node tags used to filter integration nodes (brokers) in the integration set.

Modifying integration sets in the IBM Integration Explorer

Modify an automatic integration set in the IBM Integration Explorer to change or add integration node tags used to filter integration nodes (brokers) in the integration set.

Before you begin

Before you start:

- Create a broker
- Create an automatic broker set

About this task

An automatic integration set uses integration node tags to dynamically add and remove integration nodes from a set based on values you provide or the current state of the integration nodes. You must add the appropriate tags to your integration nodes before they can be dynamically added to an automatic integration set. You can select from the following included integration node tags or any custom tags you have added:

- brokerEnvironment:Development
- brokerEnvironment:Test
- brokerEnvironment:QA
- brokerEnvironment:Production

- brokerStatus:Started
- brokerStatus:Stopped
- brokerStatus:Connected
- brokerStatus:Disconnected

To modify an automatic integration set in the IBM Integration Explorer

Procedure

1. In the Navigator view expand the Integration Nodes folder.
2. Right-click the integration set that you want to modify, and click **Modify Integration Set**.
3. Select the conditions under which you want the integration node to be a part of the integration set from the list of available filters:
 - Click the **Add** button to move them into the selected filters pane.
 - Click the **Remove** button to remove them from the selected filters pane.
4. Decide if you want the integration node to match all the selected conditions or one or more of them:
 - If you want the integration node to match all the selected conditions ensure **matches ALL of the selected filters** is selected.
 - If you want the integration node to match one or more of the selected conditions ensure **matches ANY of the selected filters** is selected.
5. Click **Finish**.

Results

The integration set is modified, and all integration nodes that match the selected filters are added to the integration set.

Related concepts:



IBM Integration Explorer

The IBM Integration Explorer is a graphical user interface based on the Eclipse platform for administering your integration nodes (brokers).

Related tasks:

“Grouping integration nodes by using integration sets” on page 13

You can create a manual or an automatic integration set to visually group your integration nodes (brokers) in the IBM Integration Explorer.

“Creating a manual integration set in the IBM Integration Explorer” on page 14

You can create a manual integration set, and add integration nodes to visually group your integration nodes in the IBM Integration Explorer.

“Creating an automatic integration set in the IBM Integration Explorer” on page 17

Create an automatic integration set in the IBM Integration Explorer to group integration nodes (brokers) dynamically based on values you provide, or their current state.

“Adding and modifying integration node tags in the IBM Integration Explorer” on page 15

Add or modify the integration node tags on your integration nodes (brokers) to change the integration nodes that are grouped in an automatic set in the IBM Integration Explorer.

Starting and stopping a broker

Run the appropriate command to start or stop a broker.

Before you begin

Before you start:

- Ensure that your user ID has the correct authorizations to perform the task. Refer to Security requirements for administrative tasks.
- On Linux, UNIX, and Windows systems, you must set up your command-line environment before performing this task, by running the product profile or console; refer to Setting up a command environment.

About this task

To start and stop a broker, you can use the **mqsistart** and **mqsistop** commands from the command line. Alternatively, on Windows and Linux, use the IBM Integration Toolkit or the IBM Integration Explorer to start and stop brokers.

To start or stop a broker using the IBM Integration Toolkit or the IBM Integration Explorer, follow the appropriate link:

- “Starting a local broker using the IBM Integration Toolkit or IBM Integration Explorer” on page 24
- “Stopping a local broker using the IBM Integration Toolkit or IBM Integration Explorer” on page 25

To start or stop a broker using commands from the command line, follow the link for the appropriate platform:

Procedure

- “Starting and stopping a broker on Linux and UNIX systems” on page 21
- “Starting and stopping a broker on Windows” on page 21
- “Starting and stopping a broker on z/OS” on page 23

Related concepts:



The IBM Integration Bus environment

An integration node (broker) is a set of execution processes that hosts one or more message flows to route, transform, and enrich in flight messages.

Related tasks:



Setting up a command environment

After you have installed the product on one of the distributed systems, you must initialize the environment before you can use a runtime component or command.

Related reference:



Security requirements for administrative tasks

You can configure access permissions to govern which users and groups can manipulate objects in the broker network. Security requirements for administrative tasks depend on the platform that you use.



mqsistart command

Use the **mqsistart** command to start the specified broker if all initial verification tests complete successfully.



mqsistop command

Use the **mqsistop** command to stop the specified component.

Starting and stopping a broker on Linux and UNIX systems

Run the appropriate command to start or stop a broker.

Procedure

1. Run `./install_dir/bin/mqsiprofile` to source the `mqsiprofile` script and set up the environment for a single targeted runtime environment. You must complete this setup before you can run an IBM Integration Bus command.

2. To start a broker, enter the following command on the command line:

```
mqsistart IB9NODE
```

Substitute your own broker name for `IB9NODE`. The broker and its associated queue manager are started.

Check the system log to ensure that the broker has initialized successfully. The log contains messages about verification procedures; if all tests are successful, only an initial start message is recorded. If any verification test is unsuccessful, the log also includes messages that provide details of the tests that have failed. If errors have been reported, review the messages and take the suggested actions to resolve these problems.

3. To stop a broker, enter the following command on the command line:

```
mqsistop IB9NODE
```

Substitute your own broker name for `IB9NODE`.

You can also request that the broker queue manager is stopped by this command. Refer to “Stopping a WebSphere MQ queue manager when you stop a broker” on page 28.

Related concepts:



The IBM Integration Bus environment

An integration node (broker) is a set of execution processes that hosts one or more message flows to route, transform, and enrich in flight messages.

Related tasks:



Creating a broker

You can create brokers on every platform that is supported by IBM Integration Bus. The broker runs as a 64-bit application on all platforms except Linux on x86 and Windows on x86.

“Linux and UNIX systems: Configuring the syslog daemon” on page 841

On Linux and UNIX systems, all IBM Integration Bus messages (other than messages that are generated by the command-line utilities) are sent to the syslog subsystem.

“Stopping a WebSphere MQ queue manager when you stop a broker” on page 28

If you are preparing to stop a broker, you can stop the broker's WebSphere MQ queue manager at the same time.

Related reference:



`mqsistart` command

Use the `mqsistart` command to start the specified broker if all initial verification tests complete successfully.



`mqsistop` command

Use the `mqsistop` command to stop the specified component.

Starting and stopping a broker on Windows

Run the appropriate command to start or stop a broker.

Procedure

1. Open the IBM Integration Bus command console. When you open the console, it sets up the environment that you need to run the IBM Integration Bus commands.

If you prefer, you can run the `install_dir/bin/mqsiprofile` command to set up the environment.

2. To start a broker, enter the following command on the command line:

```
mqsistart IB9NODE
```

Substitute your own broker name for IB9NODE.

You can also request that the broker queue manager is started as a Windows service. Refer to “Starting a WebSphere MQ queue manager as a Windows service” on page 27.

The broker and its associated queue manager are started. The command initiates the startup of the broker's Windows service.

Check the Application Log in the Event Viewer to ensure that the broker has initialized successfully. The log contains messages about verification procedures; if all tests are successful, only an initial start message is recorded. If one or more verification tests are unsuccessful, the log also includes messages that provide details of the tests that have failed. If errors have been reported, review the messages and take the suggested actions to resolve these problems.

3. To stop a broker, enter the following command on the command line:

```
mqsistop IB9NODE
```

Substitute your own broker name for IB9NODE.

You can also request that the broker queue manager is stopped by this command. Refer to “Stopping a WebSphere MQ queue manager when you stop a broker” on page 28.

Related concepts:



The IBM Integration Bus environment

An integration node (broker) is a set of execution processes that hosts one or more message flows to route, transform, and enrich in flight messages.

Related tasks:



Creating a broker on Windows

On Windows, you can create brokers on the command line.

“Starting a WebSphere MQ queue manager as a Windows service” on page 27

On Windows, you can start a WebSphere MQ queue manager as a Windows service to ensure the queue manager starts when you start your other components.

“Stopping a WebSphere MQ queue manager when you stop a broker” on page 28

If you are preparing to stop a broker, you can stop the broker's WebSphere MQ queue manager at the same time.

“Windows: Viewing the local error log” on page 840

The Windows Event Viewer is where IBM Integration Bus writes records to the local system. Use Windows system facilities to view this log.

Related reference:



mqsistart command

Use the **mqsistart** command to start the specified broker if all initial verification tests complete successfully.

mqsistop command

Use the **mqsistop** command to stop the specified component.

Starting and stopping a broker on z/OS

Run the appropriate command from SDSF to start or stop a broker.

Procedure

1. Start the component by using the command `/S <broker name>`. This command produces the following output, where MQP1BRK is the name of the broker:

```
+BIP9141I MQP1BRK 0 The component was started
```

Substitute your own broker name for MQP1BRK.

The verification step runs, followed by starting the control process and any DataFlowEngine (integration server) address spaces.

If the verification step fails, the errors are reported to the STDOUT stream in the JOBLLOG. The control process and DataFlowEngine address spaces are not started. Review the messages to see what errors have been reported, and take the suggested actions to resolve these problems.

2. Alternatively, start the control process only by using the command:

```
/S broker_name,STRTP=MAN
```

If the verification step fails for any reason, the errors are reported to the STDOUT stream in the JOBLLOG; the control process is not started. Review the messages to see what errors have been reported, and take the suggested actions to resolve these problems.

No DataFlowEngine address spaces are started automatically if you specify STRTP=MAN. If the verification step is successful and the control process starts successfully, fully start the broker by issuing the console command:

```
/F <broker name>, SC
```

3. To stop a broker, run the following command:

```
/P <broker name>
```

Related concepts:

The IBM Integration Bus environment

An integration node (broker) is a set of execution processes that hosts one or more message flows to route, transform, and enrich in flight messages.

Related tasks:

Creating a broker on z/OS

Create the broker component and the other resources on which it depends.

Related reference:

START and STOP commands on z/OS

Issuing commands to the z/OS console

You operate the broker using the z/OS START, STOP, and MODIFY commands.

mqsistart command

Use the **mqsistart** command to start the specified broker if all initial verification tests complete successfully.

mqsistop command

Use the **mqsistop** command to stop the specified component.

Starting a local broker using the IBM Integration Toolkit or IBM Integration Explorer

You can start your local brokers by using the IBM Integration Toolkit or the IBM Integration Explorer.

About this task

On Windows and Linux use the IBM Integration Toolkit or the IBM Integration Explorer to start and stop brokers. Alternatively, you can use the **mqsistart** and **mqsistop** commands from the command line to start or stop a broker.

Starting a local broker using the IBM Integration Toolkit: About this task

To start a local broker by using the IBM Integration Toolkit:

Procedure

1. Open the IBM Integration Toolkit, and switch to the Integration Development perspective.
2. In the Integration Nodes view, right-click the broker, and click **Start**.

Results

You have started the selected broker.

Starting a local broker using the IBM Integration Explorer: About this task

To start a local broker by using the IBM Integration Explorer:

Procedure

1. Open the IBM Integration Explorer.
2. In the Navigator view, expand the Integration Nodes folder.
3. Right-click the broker you want to start, and click **Start**.

Results

You have started the selected broker.

Related concepts:



IBM Integration Explorer

The IBM Integration Explorer is a graphical user interface based on the Eclipse platform for administering your integration nodes (brokers).

Related tasks:



Creating an integration node using the IBM Integration Explorer

On Linux on x86 or Windows, you can create integration nodes (brokers) by using the IBM Integration Explorer.

“Connecting to a local broker” on page 3

To administer a local broker by using the IBM Integration Toolkit or IBM Integration Explorer, you must connect to the broker.

“Starting and stopping a broker” on page 19

Run the appropriate command to start or stop a broker.

“Creating an integration server using the IBM Integration Toolkit or IBM Integration Explorer” on page 36

Use the IBM Integration Toolkit or IBM Integration Explorer to create integration servers on your broker.

“Stopping a local broker using the IBM Integration Toolkit or IBM Integration Explorer”

You can stop your local brokers by using the IBM Integration Toolkit or IBM Integration Explorer.

Chapter 1, “Administering brokers and broker resources,” on page 1

Administering brokers and associated integration node resources includes the tasks that you perform frequently to activate and manage those resources. Choose the method you prefer to administer your brokers and associated resources.

Related reference:



mqsistart command

Use the **mqsistart** command to start the specified broker if all initial verification tests complete successfully.



mqsistop command

Use the **mqsistop** command to stop the specified component.

Stopping a local broker using the IBM Integration Toolkit or IBM Integration Explorer

You can stop your local brokers by using the IBM Integration Toolkit or IBM Integration Explorer.

About this task

On Windows and Linux on x86, use the IBM Integration Toolkit or the IBM Integration Explorer to start and stop brokers. Alternatively, you can use the **mqsistop** command from the command line to stop a broker.

Stopping a local broker using the IBM Integration Toolkit:

About this task

To stop a local broker by using the IBM Integration Toolkit:

Procedure

1. Open the IBM Integration Toolkit, and switch to the Integration Development perspective.
2. In the Integration Nodes view, right-click the broker, and click **Stop**.

Results

You have stopped the selected broker.

Stopping a local broker using the IBM Integration Explorer:

About this task

To stop a local broker by using the IBM Integration Explorer:

Procedure

1. Open the IBM Integration Explorer.
2. In the Navigator view, expand the Integration Nodes folder.

3. Right-click the broker you want to stop, and click **Stop > Broker**. You can also click **Stop > Broker Immediately** if you have already tried, and failed, to stop the broker in a controlled fashion.

Results

You have stopped the selected broker.

Related concepts:



IBM Integration Explorer

The IBM Integration Explorer is a graphical user interface based on the Eclipse platform for administering your integration nodes (brokers).

Related tasks:



Creating an integration node using the IBM Integration Explorer

On Linux on x86 or Windows, you can create integration nodes (brokers) by using the IBM Integration Explorer.

“Connecting to a local broker” on page 3

To administer a local broker by using the IBM Integration Toolkit or IBM Integration Explorer, you must connect to the broker.

“Starting and stopping a broker” on page 19

Run the appropriate command to start or stop a broker.

“Creating an integration server using the IBM Integration Toolkit or IBM Integration Explorer” on page 36

Use the IBM Integration Toolkit or IBM Integration Explorer to create integration servers on your broker.

“Starting a local broker using the IBM Integration Toolkit or IBM Integration Explorer” on page 24

You can start your local brokers by using the IBM Integration Toolkit or the IBM Integration Explorer.

Chapter 1, “Administering brokers and broker resources,” on page 1

Administering brokers and associated integration node resources includes the tasks that you perform frequently to activate and manage those resources. Choose the method you prefer to administer your brokers and associated resources.

Viewing broker properties

You can view broker properties by using the `mqsireportbroker` command. You can also use the IBM Integration Explorer to view broker properties.

Before you begin

Before you start:

You must have completed the following tasks:

- Ensure that your user ID has the correct authorizations to perform the task; see Security requirements for administrative tasks.
- Create a broker.
- On Windows, Linux, and UNIX systems, you must set up your command-line environment before performing this task by running the product profile or console; see Setting up a command environment.

About this task

Use the **mqsireportbroker** command or domain to view all the properties associated with a broker. The command shows both parameters entered on the command line and viewable in the workbench.

Procedure

1. Run the **mqsireportbroker** command. For example, to view the properties of the broker SOAPBR, run the following command:
mqsireportbroker SOAPBR
2. View responses of the **mqsireportbroker** command to check on current settings. Examples are given in **mqsireportbroker** command.
3. If you want to make any changes, run the **mqsichangebroker** command, specifying the required parameters.

Related concepts:



The IBM Integration Bus environment

An integration node (broker) is a set of execution processes that hosts one or more message flows to route, transform, and enrich in flight messages.

Related tasks:



Creating a broker

You can create brokers on every platform that is supported by IBM Integration Bus. The broker runs as a 64-bit application on all platforms except Linux on x86 and Windows on x86.

“Starting and stopping a broker” on page 19

Run the appropriate command to start or stop a broker.



Configuring integration nodes using the IBM Integration Explorer

Configure your local and remote integration nodes (brokers) by using the IBM Integration Explorer.

Related reference:



Security requirements for administrative tasks

You can configure access permissions to govern which users and groups can manipulate objects in the broker network. Security requirements for administrative tasks depend on the platform that you use.



mqsichangebroker command

Use the **mqsichangebroker** command to change one or more of the configuration parameters of the broker.

Starting a WebSphere MQ queue manager as a Windows service

On Windows, you can start a WebSphere MQ queue manager as a Windows service to ensure the queue manager starts when you start your other components.

Before you begin

Before you start:

You must complete the following task:

- Stop the queue manager for the Integration Bus component by using the **endmqm** command. If you prefer, you can use the WebSphere MQ Explorer.

About this task

When you start a broker, the **mqsistart** command starts the associated queue manager if it is not already running.

When you start a broker on Windows, the broker starts as a service on Windows, but the associated queue manager does not.

You can change the properties of the queue manager service to set the startup type to automatic to enable the queue manager to run as a Windows service.

This change ensures that the operation of the queue manager is independent of the logged-on status of the user that starts the broker.

To start a WebSphere MQ queue manager as a Windows service:

Procedure

1. Click **Start > Programs > IBM WebSphere MQ > WebSphere MQ Explorer**.
2. In the left pane, right-click the queue manager and select **Properties**. The Properties dialog opens. The General properties are displayed.
3. Find the Startup property and set it to *Automatic*.
4. Click **OK**. The Properties dialog closes and the change is applied.
5. Restart the queue manager for the broker by using the **strmqm** command or WebSphere MQ Explorer. The changes to the queue manager's startup type take effect when you restart Windows.
6. Start the broker by using the **mqsistart** command.

Related tasks:

“Starting and stopping a broker” on page 19

Run the appropriate command to start or stop a broker.

Related reference:



mqsistart command

Use the **mqsistart** command to start the specified broker if all initial verification tests complete successfully.

Stopping a WebSphere MQ queue manager when you stop a broker

If you are preparing to stop a broker, you can stop the broker's WebSphere MQ queue manager at the same time.

About this task

You can specify a **-q** parameter on the **mqsistop** command to initiate a controlled shutdown of the queue manager for a broker.

Procedure

To stop a WebSphere MQ queue manager enter the following command on the command line:

```
mqsistop IB9NODE -q
```

where:

IB9NODE is the name of the broker.

-q stops the WebSphere MQ queue manager associated with the component. The command cannot complete until shutdown of the queue manager has completed.

Related tasks:

“Starting and stopping a broker” on page 19

Run the appropriate command to start or stop a broker.

Related reference:



mqsistop command

Use the **mqsistop** command to stop the specified component.

Deleting a broker

Delete a broker using the command line on the system where the Integration Bus component is installed.

Before you begin

Before you start: Check that your user ID has the correct authorizations to perform the task; for details, see Security requirements for administrative tasks.

About this task

On Windows and Linux systems, you can delete a broker using the IBM Integration Toolkit or the IBM Integration Explorer. For instructions on deleting the broker using the IBM Integration Toolkit or the IBM Integration Explorer, see “Deleting a broker using the IBM Integration Toolkit or IBM Integration Explorer” on page 33.

On Windows, Linux, and UNIX systems, you must set up your command-line environment before deleting a broker, by running the product profile or console; see Setting up a command environment.

Follow the link for the appropriate platform:

Procedure

- “Deleting a broker on Linux and UNIX systems” on page 30
- “Deleting a broker on Windows” on page 31
- “Deleting a broker on z/OS” on page 32

Related concepts:



The IBM Integration Bus environment

An integration node (broker) is a set of execution processes that hosts one or more message flows to route, transform, and enrich in flight messages.

Related tasks:



Creating a broker

You can create brokers on every platform that is supported by IBM Integration Bus. The broker runs as a 64-bit application on all platforms except Linux on x86 and Windows on x86.



Modifying a broker

Modify a broker by using the command line on the system where the Integration

Bus component is installed.

Related reference:



Security requirements for administrative tasks

You can configure access permissions to govern which users and groups can manipulate objects in the broker network. Security requirements for administrative tasks depend on the platform that you use.



mqsdeletebroker command

Use the **mqsdeletebroker** command to delete a named broker. The command also deletes the queues on the associated queue manager (created when the broker was created). You can also specify that the queue manager is to be deleted.

Deleting a broker on Linux and UNIX systems

Delete the physical broker component.

About this task

To delete a broker on Linux and UNIX systems:

Procedure

1. Stop the broker by using the **mqsistop** command.
2. Enter the following command to delete the broker:
mqsdeletebroker IB9NODE
where IB9NODE is the broker name.
3. If you have created integration server profiles for this broker, delete these profiles if they are no longer required.
4. Any empty integration server shared-classes directories (under *workpath/config/<my_broker_name>/<my_eg_label>*) are automatically deleted. Otherwise, if no longer required, it is left to the user to manually delete the directories and contents.
5. If the broker level shared-classes directory (under *workpath/config/<my_broker_name>*) is empty, it is automatically deleted. Otherwise, if no longer required, it is left to the user to manually delete the directory and contents.

Results

On completion of this task, you have:

- Removed the broker data from the database.
- Removed the record for the component in the broker registry. It is therefore removed from the list of components that are displayed when you run the **mqsilist** command.
- Removed any integration server profile scripts that are no longer needed.
- Removed any empty integration server shared-classes directories.
- Removed any empty broker shared-classes directory.

Related concepts:



The IBM Integration Bus environment

An integration node (broker) is a set of execution processes that hosts one or more message flows to route, transform, and enrich in flight messages.

Related tasks:

“Starting and stopping a broker” on page 19
Run the appropriate command to start or stop a broker.

Related reference:

 **mqsdeletebroker** command

Use the **mqsdeletebroker** command to delete a named broker. The command also deletes the queues on the associated queue manager (created when the broker was created). You can also specify that the queue manager is to be deleted.

 **mqsistop** command

Use the **mqsistop** command to stop the specified component.

 **mqsilist** command

Use the **mqsilist** command to list installed brokers and their associated resources.

Deleting a broker on Windows

Delete the physical broker component.

About this task

 To delete a broker:

Procedure

1. Stop the broker by using the **mqsistop** command.
2. Enter the following command to delete the broker:
mqsdeletebroker IB9NODE
where IB9NODE is the broker name.
3. If you have created integration server profiles for this broker, delete these profiles if they are no longer required.
4. Any empty integration server shared-classes directories (under *workpath/config/<my_broker_name>/<my_eg_label>*) are automatically deleted. Otherwise, if no longer required, it is left to the user to manually delete the directories and contents.
5. If the broker level shared-classes directory (under *workpath/config/<my_broker_name>*) is empty, it is automatically deleted. Otherwise, if no longer required, it is left to the user to manually delete the directory and contents.

Results

On completion of this task, you have:

- Stopped the Windows service that runs the broker.
- Removed the record for the component in the broker registry. It is therefore removed from the list of components that are displayed when you run the **mqsilist** command.
- Removed any integration server profile scripts that are no longer needed.
- Removed any empty integration server shared-classes directories.
- Removed any empty broker shared-classes directory.

Related concepts:

 The IBM Integration Bus environment

An integration node (broker) is a set of execution processes that hosts one or more message flows to route, transform, and enrich in flight messages.

Related tasks:

“Starting and stopping a broker” on page 19
Run the appropriate command to start or stop a broker.

Related reference:

mqsdeletebroker command

Use the **mqsdeletebroker** command to delete a named broker. The command also deletes the queues on the associated queue manager (created when the broker was created). You can also specify that the queue manager is to be deleted.

mqsistop command

Use the **mqsistop** command to stop the specified component.

mqsilist command

Use the **mqsilist** command to list installed brokers and their associated resources.

Deleting a broker on z/OS

Delete the physical broker component.

About this task

To delete a broker:

Procedure

1. Stop the broker, by stopping the started task.
2. Customize and submit the delete job BIPDLBK in your component PDSE to delete the broker component and WebSphere MQ. This action does not delete all files from the component directory in the file system.
3. Any empty integration server shared-classes directories (under *workpath/config/<my_broker_name>/<my_eg_label>*) are automatically deleted. Otherwise, if no longer required, it is left to the user to manually delete the directories and contents.
4. If the broker level shared-classes directory (under *workpath/config/<my_broker_name>*) is empty, it is automatically deleted. Otherwise, if no longer required, it is left to the user to manually delete the directory and contents.

Related concepts:

The IBM Integration Bus environment

An integration node (broker) is a set of execution processes that hosts one or more message flows to route, transform, and enrich in flight messages.

Related tasks:

Creating a broker on z/OS

Create the broker component and the other resources on which it depends.

Modifying a broker on z/OS

Use the **mqschangebroker** command on z/OS to modify your broker.

Related reference:

mqsdeletebroker command

Use the **mqsdeletebroker** command to delete a named broker. The command also deletes the queues on the associated queue manager (created when the broker was created). You can also specify that the queue manager is to be deleted.

`mqsicreatebroker` command

Use the `mqsicreatebroker` command to create a broker and its associated resources.

Deleting a broker using the IBM Integration Toolkit or IBM Integration Explorer

On Windows and Linux on x86 you can delete brokers by using the IBM Integration Toolkit or IBM Integration Explorer.

Before you begin

Before you start: Check that your user ID has the correct authorizations to perform the task; for details, see Security requirements for administrative tasks.

About this task

You can delete a local broker using the IBM Integration Toolkit or IBM Integration Explorer. You cannot delete a remote broker using the IBM Integration Toolkit or IBM Integration Explorer, but you can remove the connection to the remote broker from your workspace. To remove the connection to a remote broker in the IBM Integration Toolkit, right-click the broker in the Integration Nodes view, and click **Remove Connection**. To remove the connection to a remote broker in the IBM Integration Explorer, right-click the broker in the Navigator view, and click **Remove**.

Deleting a broker using the IBM Integration Toolkit:

About this task

To delete a broker using the IBM Integration Toolkit:

Procedure

1. Open the IBM Integration Toolkit, and switch to the Integration Development perspective.
2. In the Integration Nodes view, right-click the broker, and click **Delete**. If the broker is a remote broker, the connection to the broker is removed from your workspace.

If the broker is a local broker, the Delete Local Broker wizard is displayed:

- a. Optional: If you want to delete the queue manager associated with the broker, ensure **Delete the broker's queue manager** is selected.
- b. Click **Finish** to delete the broker. The wizard stops the broker if it is running.

Results

You have deleted the broker.

Deleting a broker using the IBM Integration Explorer:

About this task

To delete a broker using the IBM Integration Explorer:

Procedure

1. Open the IBM Integration Explorer.

2. In the Navigator view, expand the broker's queue manager, or the Integration Nodes folder.
3. Right-click the broker, and click **Delete > Broker**. The Delete Broker wizard is displayed.
4. If you want to delete the queue manager associated with the broker, ensure **Delete the broker's queue manager** is selected.
5. Click **Next**. The wizard stops the broker if it is running.
6. Click **Finish** to close the wizard.

Results

You have deleted the broker. If you choose to delete the queue manager, the queue manager continues to be displayed in the Navigator view until the views are refreshed. The time this takes depends on your queue manager refresh settings, but the default refresh value is 15 seconds.

Related concepts:



IBM Integration Explorer

The IBM Integration Explorer is a graphical user interface based on the Eclipse platform for administering your integration nodes (brokers).

Related tasks:

“Deleting a broker” on page 29

Delete a broker using the command line on the system where the Integration Bus component is installed.



Creating a broker

You can create brokers on every platform that is supported by IBM Integration Bus. The broker runs as a 64-bit application on all platforms except Linux on x86 and Windows on x86.



Modifying a broker

Modify a broker by using the command line on the system where the Integration Bus component is installed.

Related reference:



Security requirements for administrative tasks

You can configure access permissions to govern which users and groups can manipulate objects in the broker network. Security requirements for administrative tasks depend on the platform that you use.



`mqsideletebroker` command

Use the `mqsideletebroker` command to delete a named broker. The command also deletes the queues on the associated queue manager (created when the broker was created). You can also specify that the queue manager is to be deleted.

Managing integration servers

Work with your existing integration servers to manage the message flows that you have deployed to them by using the IBM Integration Toolkit or IBM Integration Explorer. You can use the command line, and the IBM Integration API (also known as the CMP) to complete some of these actions.

About this task

- “Creating an integration server”
- “Renaming an integration server using the IBM Integration Explorer” on page 40
- “Starting an integration server using the IBM Integration Toolkit or IBM Integration Explorer” on page 41
- “Stopping an integration server using the IBM Integration Toolkit or IBM Integration Explorer” on page 43
- “Deleting an integration server” on page 45

Related concepts:



The IBM Integration Bus environment

An integration node (broker) is a set of execution processes that hosts one or more message flows to route, transform, and enrich in flight messages.

Related tasks:



Configuring brokers for development environments

Set up application development environments on Linux on x86 or Windows to create, test, and deploy message flows and associated resources.



Configuring brokers for test and production environments

Create one or more brokers on one or more computers, and configure them on your test and production systems to process messages that contain your business data.

“Managing brokers” on page 2

Work with your existing brokers to manage their connections and their active status by using the IBM Integration Toolkit or IBM Integration Explorer. You can use the IBM Integration API (also known as the CMP) to complete some of these actions.

“Managing message flows” on page 50

Work with your existing messages that you have deployed to integration servers by using the IBM Integration Toolkit or IBM Integration Explorer. You can use the IBM Integration API (also known as the CMP) to complete some of these actions.

Related information:



IBM Integration API (CMP)

Creating an integration server

Create integration servers by using IBM Integration Toolkit, the IBM Integration Explorer, or the command line.

Before you begin

Before you start:

Complete the following tasks:

- Create a broker
- Connect to the broker:
 - “Connecting to a local broker” on page 3
 - “Connecting to a remote broker” on page 4
 - “Connecting to a remote integration node on z/OS” on page 6

About this task

The mode that your broker is working in can affect the number of integration servers that you can use; see Restrictions that apply in each operation mode.

Use one of the following methods to complete this task:

- The IBM Integration Toolkit or the IBM Integration Explorer. See “Creating an integration server using the IBM Integration Toolkit or IBM Integration Explorer” for more information.
- The `mqsicreateexecutiongroup` command. Details are provided in “Creating an integration server using the `mqsicreateexecutiongroup` command” on page 38.

You can also use the CMP to create integration servers on all platforms; see “Developing applications that use the IBM Integration API” on page 58.

For information about why you might want to create multiple integration servers, see Integration servers.

Related concepts:



The IBM Integration Bus environment

An integration node (broker) is a set of execution processes that hosts one or more message flows to route, transform, and enrich in flight messages.

Related tasks:



Configuring brokers for development environments

Set up application development environments on Linux on x86 or Windows to create, test, and deploy message flows and associated resources.



Configuring brokers for test and production environments

Create one or more brokers on one or more computers, and configure them on your test and production systems to process messages that contain your business data.

“Managing brokers” on page 2

Work with your existing brokers to manage their connections and their active status by using the IBM Integration Toolkit or IBM Integration Explorer. You can use the IBM Integration API (also known as the CMP) to complete some of these actions.

“Managing message flows” on page 50

Work with your existing messages that you have deployed to integration servers by using the IBM Integration Toolkit or IBM Integration Explorer. You can use the IBM Integration API (also known as the CMP) to complete some of these actions.

Related reference:



`mqsicreateexecutiongroup` command

Use the `mqsicreateexecutiongroup` command to add a new integration server to a broker.

Related information:



IBM Integration API (CMP)

Creating an integration server using the IBM Integration Toolkit or IBM Integration Explorer

Use the IBM Integration Toolkit or IBM Integration Explorer to create integration servers on your broker.

Before you begin

Before you start: Check that you have completed the prerequisite tasks, and see what other options you can use to create integration servers, in “Creating an integration server” on page 35.

About this task

You must create an integration server before you can deploy message flows and related resources to a broker.

Creating an integration server using the IBM Integration Toolkit: Before you begin

Before you start:

- Create a broker
- Optional: “Connecting to a remote broker” on page 4

About this task

To add an integration server to a broker:

Procedure

1. In the Integration Nodes view, right-click the broker to which you want to add an integration server, and click **New Integration Server**.
2. In the New Integration Server dialog, enter the **Integration Server name**.
3. Click **OK** to add the integration server to the broker.

Results

The integration server is added to the appropriate broker, and can be viewed as a child of the selected broker.

Creating an integration server using the IBM Integration Explorer: Before you begin

Before you start:

- Create a broker
- Connect to a local broker or “Connecting to a remote broker” on page 4

About this task

To add an integration server to a broker:

Procedure

1. In the Navigator view, expand the Integration Nodes folder.
2. Right-click the broker to which you want to add an integration server, and click **New > Integration Server**.
3. In the New Integration Server dialog, enter the **Integration Server name**.
4. Click **OK** to add the integration server to the broker.

Results

The integration server is added to the appropriate broker, and can be viewed as a child of the selected broker.

Note: If you do not immediately see the integration server that you added, you can refresh the broker by right-clicking on it and selecting **Refresh**.

Related concepts:



IBM Integration Explorer

The IBM Integration Explorer is a graphical user interface based on the Eclipse platform for administering your integration nodes (brokers).

Related tasks:



Creating an integration node using the IBM Integration Explorer

On Linux on x86 or Windows, you can create integration nodes (brokers) by using the IBM Integration Explorer.

“Renaming an integration server using the IBM Integration Explorer” on page 40
You can rename an integration server by using the IBM Integration Explorer.



Importing a broker archive file to the IBM Integration Explorer

Before you can deploy resources to your integration nodes (brokers) by using the IBM Integration Explorer, you must first import a broker archive file into a Broker Archive Folder.



Deploying a broker archive file

After you have created and populated a broker archive (BAR) file, deploy the file to an integration server on a broker, so that the file contents can be used in the broker.

“Connecting to a local broker” on page 3

To administer a local broker by using the IBM Integration Toolkit or IBM Integration Explorer, you must connect to the broker.

“Connecting to a remote broker” on page 4

To administer a remote broker by using the IBM Integration Toolkit or IBM Integration Explorer, you must connect to the broker.

“Deleting an integration server by using the IBM Integration Toolkit or IBM Integration Explorer” on page 46

You can delete an integration server by using the IBM Integration Toolkit or IBM Integration Explorer.

Creating an integration server using the `mqsicreateexecutiongroup` command

Use the `mqsicreateexecutiongroup` command to create integration servers on your broker.

Before you begin

Before you start: Check that you have completed the prerequisite tasks, and see what other options you can use to create integration servers, in “Creating an integration server” on page 35.

You must create an integration server before you can deploy message flows and related resources to a broker.

Procedure

1. If you are creating an integration server on Linux, UNIX, and Windows systems:
 - a. Open a command prompt that has the environment configured for your current installation.
 - b. Enter the **mqsicreateexecutiongroup** command, specifying the parameters for the integration server that you want to create.
 - If the broker is local, specify the broker name. For example:
mqsicreateexecutiongroup IB9NODE -e EGroup_2
 - If the broker is remote, you can specify a configuration file. For example:
mqsicreateexecutiongroup -n IB9NODE.broker -e EGroup_2
 - If the broker is remote, you can alternatively specify one or more of the connection parameters **-i**, **-p**, **-q**. For example:
mqsicreateexecutiongroup -q IB9QMGR -e EGroup_2See the **mqsicreateexecutiongroup** command description for more details about these options.
2. If you are creating an integration server on z/OS:
 - a. Configure the BIPCREG job to specify the properties for the integration server that you want to create.
 - b. Run the BIPCREG job.

Results

On completion of this task, the integration server has been created on the specified broker.

What to do next

Next: You can deploy message flows to the integration server by using the IBM Integration Toolkit or the IBM Integration Explorer.

Related concepts:



Integration servers

An integration server is a named grouping of message flows that have been assigned to a broker. The broker enforces a degree of isolation between message flows in distinct integration servers by ensuring that they run in separate address spaces, or as unique processes.

Related tasks:

“Creating an integration server using the IBM Integration Toolkit or IBM Integration Explorer” on page 36

Use the IBM Integration Toolkit or IBM Integration Explorer to create integration servers on your broker.

“Renaming an integration server using the IBM Integration Explorer” on page 40
You can rename an integration server by using the IBM Integration Explorer.

“Deleting an integration server by using the IBM Integration Toolkit or IBM Integration Explorer” on page 46

You can delete an integration server by using the IBM Integration Toolkit or IBM Integration Explorer.

“Deleting an integration server by using the **mqsdeleteexecutiongroup** command” on page 48

Use the command line to delete an integration server from the broker.

Related reference:



mqsicreateexecutiongroup command

Use the **mqsicreateexecutiongroup** command to add a new integration server to a broker.



mqsdeleteexecutiongroup command

Use the **mqsdeleteexecutiongroup** command to remove an integration server from a broker.

Renaming an integration server using the IBM Integration Explorer

You can rename an integration server by using the IBM Integration Explorer.

Before you begin

Before you start:

- Create a broker
- Start the broker
- Connect to a local broker, connect to a remote broker, or connect to a remote broker on z/OS
- Create an integration server

About this task

To rename an integration server:

Procedure

1. Open the IBM Integration Explorer.
2. In the Navigator view, expand the Integration Nodes folder.
3. Expand your integration server.
4. Right-click the integration server that you want to rename, and click **Rename**.
5. Enter a name for the integration server, and click **OK**.

Results

You have renamed the selected integration server.

Related concepts:



IBM Integration Explorer

The IBM Integration Explorer is a graphical user interface based on the Eclipse platform for administering your integration nodes (brokers).

Related tasks:



Creating an integration node using the IBM Integration Explorer

On Linux on x86 or Windows, you can create integration nodes (brokers) by using the IBM Integration Explorer.

“Starting a local broker using the IBM Integration Toolkit or IBM Integration Explorer” on page 24

You can start your local brokers by using the IBM Integration Toolkit or the IBM Integration Explorer.

“Connecting to a local broker” on page 3

To administer a local broker by using the IBM Integration Toolkit or IBM Integration Explorer, you must connect to the broker.

“Creating an integration server using the IBM Integration Toolkit or IBM Integration Explorer” on page 36

Use the IBM Integration Toolkit or IBM Integration Explorer to create integration servers on your broker.

Chapter 1, “Administering brokers and broker resources,” on page 1

Administering brokers and associated integration node resources includes the tasks that you perform frequently to activate and manage those resources. Choose the method you prefer to administer your brokers and associated resources.

Starting an integration server using the IBM Integration Toolkit or IBM Integration Explorer

You can start an integration server and all the deployed message flows in an integration server by using the IBM Integration Toolkit or IBM Integration Explorer.

About this task

If resources are deployed to your integration server, the started or stopped state of the message flows is retained when you next start the integration server. You can use the menu to start or stop all message flows after the integration server has been started.

If you prefer, you can start integration servers by using the `mqsistartmsgflow` command.

Starting an integration server with the IBM Integration Toolkit About this task

To start an integration server by using the IBM Integration Toolkit:

Procedure

1. Open the IBM Integration Toolkit, and switch to the Integration Development perspective.
2. In the Integration Nodes view, expand your broker.
3. Right-click the integration server, and click **Start**.

Results

A message is sent to the broker to start the selected integration server. Right-click the message flows that you want to start, and click **Start** to start the message flows deployed to the integration server.

Starting an integration server with the IBM Integration Explorer About this task

To start an integration server by using the IBM Integration Explorer:

Procedure

1. Open the IBM Integration Explorer.
2. In the Navigator view, expand the Integration Nodes folder.

3. Expand your broker.
4. Right-click the integration server, and click **Start > Integration Server**.

Results

A message is sent to the broker to start the selected integration server. Click **Start > All Flows** to start all the message flows deployed to the integration server.

Starting an integration server with the `mqsistartmsgflow` command

About this task

To start an integration server by using the `mqsistartmsgflow` command:

Procedure

1. Open a command prompt that has the environment configured for your current installation.
2. Enter the `mqsistartmsgflow` command, specifying the parameters for the integration server that you want to start.
 - If the broker is local, specify the broker name. For example:
`mqsistartmsgflow IB9NODE -e EGroup_2`
 - If the broker is remote, you can specify a configuration file. For example:
`mqsistartmsgflow -n IB9NODE.broker -e EGroup_2`
 - If the broker is remote, you can alternatively specify one or more of the connection parameters `-i`, `-p`, `-q`. For example:
`mqsistartmsgflow -q IB9QMGR -e EGroup_2`

What to do next

If you want to start all the integration servers on a broker, specify the `-g` flag instead of `-e`.

See the `mqsistartmsgflow` command description for more details about these options.

Related concepts:



IBM Integration Explorer

The IBM Integration Explorer is a graphical user interface based on the Eclipse platform for administering your integration nodes (brokers).

Related tasks:



Creating an integration node using the IBM Integration Explorer

On Linux on x86 or Windows, you can create integration nodes (brokers) by using the IBM Integration Explorer.

“Starting a local broker using the IBM Integration Toolkit or IBM Integration Explorer” on page 24

You can start your local brokers by using the IBM Integration Toolkit or the IBM Integration Explorer.

“Connecting to a local broker” on page 3

To administer a local broker by using the IBM Integration Toolkit or IBM Integration Explorer, you must connect to the broker.

“Creating an integration server using the IBM Integration Toolkit or IBM Integration Explorer” on page 36

Use the IBM Integration Toolkit or IBM Integration Explorer to create integration servers on your broker.

“Starting a message flow by using the IBM Integration Toolkit or IBM Integration Explorer” on page 53

Use the IBM Integration Toolkit or IBM Integration Explorer to start a message flow.

“Stopping a message flow using the IBM Integration Toolkit or IBM Integration Explorer” on page 54

Use the IBM Integration Toolkit or IBM Integration Explorer to stop a message flow.

Chapter 1, “Administering brokers and broker resources,” on page 1

Administering brokers and associated integration node resources includes the tasks that you perform frequently to activate and manage those resources. Choose the method you prefer to administer your brokers and associated resources.

Related reference:



mqsistartmsgflow command

Use the **mqsistartmsgflow** command to start integration servers, applications, and message flows.

Stopping an integration server using the IBM Integration Toolkit or IBM Integration Explorer

You can stop all the message flows in an integration server by using the IBM Integration Toolkit or IBM Integration Explorer.

About this task

If resources are deployed to your integration server, the started or stopped state of the message flows is remembered when you stop the integration server. You can use the menu to start or stop all message flows before the integration server has been stopped.

If you prefer, you can stop integration servers by using the **mqsistopmsgflow** command.

The broker processes all inflight messages and associated transactions for each message flow before stopping it. See Message flow transactions for information on how outstanding units of work are handled in this situation.

Stopping an integration server with the IBM Integration Toolkit

About this task

To stop an integration server by using the IBM Integration Toolkit:

Procedure

1. Open the IBM Integration Toolkit, and switch to the Integration Development perspective.
2. In the Integration Nodes view, expand your broker.
3. Right-click the integration server, and click **Stop**.

Results

A message is sent to the broker to stop the selected integration server.

Stopping an integration server with the IBM Integration Explorer About this task

To stop an integration server by using the IBM Integration Explorer:

Procedure

1. Open the IBM Integration Explorer.
2. In the Navigator view, expand the Integration Nodes folder.
3. Expand your broker.
4. Right-click the integration server, and click **Stop > Integration Server**.

Results

A message is sent to the broker to stop the selected integration server.

Stopping an integration server with the `mqsistopmsgflow` command About this task

To stop an integration server by using the `mqsistopmsgflow` command:

Procedure

1. Open a command prompt that has the environment configured for your current installation.
2. Enter the `mqsistopmsgflow` command, specifying the parameters for the integration server that you want to stop.
 - If the broker is local, specify the broker name. For example:

```
mqsistopmsgflow IB9NODE -e EGroup_2
```
 - If the broker is remote, you can specify a configuration file. For example:

```
mqsistopmsgflow -n IB9NODE.broker -e EGroup_2
```
 - If the broker is remote, you can alternatively specify one or more of the connection parameters `-i`, `-p`, `-q`. For example:

```
mqsistopmsgflow -q IB9QMGR -e EGroup_2
```

What to do next

If you want to stop all the integration servers on a broker, specify the `-g` flag instead of `-e`.

See the `mqsistopmsgflow` command description for more details about these options.

Related concepts:



IBM Integration Explorer

The IBM Integration Explorer is a graphical user interface based on the Eclipse platform for administering your integration nodes (brokers).

Related tasks:



Creating an integration node using the IBM Integration Explorer

On Linux on x86 or Windows, you can create integration nodes (brokers) by using the IBM Integration Explorer.

“Starting a local broker using the IBM Integration Toolkit or IBM Integration Explorer” on page 24

You can start your local brokers by using the IBM Integration Toolkit or the IBM Integration Explorer.

“Connecting to a local broker” on page 3

To administer a local broker by using the IBM Integration Toolkit or IBM Integration Explorer, you must connect to the broker.

“Creating an integration server using the IBM Integration Toolkit or IBM Integration Explorer” on page 36

Use the IBM Integration Toolkit or IBM Integration Explorer to create integration servers on your broker.

“Starting an integration server using the IBM Integration Toolkit or IBM Integration Explorer” on page 41

You can start an integration server and all the deployed message flows in an integration server by using the IBM Integration Toolkit or IBM Integration Explorer.

Chapter 1, “Administering brokers and broker resources,” on page 1

Administering brokers and associated integration node resources includes the tasks that you perform frequently to activate and manage those resources. Choose the method you prefer to administer your brokers and associated resources.

Related reference:



mqsistopmsgflow command

Use the **mqsistopmsgflow** command to stop integration servers, applications, and message flows.

Deleting an integration server

Delete integration servers by using IBM Integration Toolkit, the IBM Integration Explorer, or the command line.

Before you begin

Before you start:

Complete the following tasks:

- Create an integration server
- Check that the integration server is running; you cannot delete an integration server that is stopped.

About this task

Use one of the following methods to complete this task:

- The IBM Integration Toolkit or the IBM Integration Explorer. See “Deleting an integration server by using the IBM Integration Toolkit or IBM Integration Explorer” on page 46 for more information.
- The **mqsdeleteexecutiongroup** command. Details are provided in “Deleting an integration server by using the **mqsdeleteexecutiongroup** command” on page 48.

You can also use the CMP to delete integration servers on all platforms; see “Developing applications that use the IBM Integration API” on page 58.

For information about why you might want to create multiple integration servers, see Integration servers.

Related concepts:



The IBM Integration Bus environment

An integration node (broker) is a set of execution processes that hosts one or more message flows to route, transform, and enrich in flight messages.

Related tasks:



Configuring brokers for development environments

Set up application development environments on Linux on x86 or Windows to create, test, and deploy message flows and associated resources.



Configuring brokers for test and production environments

Create one or more brokers on one or more computers, and configure them on your test and production systems to process messages that contain your business data.

“Managing brokers” on page 2

Work with your existing brokers to manage their connections and their active status by using the IBM Integration Toolkit or IBM Integration Explorer. You can use the IBM Integration API (also known as the CMP) to complete some of these actions.

“Managing message flows” on page 50

Work with your existing messages that you have deployed to integration servers by using the IBM Integration Toolkit or IBM Integration Explorer. You can use the IBM Integration API (also known as the CMP) to complete some of these actions.

Related reference:



`mqsdeleteexecutiongroup` command

Use the `mqsdeleteexecutiongroup` command to remove an integration server from a broker.

Related information:



IBM Integration API (CMP)

Deleting an integration server by using the IBM Integration Toolkit or IBM Integration Explorer

You can delete an integration server by using the IBM Integration Toolkit or IBM Integration Explorer.

Before you begin

Before you start: Check that you have completed the prerequisite tasks, and see what other options you can use to delete integration servers, in “Deleting an integration server” on page 45.

Deleting an integration server by using the IBM Integration Toolkit: About this task

To delete an integration server by using the IBM Integration Toolkit

Procedure

1. Open the IBM Integration Toolkit, and switch to the Integration Development perspective.
2. In the Integration Nodes view, expand your broker.
3. Right-click the integration server that you want to delete, and click **Delete > Integration Server**.

Results

Your integration server and all resources deployed to it are deleted.

- If you have created any integration server profiles for this integration server, delete them manually if they are no longer required.
- Any empty integration server shared-classes directory (under *workpath/config/<my_broker_name>/<my_eg_label>*) is automatically deleted. Otherwise, if no longer required, it is left to the user to manually delete the directory and contents.

Deleting an integration server by using the IBM Integration Explorer: About this task

To delete an integration server by using the IBM Integration Explorer:

Procedure

1. Open the IBM Integration Explorer.
2. In the Navigator view, expand the Integration Nodes folder.
3. Expand your broker.
4. Right-click the integration server, and click **Delete > Integration Server**.
5. If you have deployed resources, you must confirm that you want to remove all deployed resources before the integration server is deleted. Click **OK** to remove the deployed resources.

Results

Your integration server and all resources deployed to it are deleted.

- If you have created any integration server profiles for this integration server, delete them manually if they are no longer required.
- Any empty integration server shared-classes directory (under *workpath/config/<my_broker_name>/<my_eg_label>*) is automatically deleted. Otherwise, if no longer required, it is left to the user to manually delete the directory and contents.

Related concepts:



IBM Integration Explorer

The IBM Integration Explorer is a graphical user interface based on the Eclipse platform for administering your integration nodes (brokers).

Related tasks:



Creating an integration node using the IBM Integration Explorer

On Linux on x86 or Windows, you can create integration nodes (brokers) by using the IBM Integration Explorer.

“Starting a local broker using the IBM Integration Toolkit or IBM Integration Explorer” on page 24

You can start your local brokers by using the IBM Integration Toolkit or the IBM Integration Explorer.

“Connecting to a local broker” on page 3

To administer a local broker by using the IBM Integration Toolkit or IBM Integration Explorer, you must connect to the broker.

“Creating an integration server using the IBM Integration Toolkit or IBM Integration Explorer” on page 36

Use the IBM Integration Toolkit or IBM Integration Explorer to create integration servers on your broker.

“Deleting an integration server by using the **mqsideleteexecutiongroup** command”
Use the command line to delete an integration server from the broker.

“Starting a message flow by using the IBM Integration Toolkit or IBM Integration Explorer” on page 53

Use the IBM Integration Toolkit or IBM Integration Explorer to start a message flow.

“Stopping a message flow using the IBM Integration Toolkit or IBM Integration Explorer” on page 54

Use the IBM Integration Toolkit or IBM Integration Explorer to stop a message flow.

Chapter 1, “Administering brokers and broker resources,” on page 1

Administering brokers and associated integration node resources includes the tasks that you perform frequently to activate and manage those resources. Choose the method you prefer to administer your brokers and associated resources.

Deleting an integration server by using the **mqsideleteexecutiongroup** command

Use the command line to delete an integration server from the broker.

Before you begin

Before you start: Check that you have completed the prerequisite tasks, and see what other options you can use to delete integration servers, in “Deleting an integration server” on page 45.

Procedure

1. If you are deleting an integration server on Linux, UNIX, and Windows systems:
 - a. Open a command prompt that has the environment configured for your current installation.
 - b. Enter the **mqsideleteexecutiongroup** command, specifying the parameters for the integration server that you want to delete.
 - If the broker is local, specify the broker name; for example:
mqsideleteexecutiongroup IB9NODE -e EGroup_2
 - If the broker is remote, you can specify a configuration file; for example:
mqsideleteexecutiongroup -n IB9NODE.broker -e EGroup_2
 - If the broker is remote, you can alternatively specify one or more of the connection parameters **-i**, **-p**, and **-q**; for example:
mqsideleteexecutiongroup -q IB9QMGR -e EGroup_2

See the **mqsideleteexecutiongroup** command description for more details about these options.

- c. If you have created integration server profiles and they are no longer required, delete them manually.
 - d. Any empty integration server shared-classes directory (under *workpath/config/<my_broker_name>/<my_eg_label>*) is automatically deleted. Otherwise, if no longer required, it is left to the user to manually delete the directory and contents.
2. If you are deleting an integration server on z/OS:
 - a. Configure the BIPDLEG job to specify the properties for the integration server that you want to delete.
 - b. Run the BIPDLEG job.
 - c. Any empty integration server shared-classes directory (under *workpath/config/<my_broker_name>/<my_eg_label>*) is automatically deleted. Otherwise, if no longer required, it is left to the user to manually delete the directory and contents.

Results

On completion of this task, the integration server has been deleted from the specified broker. In addition:

- All message flows that were running on the integration server are stopped.
- Any integration server profile scripts that are no longer needed have been removed.
- Any empty integration server shared-classes directory has been removed.

Related concepts:



Integration servers

An integration server is a named grouping of message flows that have been assigned to a broker. The broker enforces a degree of isolation between message flows in distinct integration servers by ensuring that they run in separate address spaces, or as unique processes.

Related tasks:

“Creating an integration server using the **mqscreateexecutiongroup** command” on page 38

Use the **mqscreateexecutiongroup** command to create integration servers on your broker.

“Creating an integration server using the IBM Integration Toolkit or IBM Integration Explorer” on page 36

Use the IBM Integration Toolkit or IBM Integration Explorer to create integration servers on your broker.

“Deleting an integration server by using the IBM Integration Toolkit or IBM Integration Explorer” on page 46

You can delete an integration server by using the IBM Integration Toolkit or IBM Integration Explorer.

“Developing applications that use the IBM Integration API” on page 58

Develop Java applications that use the IBM Integration API (also known as the CMP) to communicate with, deploy to, and manage brokers and their associated resources.

Related reference:



mqscreateexecutiongroup command

Use the **mqscreateexecutiongroup** command to add a new integration server to a broker.

`mqsdeleteexecutiongroup` command

Use the `mqsdeleteexecutiongroup` command to remove an integration server from a broker.

Managing message flows

Work with your existing messages that you have deployed to integration servers by using the IBM Integration Toolkit or IBM Integration Explorer. You can use the IBM Integration API (also known as the CMP) to complete some of these actions.

About this task

- “Setting the start mode of flows and applications at run time”
- “Starting a message flow by using the IBM Integration Toolkit or IBM Integration Explorer” on page 53
- “Stopping a message flow using the IBM Integration Toolkit or IBM Integration Explorer” on page 54
- “Deleting a message flow using the IBM Integration Toolkit or IBM Integration Explorer” on page 55
- “Setting user-defined properties dynamically at run time using the IBM Integration Explorer” on page 56

Related concepts:

The IBM Integration Bus environment

An integration node (broker) is a set of execution processes that hosts one or more message flows to route, transform, and enrich in flight messages.

Related tasks:

Configuring brokers for development environments

Set up application development environments on Linux on x86 or Windows to create, test, and deploy message flows and associated resources.

Configuring brokers for test and production environments

Create one or more brokers on one or more computers, and configure them on your test and production systems to process messages that contain your business data.

“Managing brokers” on page 2

Work with your existing brokers to manage their connections and their active status by using the IBM Integration Toolkit or IBM Integration Explorer. You can use the IBM Integration API (also known as the CMP) to complete some of these actions.

“Managing integration servers” on page 34

Work with your existing integration servers to manage the message flows that you have deployed to them by using the IBM Integration Toolkit or IBM Integration Explorer. You can use the command line, and the IBM Integration API (also known as the CMP) to complete some of these actions.

Related information:

IBM Integration API (CMP)

Setting the start mode of flows and applications at run time

You can configure the run state of message flows and applications when you deploy or when you restart an integration server.

Before you begin

Before you start:

This topic assumes that a broker archive (BAR) file has been created and contains a message flow or application. For more information, see the following topics:

- Creating a broker archive (BAR) file
- Adding files to a broker archive

About this task

In previous versions of IBM Integration Bus, when you deploy a new message flow to an integration server, the flow is started automatically. If you deploy an existing message flow to an integration server, the existing run state of the flow is maintained. Therefore, if the flow was stopped before you deployed it, it remains stopped when you deploy it. This run state is also maintained when you restart the integration server.

From WebSphere Message Broker Version 8.0 onwards, you can set the default behavior of message flows and applications when you deploy or restart an integration server. For example, you might have a message flow that creates resources that are required by other message flows. Therefore, you might want to start one message flow before all others when the flows are deployed, or when the broker, integration server, or containing application are started. You can set one message flow to start automatically, then set other message flows to require a manual restart.

You can configure the run state by using the `startMode` property on the `mqsipplybaroverride` command. You can also specify the default behavior by setting the Start Mode property on the broker archive BAR file. For instructions, see [Configuring the start mode of flows and applications at development time](#). Use the `mqsireadbar` command to view the current values of configurable properties in the BAR file. You can set the `startMode` property to one of the following values:

Maintained

This value is the default, and indicates that the flow or application is started when deployment is complete, and remains running until a stop command is issued. After a stop command has been issued, the flow or application remains stopped until a start command is issued. The state of the flow or application remains unchanged after redeployment, or after the broker, integration server, or containing application has been restarted.

Manual

This value indicates that the flow or application must always be started manually after deployment or after the broker, integration server, or containing application has been restarted. The flow or application is in stopped state after deployment or redeployment, and after the broker, integration server, or containing application is restarted, even if the flow or application was running before the deployment or restart.

Automatic

This value indicates that the flow or application is always started automatically after deployment, redeployment, or after the broker, integration server, or containing application is restarted.

You can set this property for message flows and applications. The state of an application overrides the state of any message flows that it contains. For example, if an application is stopped, no flows in that application can run, even if they have been set to start automatically.

To find out how to use the **mqsiapplybaroverride** command to set the `startMode` property, see **mqsiapplybaroverride** command.

What to do next

Next: Deploy the BAR file by following the instructions in Deploying a broker archive file.

Related concepts:



Packaging and deployment overview

Deployment is the process of transferring data to an integration server on a broker so that it can take effect in the broker. Message flows and associated resources are packaged in broker archive (BAR) files for deployment.



Applications and libraries

Applications and libraries are deployable containers of resources, such as message flows, subflows, message definitions (DFDL, XSD files), JAR files, XSL style sheets, and WebSphere Adapters files.



Configurable properties of a broker archive

System objects that are defined in message flows can have properties that you can update in the broker archive (BAR) file before deployment.

Related tasks:



Deploying resources

Deploy message flow applications to integration servers by sending a broker archive (BAR) file to a broker, which unpacks and stores the contents ready for when your message flows are started.



Managing message flow resources

You can use applications, libraries, and integration projects to organize your resources.



Editing configurable properties

You can edit configurable properties in the deployment descriptor file (`broker.xml`) of your broker archive.

Related reference:



mqsiapplybaroverride command

Use the **mqsiapplybaroverride** command to replace configurable values in the broker archive (BAR) deployment descriptor with new values that you specify in a properties file.



mqsireadbar command

Use the **mqsireadbar** command to read a deployable BAR file and identify its defined keywords.



Configurable properties

Some properties of message flow nodes are configurable and can be changed by using the **mqsiapplybaroverride** command. The following table maps the message flow node properties to the corresponding properties of the **mqsiapplybaroverride**

command.

Starting a message flow by using the IBM Integration Toolkit or IBM Integration Explorer

Use the IBM Integration Toolkit or IBM Integration Explorer to start a message flow.

About this task

You can start an individual message flow in the IBM Integration Toolkit, or multiple message flows in the IBM Integration Explorer. If you stop the integration server in which the message flow is running, the started or stopped status of the message flow is retained. From WebSphere Message Broker Version 8.0 onwards, you can set the default behavior of message flows and applications when you deploy or restart an integration server.

If your message flow includes a subflow that is defined in a .subflow file, you can start your message flow only if the subflow is deployed to the same integration server.

Starting a message flow by using the IBM Integration Toolkit Procedure

1. In the Integration Nodes view, expand the broker and integration server.
2. Right-click the message flow that you want to start, and click **Start**.

Results

A message is sent to the broker to start the message flow.

Starting a message flow by using the IBM Integration Explorer Procedure

1. In the Navigator view, expand the Integration Nodes folder.
2. Expand the broker and integration server, and right-click the message flow or message flows that you want to start, and click **Start**.

Results

A message is sent to the broker to start the message flows.

Note: A message flow can also be started using the command line utility `mqsistartmsgflow`. For more information see `mqsistartmsgflow` command.

Related concepts:



IBM Integration Explorer

The IBM Integration Explorer is a graphical user interface based on the Eclipse platform for administering your integration nodes (brokers).

Related tasks:

“Connecting to a local broker” on page 3

To administer a local broker by using the IBM Integration Toolkit or IBM Integration Explorer, you must connect to the broker.

“Stopping a message flow using the IBM Integration Toolkit or IBM Integration Explorer” on page 54

Use the IBM Integration Toolkit or IBM Integration Explorer to stop a message

flow.

“Setting the start mode of flows and applications at run time” on page 50
You can configure the run state of message flows and applications when you deploy or when you restart an integration server.

Chapter 1, “Administering brokers and broker resources,” on page 1
Administering brokers and associated integration node resources includes the tasks that you perform frequently to activate and manage those resources. Choose the method you prefer to administer your brokers and associated resources.

Related reference:



mqsistartmsgflow command

Use the **mqsistartmsgflow** command to start integration servers, applications, and message flows.

Stopping a message flow using the IBM Integration Toolkit or IBM Integration Explorer

Use the IBM Integration Toolkit or IBM Integration Explorer to stop a message flow.

About this task

You can stop an individual message flow in the IBM Integration Toolkit, or multiple message flows in the IBM Integration Explorer. If you stop the integration server in which the message flow is running, the started or stopped status of the message flow is remembered.

You can also use the **mqsistopmsgflow** command to stop a message flow.

Stopping a message flow using the IBM Integration Toolkit Procedure

1. In the Integration Nodes view, expand the broker and integration server.
2. Right-click the message flow that you want to stop, and click **Stop**.

Results

A message is sent to the broker to stop the selected message flow.

Stopping a message flow using the IBM Integration Explorer Procedure

1. In the Navigator view, expand the Integration Nodes folder.
2. Expand the broker and integration server, and right-click the message flow or message flows that you want to stop, and click **Stop**.

Results

A message is sent to the broker to stop the selected message flows.

Related concepts:



IBM Integration Explorer

The IBM Integration Explorer is a graphical user interface based on the Eclipse platform for administering your integration nodes (brokers).

Related tasks:

“Connecting to a local broker” on page 3

To administer a local broker by using the IBM Integration Toolkit or IBM Integration Explorer, you must connect to the broker.

“Starting a message flow by using the IBM Integration Toolkit or IBM Integration Explorer” on page 53

Use the IBM Integration Toolkit or IBM Integration Explorer to start a message flow.

Chapter 1, “Administering brokers and broker resources,” on page 1

Administering brokers and associated integration node resources includes the tasks that you perform frequently to activate and manage those resources. Choose the method you prefer to administer your brokers and associated resources.

“Setting the start mode of flows and applications at run time” on page 50

You can configure the run state of message flows and applications when you deploy or when you restart an integration server.

Related reference:



mqsistopmsgflow command

Use the **mqsistopmsgflow** command to stop integration servers, applications, and message flows.

Deleting a message flow using the IBM Integration Toolkit or IBM Integration Explorer

Use the IBM Integration Toolkit or IBM Integration Explorer to remove message flows from an integration server.

About this task

You can delete an individual message flow in the IBM Integration Toolkit, or multiple message flows in the IBM Integration Explorer.

Deleting a message flow using the IBM Integration Toolkit Procedure

1. In the Integration Nodes view, expand the broker and integration server.
2. Right-click the message flow that you want to delete, and click **Delete**.

Results

A message is sent to the broker to delete the selected message flow.

Deleting a message flow using the IBM Integration Explorer Procedure

1. In the Navigator view, expand the Integration Nodes folder.
2. Expand the broker and integration server, and right-click the message flow or message flows that you want to delete, and click **Delete**.

Results

A message is sent to the broker to delete the selected message flows.

Related concepts:



IBM Integration Explorer

The IBM Integration Explorer is a graphical user interface based on the Eclipse platform for administering your integration nodes (brokers).

Related tasks:



Creating an integration node using the IBM Integration Explorer

On Linux on x86 or Windows, you can create integration nodes (brokers) by using the IBM Integration Explorer.

“Starting a local broker using the IBM Integration Toolkit or IBM Integration Explorer” on page 24

You can start your local brokers by using the IBM Integration Toolkit or the IBM Integration Explorer.

“Connecting to a local broker” on page 3

To administer a local broker by using the IBM Integration Toolkit or IBM Integration Explorer, you must connect to the broker.

“Creating an integration server using the IBM Integration Toolkit or IBM Integration Explorer” on page 36

Use the IBM Integration Toolkit or IBM Integration Explorer to create integration servers on your broker.

“Starting a message flow by using the IBM Integration Toolkit or IBM Integration Explorer” on page 53

Use the IBM Integration Toolkit or IBM Integration Explorer to start a message flow.

“Stopping a message flow using the IBM Integration Toolkit or IBM Integration Explorer” on page 54

Use the IBM Integration Toolkit or IBM Integration Explorer to stop a message flow.

Chapter 1, “Administering brokers and broker resources,” on page 1

Administering brokers and associated integration node resources includes the tasks that you perform frequently to activate and manage those resources. Choose the method you prefer to administer your brokers and associated resources.

Setting user-defined properties dynamically at run time using the IBM Integration Explorer

Use the IBM Integration Explorer to view and set user-defined properties on a message flow dynamically at run time.

Before you begin

For user-defined properties on a message flow to be discoverable, the message flow must comply with the following conditions:

- The message flow must contain at least one of the following nodes:
 - JavaCompute
 - Compute
 - Database
 - Filter
 - PHPCompute
- The message flow must define the relevant user-defined property and provide an override value.

About this task

Tip: Use meaningful names and values for the properties that you define, so that you can understand their purpose and intent quickly. For example, a user-defined property named `property01`, with an initial value of `valueA` is not as useful as a property named `RouteToAorB` with an initial value of `RouteA`.

Use the following instructions to view and modify user-defined properties on your message flows using the IBM Integration Explorer:

Procedure

1. In the Navigator view, expand the Integration Nodes folder, and navigate to the message flow on which you want view the user-defined properties.
2. Right-click the message flow, and click **Properties**.
3. Click **User Defined Properties** to display a list of the user-defined properties and their value defined on the message flow.
4. Optional: if you want to modify the user-defined properties, change the value in the appropriate rows in the Value column, and click **Apply**. A message is sent to the integration node (broker) to change the value of the user-defined property.
5. Click **OK** to close the Properties window.

Related concepts:

IBM Integration Explorer

The IBM Integration Explorer is a graphical user interface based on the Eclipse platform for administering your integration nodes (brokers).


User-defined properties


A user-defined property (UDP) is a property that is defined when you construct a message flow by using the Message Flow editor. This property can be used by the ESQL or Java program inside message flow nodes, such as a Compute node.

User-defined properties in ESQL

Access user-defined properties (UDPs) as variables in your ESQL program by specifying the `EXTERNAL` keyword on a `DECLARE` statement. For example, the ESQL statement `DECLARE today EXTERNAL CHARACTER 'monday'` defines a user-defined property called `today` with an initial value `monday`.

Related tasks:

 **Configuring a message flow at deployment time with user-defined properties**
Use user-defined properties (UDPs) to configure message flows at deployment and run time, without modifying program code. You can give a UDP an initial value when you declare it in your program, or when you use the Message Flow editor to create or modify a message flow.

 **Accessing message flow user-defined properties from a JavaCompute node**
Customize a JavaCompute node to access properties that you have associated with the message flow in which the node is included.

Related reference:

ESQL variables

ESQL variables can be described as *external variables*, *normal variables*, or *shared variables*; their use is defined in the `DECLARE` statement.

Configurable message flow properties

When you add a message flow to a broker archive (BAR) file in preparation for deploying it to a broker, you can set additional properties that influence its run time operation. These properties are available for review and update when you select the **Manage and Configure** tab for the broker archive file.

Developing applications that use the IBM Integration API

Develop Java applications that use the IBM Integration API (also known as the CMP) to communicate with, deploy to, and manage brokers and their associated resources.

Before you begin

Before you start:

Read an overview of the The IBM Integration API, and how you can use it to manage the resources associated with your brokers.

Samples are provided to demonstrate typical CMP scenarios. Run and explore the samples to learn about what you can do with the CMP; see “The IBM Integration API samples” on page 60.

About this task

You can use the IBM Integration API to develop administrative applications or to develop message flows.

To write administrative applications that run tasks in your brokers, see the following information:

- Configuring the environment
- Connecting to a broker
- Navigating brokers and broker resources
- Deploying resources
- Setting message flow user-defined properties at run time
- Working with UserDefined configurable service properties in JavaCompute nodes
- Managing brokers
- Managing brokers from JavaCompute nodes
- Working with resource manager statistics
- Working with activity logs
- Working with data capture
- Submitting batch requests

Results

When you have created your applications, test their operation in your test environment. Check the results of the operations that the programs have performed by using the Deployment Log view.

What to do next

When you have written and tested your applications, distribute them to the computers from which you want your administrators to perform the tasks that you have developed.

Related concepts:



The IBM Integration API

The IBM Integration API is a programming interface that your applications can use to control brokers and their resources through a remote interface.

“The IBM Integration API samples” on page 60

Explore the samples to learn the basic features that are provided by the IBM Integration API (also known as the CMP). Run the samples to deploy a BAR file or manage a broker, or use the IBM Integration API (CMP) Exerciser to implement various tasks.

Related tasks:

“Configuring an environment for developing and running CMP applications” on page 72

Prepare the environment in which you want to run your CMP applications.



Packaging and deploying

Package resources that you create in the IBM Integration Toolkit, such as message flows, and deploy them to integration servers on brokers.

Related information:



IBM Integration API (CMP)

The IBM Integration API

The IBM Integration API is a programming interface that your applications can use to control brokers and their resources through a remote interface.

The IBM Integration API is also known as the Configuration Manager Proxy, or CMP

The Configuration Manager has been removed from Version 7.0, and the full name of the API has changed. However, the terms CMP application and CMP have been retained, and are used in this information center to refer to the IBM Integration API, for continuity and consistency with the JAR file `ConfigManagerProxy.jar` that supplies all the classes.

The IBM Integration API (CMP) consists solely of a Java implementation, and is sometimes referred to as the IBM Integration Bus Java API. Your applications have complete access to the broker functions and resources through the set of Java classes that constitute the IBM Integration API. Use the IBM Integration API to interact with the broker to perform the following tasks:

- Deploy BAR files
- Change the broker configuration properties
- Create, modify, and delete integration servers
- Inquire and set the status of the broker and its associated resources, and to be informed if status changes
 - Integration servers
 - Deployed message flows
 - Deployed files used by the message flows (for example, JAR files)
- View the Administration log
- View the Activity log
- Create and modify message flow applications.

Related concepts:



The IBM Integration Bus environment

An integration node (broker) is a set of execution processes that hosts one or more message flows to route, transform, and enrich in flight messages.

Integration servers

An integration server is a named grouping of message flows that have been assigned to a broker. The broker enforces a degree of isolation between message flows in distinct integration servers by ensuring that they run in separate address spaces, or as unique processes.

IBM Integration Toolkit

The IBM Integration Toolkit is an integrated development environment and graphical user interface based on the Eclipse platform.

IBM Integration Explorer

The IBM Integration Explorer is a graphical user interface based on the Eclipse platform for administering your integration nodes (brokers).

“The IBM Integration API samples”

Explore the samples to learn the basic features that are provided by the IBM Integration API (also known as the CMP). Run the samples to deploy a BAR file or manage a broker, or use the IBM Integration API (CMP) Exerciser to implement various tasks.

Related tasks:

“Developing applications that use the IBM Integration API” on page 58
Develop Java applications that use the IBM Integration API (also known as the CMP) to communicate with, deploy to, and manage brokers and their associated resources.

Related information:

IBM Integration API (CMP)

The IBM Integration API samples

Explore the samples to learn the basic features that are provided by the IBM Integration API (also known as the CMP). Run the samples to deploy a BAR file or manage a broker, or use the IBM Integration API (CMP) Exerciser to implement various tasks.

Deploy BAR

The Deploy BAR sample deploys a BAR file to an integration server, and displays the outcome. Read about this sample in “Running the CMP Deploy BAR sample” on page 61.

Broker management

The broker management sample uses the CMP to display to the screen the complete run state of the broker. “Running the CMP broker management sample” on page 63 describes this sample in more detail.

The IBM Integration API (CMP) Exerciser

The IBM Integration API (CMP) Exerciser is a graphical interface to the CMP that you can use to view and manipulate brokers. Use the IBM Integration API (CMP) Exerciser sample to view and manage a broker, or record and play back configuration scripts. You can also customize the IBM Integration API (CMP) Exerciser to tailor the tasks to suit your requirements. See “Running the IBM Integration API (CMP) Exerciser sample” on page 64 for more information.

Modify CMP samples

You can modify the CMP samples, and change various parameters that affect how the samples run. Read “Modifying the CMP samples” on page 71 and follow the guidance given about possible changes.

Related concepts:



The IBM Integration API

The IBM Integration API is a programming interface that your applications can use to control brokers and their resources through a remote interface.

Related tasks:

“Running the CMP Deploy BAR sample”

Run the CMP Deploy BAR sample to deploy a broker archive file to a broker.

“Running the CMP broker management sample” on page 63

Run the broker management sample to display the complete run state of a broker.

“Running the IBM Integration API (CMP) Exerciser sample” on page 64

Run the IBM Integration API (CMP) Exerciser sample to view and manage a broker, customize the IBM Integration API (CMP) Exerciser, or record and play back configuration scripts.

“Modifying the CMP samples” on page 71

Modify the CMP samples to change the parameters that they use to complete their tasks.

Related information:




IBM Integration API (CMP)

Running the CMP Deploy BAR sample

Run the CMP Deploy BAR sample to deploy a broker archive file to a broker.

Before you begin

Before you begin

- You must set up a default configuration.
-  On Windows, the BAR file must be in the same directory as the CMP .bat sample files.

The BAR file and integration server name, and the connection details that define the target broker, are hard coded into the application. You can run this sample unchanged, or you can modify the values and parameters that it uses to apply them to your own configuration.

If you modify the Deploy BAR sample, you must update and recompile the source file before you run it. The source file for this sample is located in the following directory:

```
install_dir/sample/ConfigManagerProxy/cmp/DeployBar.java
```

About this task

Use the Deploy BAR sample to deploy a BAR file to an integration server, and display the outcome.

Procedure

1. Run the Deploy BAR sample by entering the appropriate command for your platform:

- **Windows** On Windows, open a Command Console and run the following command:

```
install_dir\sample\ConfigManagerProxy\StartDeployBAR.bat
```

- **Linux** **UNIX** **z/OS** On other platforms:
 - a. Start a broker command environment by running **mqsipofile**, or follow the guidance provided in the StartDeployBar shell script to configure the correct CLASSPATH for your environment.
 - b. Run the shell script:

```
install_dir\sample\ConfigManagerProxy\StartDeployBAR
```

The default connection parameters used by the sample are shown in the following table.

Connection parameter	Description
"localhost"	Host name of the computer where the broker is running.
"IB9QMGR"	Name of the broker queue manager.
2414	Port on which the broker queue manager is listening
"default"	Name of the integration server.
"mybar.bar"	Fully qualified name of the BAR file to deploy.

The CMP connects to the broker that is running on the local computer (defined by localhost). The queue manager IB9QMGR must be listening on port 2414. Next, the CMP deploys the file mybar.bar to the predefined integration server default.

2. Check the results of the sample by viewing the broker in the IBM Integration Toolkit, or by using the IBM Integration API (CMP) Exerciser.

What to do next

Next: Run another sample, or work with the IBM Integration API (CMP) Exerciser.

Related concepts:



The IBM Integration API

The IBM Integration API is a programming interface that your applications can use to control brokers and their resources through a remote interface.

“The IBM Integration API samples” on page 60

Explore the samples to learn the basic features that are provided by the IBM Integration API (also known as the CMP). Run the samples to deploy a BAR file or manage a broker, or use the IBM Integration API (CMP) Exerciser to implement various tasks.

Related tasks:



Packaging and deploying

Package resources that you create in the IBM Integration Toolkit, such as message flows, and deploy them to integration servers on brokers.

“Running the CMP broker management sample” on page 63

Run the broker management sample to display the complete run state of a broker.

“Running the IBM Integration API (CMP) Exerciser sample” on page 64

Run the IBM Integration API (CMP) Exerciser sample to view and manage a broker, customize the IBM Integration API (CMP) Exerciser, or record and play

back configuration scripts.

“Modifying the CMP samples” on page 71

Modify the CMP samples to change the parameters that they use to complete their tasks.

Related information:



IBM Integration API (CMP)

Running the CMP broker management sample

Run the broker management sample to display the complete run state of a broker.

Before you begin

Before you start:

You must create a connection file that describes the connection details for the integration node (broker) against which the sample runs. For information about how to create the connection file, see “Exporting integration node connection details from the IBM Integration Explorer” on page 10.

You can run this sample unchanged, or you can modify the values and parameters that it uses to apply them to your own configuration.

If you modify this management sample, you must update and recompile the source file before you run it. The source file for this sample is located in the following directory:

```
install_dir/sample/ConfigManagerProxy/cmp/BrokerInfo.java
```

About this task

Use the broker management sample to display the complete run state of the target broker.

Procedure

- Run the broker management sample by entering the appropriate command for your platform. Replace *connection_file* with the fully qualified file path to the file that describes the connection details for the broker (with extension *.broker*).

- **Windows** On Windows, use the Command Console to run the following command:

```
install_dir\sample\ConfigManagerProxy\StartBrokerInfo.bat connection_file
```

- **Linux** **UNIX** **z/OS** On other platforms:

1. Start a broker command environment by running **mqsiprofile**, or follow the guidance provided in the **StartBrokerInfo** shell script to configure the correct CLASSPATH for your environment.
2. Run the shell script:

```
install_dir\sample\ConfigManagerProxy\StartBrokerInfo connection_file
```

The complete run state of the broker is displayed. You can see responses like the following output:

```
(28/01/09 11:47:43) Connecting. Please wait...  
(28/01/09 11:47:46) Successfully connected to the broker's queue manager.  
(28/01/09 11:47:49) Successfully connected to the broker.
```

```
(28/01/09 11:47:49) Broker 'IB9NODE' is running.
(28/01/09 11:47:50) Integration server 'default' on 'IB9NODE' is running.
(28/01/09 11:47:51) Message flow 'simpleflow' on 'default' on 'IB9NODE' is running.
(28/01/09 11:47:51) Disconnected.
```

- If you prefer, you can run this sample in interactive mode, which causes the sample to listen for changes to the broker.

1. To run the sample in interactive mode, specify the `-i` option on the command.

For example:

```
\sample\ConfigManagerProxy\StartBrokerInfo.bat
c:\myBroker.broker -i
```

In interactive mode, you see the output shown earlier, and the additional response:

```
(13/08/08 15:53:46) Listening for changes to the broker...
```

2. To stop the sample when it is running in interactive mode, force it to end by using CTRL+C.

What to do next

Next: Run another sample, or work with the IBM Integration API (CMP) Exerciser.

Related concepts:



The IBM Integration API

The IBM Integration API is a programming interface that your applications can use to control brokers and their resources through a remote interface.

“The IBM Integration API samples” on page 60

Explore the samples to learn the basic features that are provided by the IBM Integration API (also known as the CMP). Run the samples to deploy a BAR file or manage a broker, or use the IBM Integration API (CMP) Exerciser to implement various tasks.

Related tasks:

“Running the CMP Deploy BAR sample” on page 61

Run the CMP Deploy BAR sample to deploy a broker archive file to a broker.

“Running the IBM Integration API (CMP) Exerciser sample”

Run the IBM Integration API (CMP) Exerciser sample to view and manage a broker, customize the IBM Integration API (CMP) Exerciser, or record and play back configuration scripts.

“Modifying the CMP samples” on page 71

Modify the CMP samples to change the parameters that they use to complete their tasks.

Related information:



IBM Integration API (CMP)

Running the IBM Integration API (CMP) Exerciser sample

Run the IBM Integration API (CMP) Exerciser sample to view and manage a broker, customize the IBM Integration API (CMP) Exerciser, or record and play back configuration scripts.

Before you begin

Before you start:

You can run this sample unchanged, or you can modify the values and parameters that it uses to apply them to your own configuration.

If you modify the IBM Integration API (CMP) Exerciser sample, you must update and recompile the source file before you run it. The source file for this sample is located in the following directory:

```
install_dir/sample/ConfigManagerProxy/cmp/exerciser
```

About this task

Use the IBM Integration API (CMP) Exerciser to complete the following tasks:

- “Viewing and managing a broker in the IBM Integration API (CMP) Exerciser”
- “Customizing the IBM Integration API (CMP) Exerciser” on page 67
- “Recording and playing back configuration scripts using the IBM Integration API (CMP) Exerciser” on page 69

Related concepts:



The IBM Integration API

The IBM Integration API is a programming interface that your applications can use to control brokers and their resources through a remote interface.

“The IBM Integration API samples” on page 60

Explore the samples to learn the basic features that are provided by the IBM Integration API (also known as the CMP). Run the samples to deploy a BAR file or manage a broker, or use the IBM Integration API (CMP) Exerciser to implement various tasks.

Related tasks:

“Modifying the CMP samples” on page 71

Modify the CMP samples to change the parameters that they use to complete their tasks.

Related information:



IBM Integration API (CMP)

Viewing and managing a broker in the IBM Integration API (CMP) Exerciser:

Use the IBM Integration API (CMP) Exerciser sample to view and manage a broker.

About this task

To view and manage a broker by using the IBM Integration API (CMP) Exerciser, complete the following steps:

Procedure

1. Start the IBM Integration API (CMP) Exerciser:
 - **Windows** On Windows, click **Start > All Programs > IBM Integration Bus 9.0.0.0 > IBM Integration Java APIs > IBM Integration CMP API Exerciser**.
 - **Linux** **UNIX** **z/OS** On other platforms:
 - a. Start a broker command environment by running **mqsiprofile**, or follow the guidance provided in the StartConfigManagerProxyExerciser shell script to configure the correct CLASSPATH for your environment.

b. Ensure that your user ID has writer permission to the current directory. The CMP API Exerciser stores its configuration settings in a file in this directory.

c. Run the shell script:

```
install_dir\sample\ConfigManagerProxy\StartConfigManagerProxyExerciser
```

The IBM Integration API (CMP) Exerciser window opens.

2. Connect to a running broker by clicking either **File > Connect to Local Broker** or **File > Connect to Remote Broker**.

The Connect to a Broker dialog opens.

3. Enter the connection parameters to the broker, then click **Submit**.

Broker information is retrieved and displayed in the IBM Integration API (CMP) Exerciser window. You have now connected to the broker.

The upper left of the screen contains a hierarchical representation of the broker to which you are connected. Selecting objects in the tree causes the table on the right to change, reflecting the attributes of the object that you select. The Method column lists CMP methods that you can call in your own Java applications, and the Result column indicates the data that is returned by calling the CMP method on the selected object.

4. Run an CMP method against a broker object. CMP methods are used to manage objects in a broker.

a. In the navigation tree, right-click a broker.

A pop-up menu opens to show all the available CMP methods.

b. Select **Create integration server**. The Create integration server dialog opens.

c. Enter the name for a new integration server and click **Submit**. The output from the method is displayed in the log window at the bottom of the screen. For example:

```
(31/03/09 16:53:50) ----> cmp.exerciser.ClassTesterForBrokerProxy.testCreateEG
(IB9NODE, "eg1")
(31/03/09 16:53:50) The request was successfully sent to the broker.
(31/03/09 16:53:50) <---- cmp.exerciser.ClassTesterForBrokerProxy.testCreateEG
```

You also see messages that are returned from the broker when the method is called. For example:

```
(31/03/09 16:53:50) ----> cmp.exerciser.ExerciserAdministeredObjectListener.processActionResponse(...)
(31/03/09 16:53:50) affectedObject = IB9NODE
(31/03/09 16:53:50) completionCode = success
(31/03/09 16:53:50) (Reference property) commsmessage.lastinbatch=true
(31/03/09 16:53:50) (Reference property) uuid=595e1d10-3875-11d4-a485-000629be5bf8
(31/03/09 16:53:50) (Reference property) child.uuid=1d8b3c5d-2001-0000-0080-c2000502e620
(31/03/09 16:53:50) (Reference property) configmanagerproxy.osname=Windows
(31/03/09 16:53:50) (Reference property) child.name=eg1
(31/03/09 16:53:50) (Reference property) userid=Matt
(31/03/09 16:53:50) (Reference property) configmanagerproxy.hostname=lucas
(31/03/09 16:53:50) (Reference property) commsmessage.configobjecttype=Broker
(31/03/09 16:53:50) (Reference property) type=Broker
(31/03/09 16:53:50) (Reference property) child.type=ExecutionGroup
(31/03/09 16:53:50) (Reference property) commsmessage.operationtype=createchild
(31/03/09 16:53:50) (Reference property) configmanagerproxy.noeventlog=false
(31/03/09 16:53:50) (Reference property) eg.arch=0
(31/03/09 16:53:50) <---- cmp.exerciser.ExerciserAdministeredObjectListener.processActionResponse()
```

In this example, completionCode = success means that the request to create the integration server is successful. The lines marked (Reference property) describe the request to which the response refers.

Results

During these steps you connected to a broker, viewed the broker information, and performed a management task by using the IBM Integration API (CMP) Exerciser.

What to do next

Next: Continue to work with the IBM Integration API (CMP) Exerciser, or run another sample.

Related concepts:



The IBM Integration API

The IBM Integration API is a programming interface that your applications can use to control brokers and their resources through a remote interface.

“The IBM Integration API samples” on page 60

Explore the samples to learn the basic features that are provided by the IBM Integration API (also known as the CMP). Run the samples to deploy a BAR file or manage a broker, or use the IBM Integration API (CMP) Exerciser to implement various tasks.

Related tasks:

“Running the IBM Integration API (CMP) Exerciser sample” on page 64

Run the IBM Integration API (CMP) Exerciser sample to view and manage a broker, customize the IBM Integration API (CMP) Exerciser, or record and play back configuration scripts.

“Customizing the IBM Integration API (CMP) Exerciser”

Enable or disable a selection of options to customize the IBM Integration API (CMP) Exerciser to meet your requirements.

“Recording and playing back configuration scripts using the IBM Integration API (CMP) Exerciser” on page 69

Use the IBM Integration API (CMP) Exerciser to record and play back configuration scripts.

“Modifying the CMP samples” on page 71

Modify the CMP samples to change the parameters that they use to complete their tasks.

Related information:



IBM Integration API (CMP)


Customizing the IBM Integration API (CMP) Exerciser:

Enable or disable a selection of options to customize the IBM Integration API (CMP) Exerciser to meet your requirements.

About this task

To customize the IBM Integration API (CMP) Exerciser, complete the following steps:

Procedure

1. Start the IBM Integration API (CMP) Exerciser.
 -  On Windows, click **Start** > **All Programs** > **IBM Integration Bus 9.0.0.0** > **IBM Integration Java APIs** > **Integration Java API Exerciser**.

- **Linux** **UNIX** **z/OS** On other operating systems, run the following shell script:

```
install_dir\sample\ConfigManagerProxy\StartConfigManagerProxyExerciser
```

The IBM Integration API (CMP) Exerciser window opens.

2. Customize the IBM Integration API (CMP) Exerciser by selecting one or more of the following options from the **File** and **View** menus.
 - a. Optional: Click **File** > **Use Incremental Deployment** to enable or disable this option.
 - If you enable this option, all deploy operations cause a delta (incremental) deploy where relevant.
 - If you disable this option, all deploy operations cause a complete deploy.
 - b. Optional: Click **File** > **Connect to Remote Brokers Using .broker Properties File** to enable or disable this option.
 - If you enable this option, a file dialog opens when you connect to a broker. Use the file dialog to navigate to a file with a .broker extension, which provides the connection parameters to the queue manager that hosts the broker.
 - If you disable this option, you are prompted to enter the queue manager connection parameters, security exit parameters, host name, and port.
 - c. Optional: Click **File** > **Render Console in HTML** to enable or disable this option.
 - If you enable this option, the console window is rendered in HTML.
 - If you disable this option, the console window is rendered in plain text.
 - d. Optional: Click **File** > **Enable MQ Java Client Service Trace** to enable or disable this option.
 - If you enable this option, a level 5 service trace of the WebSphere MQ Classes for Java runs. A trace dialog opens; specify a file name to which trace records are written.
 - If you disable this option, level 5 service tracing of the WebSphere MQ Classes for Java is not done.
 - e. Optional: Click **File** > **Enable API Service Trace** to enable or disable this option.
 - If you enable this option, a service trace of the CMP run. A trace dialog opens; specify a file name to which trace records are written.
 - If you disable this option, service tracing of the CMP is not done.
 - f. Optional: Click **File** > **Set Timeout Characteristics**.
Specify the time, in seconds, that the IBM Integration API (CMP) Exerciser waits for responses from the broker. The default wait interval is 6 seconds.
 - g. Optional: Click **View** and then click **Advanced**, **Basic**, or **Everything**.
 - If you select **Advanced**, output from all available methods is displayed in the IBM Integration API (CMP) Exerciser.
 - If you select **Basic**, output from a subset of the available methods is displayed in the IBM Integration API (CMP) Exerciser.
 - If you select **Everything**, output from all available methods is displayed in the IBM Integration API (CMP) Exerciser and additional information about AdministeredObjects is included, if it is available. For example, for MessageFlowProxy objects, the MessageFlowProxy.Node child objects are displayed.
 - h. Optional: Click **View** > **BAR Files in Tree**.

- If you enable this option, the deployed resources that are shown in the tree are grouped by the BAR file name under which they were last deployed.
 - If you disable this option, the BAR file names are not displayed.
- i. Optional: Click **View > Show Resource Managers** to enable or disable this option.
- If you enable this option, the resource managers are shown in the tree for each integration server.
 - If you disable this option, the resource managers are not shown.

What to do next

Next: Continue to work with the IBM Integration API (CMP) Exerciser, or run another sample.

Related concepts:



The IBM Integration API

The IBM Integration API is a programming interface that your applications can use to control brokers and their resources through a remote interface.

“The IBM Integration API samples” on page 60

Explore the samples to learn the basic features that are provided by the IBM Integration API (also known as the CMP). Run the samples to deploy a BAR file or manage a broker, or use the IBM Integration API (CMP) Exerciser to implement various tasks.

Related tasks:

“Modifying the CMP samples” on page 71

Modify the CMP samples to change the parameters that they use to complete their tasks.

“Running the IBM Integration API (CMP) Exerciser sample” on page 64

Run the IBM Integration API (CMP) Exerciser sample to view and manage a broker, customize the IBM Integration API (CMP) Exerciser, or record and play back configuration scripts.

“Viewing and managing a broker in the IBM Integration API (CMP) Exerciser” on page 65

Use the IBM Integration API (CMP) Exerciser sample to view and manage a broker.

“Recording and playing back configuration scripts using the IBM Integration API (CMP) Exerciser”

Use the IBM Integration API (CMP) Exerciser to record and play back configuration scripts.

Related information:



IBM Integration API (CMP)

Recording and playing back configuration scripts using the IBM Integration API (CMP) Exerciser:

Use the IBM Integration API (CMP) Exerciser to record and play back configuration scripts.

About this task

You can run a script file from the command line, a shell window, or from a batch file. If you run the script by using one of these methods, ensure the first action completed by the script is to connect to a broker.

Procedure

1. Start the IBM Integration API (CMP) Exerciser.
 - **Windows** On Windows, click **Start > All Programs > IBM Integration Bus 9.0.0.0 > IBM Integration Java APIs > IBM Integration CMP API Exerciser**.
 - **Linux** **UNIX** **z/OS** On other operating systems, run the following shell script:

```
install_dir\sample\ConfigManagerProxy\StartConfigManagerProxyExerciser
```
- The IBM Integration API (CMP) Exerciser window opens.
2. Start recording a script by clicking **Scripting > Record New Script**. The Save dialog opens. Type a name for the script file, select an appropriate file location, and click **Save**.
 - a. Complete one or more actions on a broker by using the IBM Integration API (CMP) Exerciser.

If you start recording before you run actions against the broker, the first action taken is to connect to the broker; however, you can start recording a script at any point during the management of a broker.
 - b. Optional: Insert a pause by clicking **Scripting > Insert a pause**.

The IBM Integration API (CMP) Exerciser pauses so that responses can be returned before the next action is issued. Use this option to avoid naming conflicts if you are deleting and recreating objects of the same name.

The Insert a pause dialog opens; specify the duration of the pause in seconds.
 - c. Stop recording the script by clicking **Scripting > Stop Recording**.

Information about the actions taken are saved to the script file.
 3. To replay the script file:
 - a. Click **Scripting > Play Back Recorded Script**.

The Open dialog opens.
 - b. Select the appropriate script file and click **Open**. The script file is replayed.

What to do next

Next: Continue to work with the IBM Integration API (CMP) Exerciser, or run another sample.

Related concepts:



The IBM Integration API

The IBM Integration API is a programming interface that your applications can use to control brokers and their resources through a remote interface.

“The IBM Integration API samples” on page 60

Explore the samples to learn the basic features that are provided by the IBM Integration API (also known as the CMP). Run the samples to deploy a BAR file or manage a broker, or use the IBM Integration API (CMP) Exerciser to implement various tasks.

Related tasks:

“Running the IBM Integration API (CMP) Exerciser sample” on page 64
Run the IBM Integration API (CMP) Exerciser sample to view and manage a broker, customize the IBM Integration API (CMP) Exerciser, or record and play back configuration scripts.

“Viewing and managing a broker in the IBM Integration API (CMP) Exerciser” on page 65

Use the IBM Integration API (CMP) Exerciser sample to view and manage a broker.

“Customizing the IBM Integration API (CMP) Exerciser” on page 67

Enable or disable a selection of options to customize the IBM Integration API (CMP) Exerciser to meet your requirements.

“Modifying the CMP samples”

Modify the CMP samples to change the parameters that they use to complete their tasks.

Related information:



IBM Integration API (CMP)

Modifying the CMP samples

Modify the CMP samples to change the parameters that they use to complete their tasks.

About this task

If you change a sample, you must recompile the source code to implement those changes. For example, you might choose to change the default connection parameters. If you modify these and other values, you change the behavior of the sample.

To modify a sample, perform the following steps:

Procedure

1. Set up the environment by following the instructions provided in “Configuring an environment for developing and running CMP applications” on page 72.
2. Locate the source file for the sample that you want to change.

The source files for the samples are located in the following directories:

Deploy BAR sample

```
install_dir/sample/ConfigManagerProxy/cmp/DeployBar.java
```

Broker management sample

```
install_dir/sample/ConfigManagerProxy/cmp/BrokerInfo.java
```

IBM Integration API (CMP) Exerciser sample

```
install_dir/sample/ConfigManagerProxy/cmp/exerciser.java
```

3. Open the source file, and modify the appropriate parameters.
4. Save and recompile the source file.

What to do next

Next: Run the modified sample by following the instructions in the appropriate link:

- “Running the CMP Deploy BAR sample” on page 61
- “Running the CMP broker management sample” on page 63
- “Running the IBM Integration API (CMP) Exerciser sample” on page 64

Related concepts:



The IBM Integration API

The IBM Integration API is a programming interface that your applications can use to control brokers and their resources through a remote interface.

“The IBM Integration API samples” on page 60

Explore the samples to learn the basic features that are provided by the IBM Integration API (also known as the CMP). Run the samples to deploy a BAR file or manage a broker, or use the IBM Integration API (CMP) Exerciser to implement various tasks.

Related information:



IBM Integration API (CMP)

Configuring an environment for developing and running CMP applications

Prepare the environment in which you want to run your CMP applications.

Before you begin

Before you start:

To develop and run Java applications that use the CMP, you must install the following prerequisite software in your local computer environment:

- The WebSphere MQ Classes for Java.

These classes provide the internal wire protocol that your applications use to communicate with the broker.

You can install the classes from the media supplied with IBM Integration Bus.

- An IBM Java Development Kit (JDK) at a supported Java level. Java support is defined in Additional software requirements.

IBM Integration Bus does not supply a JDK; you must acquire and install a suitable product yourself.

You must use an IBM JDK to develop CMP applications and an IBM Java Runtime Environment (JRE) to run these applications.

About this task

To set up your computer in preparation for building and running CMP applications, you must configure your class path environment variable so that it includes the WebSphere MQ Classes for Java, and the JAR file that defines the CMP.

Follow the instructions provided for the appropriate environment:

- “Configuring the Windows command-line environment to run CMP applications” on page 73
- “Configuring Linux, UNIX, and z/OS command-line environments to run CMP applications” on page 74
- “Configuring the Eclipse environment to run CMP applications” on page 75

You can also run CMP applications, and therefore control one or more broker components, from computers on which you have not installed IBM Integration Bus. For more information, see “Configuring environments without the Integration Bus component installed” on page 76.

The JAR file `ConfigManagerProxy.jar` contains the English message catalog for displaying broker (BIP) messages from the Administration log of the broker. If you want a CMP application to display broker messages in a language other than English, you must also add the directory that contains the localized message catalogs to your class path; for example, `C:\Program Files\IBM\MQSI\v.r.m.f\messages` on Windows operating systems. The default directory includes the version, release, modification, and fix of the product, in the format `v.r.m.f` (version.release.modification.fix).

You can also use the CMP to display or log messages from a catalog that you create yourself.

Related concepts:



The IBM Integration API

The IBM Integration API is a programming interface that your applications can use to control brokers and their resources through a remote interface.

Related tasks:

“Configuring the Windows command-line environment to run CMP applications”

Use system facilities to configure environment variables on your Windows computers to run your CMP applications.

“Configuring Linux, UNIX, and z/OS command-line environments to run CMP applications” on page 74

Use system facilities to configure environment variables on your Linux, UNIX, and z/OS computers to run your CMP applications.

“Configuring the Eclipse environment to run CMP applications” on page 75

Use Eclipse facilities to configure the environment to run your CMP applications.

“Configuring environments without the Integration Bus component installed” on page 76

Install and run CMP applications and other utilities on computers, when you are using a local ID only, on which you have not installed the Integration Bus component.



Creating message catalogs

Create your own message catalogs to write tailored entries to the local error log.

Related reference:



Additional software requirements

IBM Integration Bus requires additional software products to run successfully.

Related information:



IBM Integration API (CMP)



WebSphere MQ Version 7 product documentation

Configuring the Windows command-line environment to run CMP applications

Use system facilities to configure environment variables on your Windows computers to run your CMP applications.

About this task

Update the CLASSPATH environment variable:

Procedure

1. Add the CMP JAR file to your CLASSPATH. For example:

```
set CLASSPATH = %CLASSPATH%;%install_dir%\classes\ConfigManagerProxy.jar
```

2. Add the WebSphere MQ Classes for Java JAR file `com.ibm.mq.jar` to your CLASSPATH in the same way.

On 32-bit operating system editions, the file is typically stored in the directory `C:\Program Files\IBM\WebSphere MQ\java\lib`. On 64-bit operating system editions, the file is typically stored in the directory `C:\Program Files (x86)\IBM\WebSphere MQ\java\lib`.

For more information about the WebSphere MQ Classes for Java, see the *Using Java* section in the WebSphere MQ information center.

3. Add your Java development directory to the CLASSPATH in the same way.

What to do next

Next: Use the tools that are provided by your JDK to build and run your CMP applications.

Related concepts:



The IBM Integration API

The IBM Integration API is a programming interface that your applications can use to control brokers and their resources through a remote interface.

Related tasks:

“Configuring an environment for developing and running CMP applications” on page 72

Prepare the environment in which you want to run your CMP applications.

“Configuring Linux, UNIX, and z/OS command-line environments to run CMP applications”

Use system facilities to configure environment variables on your Linux, UNIX, and z/OS computers to run your CMP applications.

“Configuring the Eclipse environment to run CMP applications” on page 75

Use Eclipse facilities to configure the environment to run your CMP applications.

Related information:



IBM Integration API (CMP)



WebSphere MQ Version 7 product documentation

Configuring Linux, UNIX, and z/OS command-line environments to run CMP applications

Use system facilities to configure environment variables on your Linux, UNIX, and z/OS computers to run your CMP applications.

About this task

Update the CLASSPATH environment variable:

Procedure

1. Add the CMP JAR to your CLASSPATH. For example:

```
export CLASSPATH = $CLASSPATH%:$install_dir/sample/ConfigManagerProxy/ConfigManagerProxy.jar
```

2. Add the WebSphere MQ Classes for Java JAR file `com.ibm.mq.jar` to your CLASSPATH in the same way.

On Linux and UNIX systems, the file is typically installed in the directory `MQ_INSTALLATION_PATH/java/lib`.

For more information about the WebSphere MQ Classes for Java, see the *Using Java* section in the WebSphere MQ information center.

3. Add your Java development directory to the CLASSPATH in the same way.

What to do next

Next: Use the tools that are provided by your JDK to build and run your CMP applications.

Related concepts:



The IBM Integration API

The IBM Integration API is a programming interface that your applications can use to control brokers and their resources through a remote interface.

Related tasks:

“Configuring an environment for developing and running CMP applications” on page 72

Prepare the environment in which you want to run your CMP applications.

“Configuring the Windows command-line environment to run CMP applications” on page 73

Use system facilities to configure environment variables on your Windows computers to run your CMP applications.

“Configuring the Eclipse environment to run CMP applications”

Use Eclipse facilities to configure the environment to run your CMP applications.

Related information:



IBM Integration API (CMP)



WebSphere MQ Version 7 product documentation

Configuring the Eclipse environment to run CMP applications

Use Eclipse facilities to configure the environment to run your CMP applications.

About this task

Update the CLASSPATH environment variable:

Procedure

1. In your Eclipse environment, select **File > New > Project**. The New Project wizard opens.
2. Select **Java Project** from the options displayed. Click **Next**. The New Java Project window opens.
3. Enter a name for your new project. Click **Next**.
4. Select the **Libraries** tab, and click **Add External Jars**.
5. To set up the build environment, navigate to the `install_dir/classes` subdirectory. For example, for Version 9.0.0.0 on Windows 32-bit operating

system editions, navigate to the directory C:\Program Files\IBM\MQSI\9.0.0.0\classes. Select the file ConfigManagerProxy.jar, and click **Open**. The file is added to the list in the window for the **Libraries** tab.

6. When you have added the file ConfigManagerProxy.jar to the build path, set up the runtime environment. Click **Add External Jars** again, and navigate to the *WebSphere_MQ_installation_directory*/java/lib subdirectory. For example, on Linux on x86, navigate to the directory /opt/mqm/java/lib. Select the file com.ibm.mq.jar, and click **Open**. This file is also added to the list.

For more information about the WebSphere MQ Classes for Java, see the *Using Java* section in the WebSphere MQ information center.

7. Click **Finish** to close the New Project wizard.

What to do next

Next: Use the Eclipse development tools to build and run your CMP applications.

Related concepts:



The IBM Integration API

The IBM Integration API is a programming interface that your applications can use to control brokers and their resources through a remote interface.

Related tasks:

“Configuring an environment for developing and running CMP applications” on page 72

Prepare the environment in which you want to run your CMP applications.

“Configuring the Windows command-line environment to run CMP applications” on page 73

Use system facilities to configure environment variables on your Windows computers to run your CMP applications.

“Configuring Linux, UNIX, and z/OS command-line environments to run CMP applications” on page 74

Use system facilities to configure environment variables on your Linux, UNIX, and z/OS computers to run your CMP applications.

Related information:



IBM Integration API (CMP)



WebSphere MQ Version 7 product documentation

Configuring environments without the Integration Bus component installed

Install and run CMP applications and other utilities on computers, when you are using a local ID only, on which you have not installed the Integration Bus component.

About this task

You can run Java applications that use the CMP even where you have not installed the Integration Bus component. These CMP applications include your own applications, and the following command utilities:

- **mqsicreateexecutiongroup**
- **mqsdeleteexecutiongroup**
- **mqsimode**
- **mqsipackagebar**

- **mqsireloadsecurity**; the CMP application version of **mqsireloadsecurity** is called **mqsireloadsecurityscript**.
- **mqsistartmsgflow**
- **mqsistopmsgflow**

To install CMP applications in an environment that does not have the Integration Bus component installed, complete the following steps:

Procedure

1. Ensure that the target computer has a compatible Java Runtime Environment (JRE). Because you are not installing the Integration Bus component, which includes a JRE, you must use an alternative option.
Note, that you must also set MQSI_JREPATH to the installation path of your JRE.
Java support is defined in Additional software requirements.
2. Copy the following set of files from a computer that has the Integration Bus component installed to the target computer:
 - a. ConfigManagerProxy.jar from the classes directory.
 - b. The WebSphere MQ Classes for Java.
 - On Windows, these classes are located in the file com.ibm.mq.jar.
 - On other platforms, these classes are located in the component's installation image.
 - c. Your CMP application and all configuration files, for example .broker files.
 - d. Optional: If you want to run the broker commands on the target computer, complete the following steps:
 - 1) Copy brokerutil.jar from the classes directory.
 - 2) Copy the required utility bat files, or shell scripts, from the bin directory. Shell scripts have a .bat extension on Windows or no extension on UNIX platforms:
 - **mqsicreateexecutiongroup** or **mqsicreateexecutiongroup.bat**
 - **mqsdeleteexecutiongroup** or **mqsdeleteexecutiongroup.bat**
 - **mqsimode** or **mqsimode.bat**
 - **mqsipackagebar** or **mqsipackagebar.bat**
 - **mqsireloadsecurity** or **mqsireloadsecurityscript.bat**
 - **mqsistartmsgflow** or **mqsistartmsgflow.bat**
 - **mqsistopmsgflow** or **mqsistopmsgflow.bat**
 - e. Optional: If you want to display broker (BIP) messages in English environments other than US English, copy all BIPmsgs*.properties files from the messages directory.
3. On the target computer, use system facilities to update the CLASSPATH environment variable to include the following files:
 - The JAR file that contains the definitions of the CMP classes, ConfigManagerProxy.jar.
 - Your applications that import the CMP classes.
 - The WebSphere MQ Classes for Java, com.ibm.mq.jar, and any additional JAR files required by this package.
 - Any other required JAR files and directories. For example, if you require any of the available command utilities on the target computer, include

brokerutil.jar; if you require the broker (BIP) messages to be displayed in locales other than US English, include a directory that contains BIPmsgs*.properties.

4. Ensure that the user ID that the target computer uses has the following authorities:
 - Authority to connect to the queue manager that the broker uses.
 - Authority to manipulate broker objects.

What to do next

Next: You can run your CMP applications, and the specified command utilities, on the target computer.

Related concepts:



The IBM Integration API

The IBM Integration API is a programming interface that your applications can use to control brokers and their resources through a remote interface.

Related tasks:

“Configuring an environment for developing and running CMP applications” on page 72

Prepare the environment in which you want to run your CMP applications.

“Managing brokers in a CMP application” on page 94

Manage the brokers and their resources from a CMP application.

Related reference:



Additional software requirements

IBM Integration Bus requires additional software products to run successfully.



mqscreateexecutiongroup command

Use the **mqscreateexecutiongroup** command to add a new integration server to a broker.



mqsdeleteexecutiongroup command

Use the **mqsdeleteexecutiongroup** command to remove an integration server from a broker.



mqsimode command

Use the **mqsimode** command to configure and retrieve operation mode information.



mqsipackagebar command

Use the **mqsipackagebar** command to create deployable broker archive (BAR) files. You can use this command to create BAR files on machines that do not have the IBM Integration Toolkit installed.



mqsireloadsecurity command

Use the **mqsireloadsecurity** command to force the immediate expiry of some or all the entries in the security cache.



mqsistartmsgflow command

Use the **mqsistartmsgflow** command to start integration servers, applications, and message flows.



mqsistopmsgflow command

Use the **mqsistopmsgflow** command to stop integration servers, applications, and

message flows.

Related information:

[WebSphere MQ Version 7 product documentation](#)

Connecting to a broker from a CMP application

Connect an application that uses the CMP to a broker, to send requests about its status and its resources.

Before you begin

Before you start

Before starting this step, you must have completed “Configuring an environment for developing and running CMP applications” on page 72.

About this task

Consider the following program `BrokerRunStateChecker.java`. It connects to a broker that is running on the default queue manager of the local computer.

```
import com.ibm.broker.config.proxy.*;

public class BrokerRunStateChecker {

    public static void main(String[] args) {

        // The ip address of where the broker is running
        // and the port number of the queue manager listener.
        displayBrokerRunState("localhost", 2414, "");
    }

    public static void displayBrokerRunState(String hostname,
                                           int port,
                                           String qmgr) {

        BrokerProxy b = null;
        try {
            BrokerConnectionParameters bcp =
                new MQBrokerConnectionParameters(hostname, port, qmgr);
            b = BrokerProxy.getInstance(bcp);
            String brokerName = b.getName();

            System.out.println("Broker '"+brokerName+
                               "' is available!");
            b.disconnect();
        } catch (ConfigManagerProxyException ex) {
            System.out.println("Broker is NOT available"+
                               " because "+ex);
        }
    }
}
```

The first line of the program requests Java to import the CMP classes, which are supplied in the package `com.ibm.broker.config.proxy`.

The first line of code inside the try block of the `displayBrokerRunState()` method instantiates a `BrokerConnectionParameters` object. This method is an interface which states that implementing classes are able to provide the parameters to connect to a broker.

The class `MQBrokerConnectionParameters` implements this interface; it defines a set of WebSphere MQ connection parameters. The constructor used here takes three parameters:

1. The host name of the computer that the broker is running on
2. The port on which the WebSphere MQ listener service for the broker is listening
3. The name of the queue manager that is associated with the broker

When you have defined this object, you can connect to the queue manager that is defined by those characteristics. The connection is achieved by the static `getInstance()` factory method just inside the try block. When a valid handle to the queue manager is returned, the application requests the name of the broker by using `b.getName()`, and displays it.

`getName()`, and other methods that request information from the broker, block until the information is supplied, or a timeout occurs. Therefore, if the broker is not running, the application hangs for a period. You can control the timeout period by using the `BrokerProxy.setRetryCharacteristics()` method. Typically, blocking only occurs when a given resource is accessed for the first time within an application.

Finally, the program calls the `disconnect()` method. This method frees up resources associated with the connection in both the CMP and the broker.

When a `BrokerProxy` handle is first returned from the `getInstance()` method, the broker service does not have to be running. It is only when the application uses the handle (by calling `getName()` in this example) that the application can be assured that a two-way connection with the broker is active.

Related concepts:



The IBM Integration API

The IBM Integration API is a programming interface that your applications can use to control brokers and their resources through a remote interface.

Related tasks:

“Configuring an environment for developing and running CMP applications” on page 72

Prepare the environment in which you want to run your CMP applications.

Related information:



IBM Integration API (CMP)

Navigating brokers and broker resources in a CMP application

Explore the status and attributes of the broker that your CMP application is connected to, and discover information about its resources.

Before you begin

You must have completed “Connecting to a broker from a CMP application” on page 79.

About this task

Each resource that the broker can control is represented as a single object in the IBM Integration API. A CMP application can request status and other information about the following objects:

- Brokers
- Integration servers
- Deployed message flows
- Administration log
- Activity logs
- data capture stores

The IBM Integration API also handles deployed message sets; these resources are handled as attributes of deployed integration servers.

Collectively known as *administered objects*, these objects provide most of the interface to the broker, and are therefore fundamental to an understanding of the IBM Integration API.

Each administered object is an instance of a Java class that describes the underlying type of object in the broker. The main Java classes are shown in the following table.

Java class	Class function
BrokerProxy	Describes brokers.
ExecutionGroupProxy	Describes integration servers.
MessageFlowProxy	Describes message flows that have already been deployed to integration servers; does not describe message flows in the Integration Development perspective of the IBM Integration Toolkit.
LogProxy	Represents the log of recent administrative activity on the broker, for all users.
ActivityLogProxy	Represents a log of recent message flow and resource activity.
DataCaptureProxy	Represents data that is held in a broker data store. Retrieved data is contained in DataCaptureEntry objects.

Each administered object describes a single object that can be controlled by the broker. For example, every integration server within a broker has one ExecutionGroupProxy instance that represents it within the application.

The LogProxy object includes messages, created as LogEntry objects, that record recent changes made to the administered objects. These messages include the following information:

- Administration requests; for example, a request to deploy a BAR file
- Results of administration requests
- Changes that have been made to objects; for example, a change to a property value through an administration task
- Changes that have been made to objects as a result of starting a broker at a new fix pack level or version.

The ActivityLogProxy object includes messages, created as ActivityLogEntry objects, that record recent high-level activities involving message flows and their interactions with external resources. For more information about these messages, see “Using Activity logs” on page 623. Activity logs are generated for message flows and for specific resource types.

You can retrieve an ActivityLogProxy object for a message flow from a MessageFlowProxy object. To get an ActivityLogProxy object for a resource type, use the ResourceManagerProxy object. Invoke the method getActivityLog() on the MessageFlowProxy object or the ResourceManagerProxy object. For further information, see “Working with Activity logs in a CMP application” on page 105.

You can use the `AdminQueueProxy` object to examine what items of work are in progress or are waiting for processing by the broker. The following code shows how you can access the queue:

```
BrokerConnectionParameters bcp =  
    new MQBrokerConnectionParameters("localhost", 1414, "QMGR");  
BrokerProxy b = BrokerProxy.getInstance(bcp);  
AdminQueueProxy l = b.getAdministrationQueue();
```

A set of public methods is available for each administered object, which applications can use to inquire and manipulate properties of the underlying broker to which the instance refers. To access an administered object through its API, your application must first request a handle to that object from the object that logically owns it.

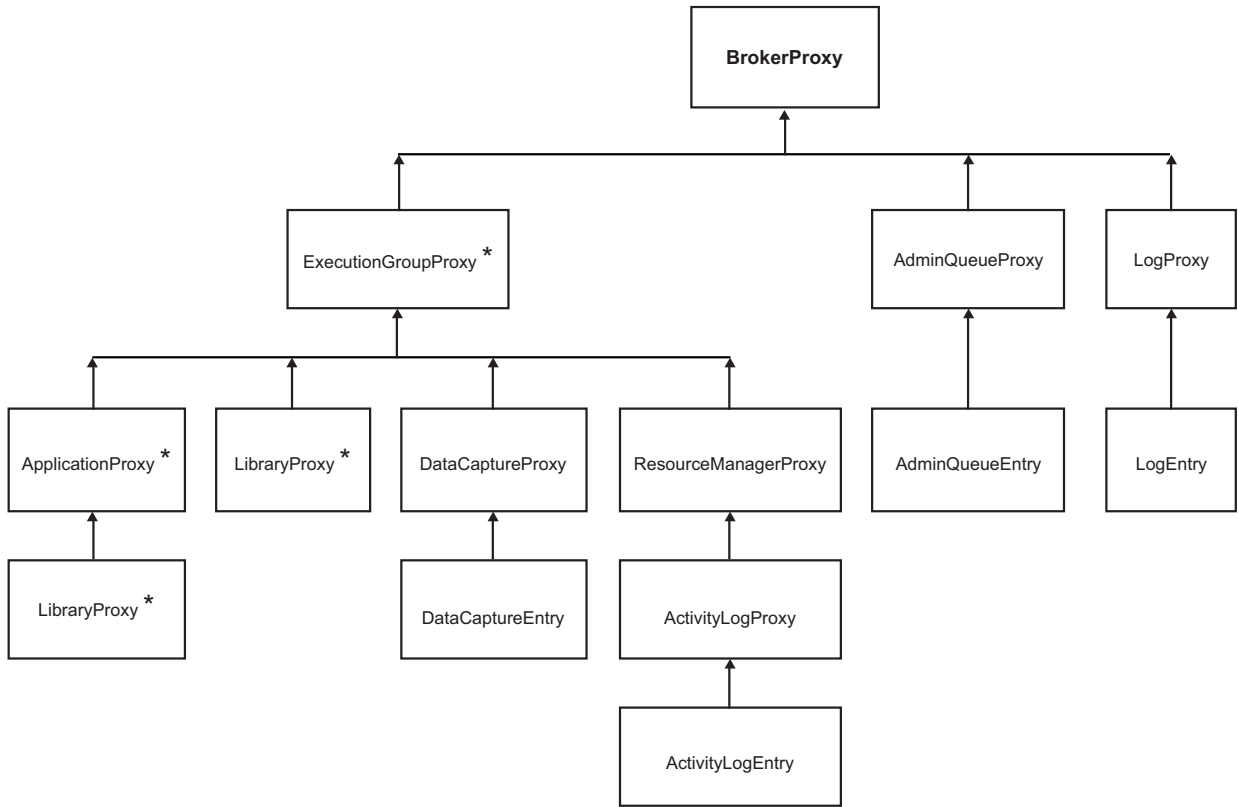
For example, because brokers logically own integration servers, to gain a handle to integration server EG1 running on broker B1, the application must ask the `BrokerProxy` object represented by B1 for a handle to the `ExecutionGroupProxy` object represented by EG1.

On a `BrokerProxy` object that refers to broker B1, the application can call methods that cause the broker to reveal its run-state, or cause it to start all its message flows. You can write applications to track the changes that are made to the broker objects by reading the messages maintained as `LogEntry` objects.

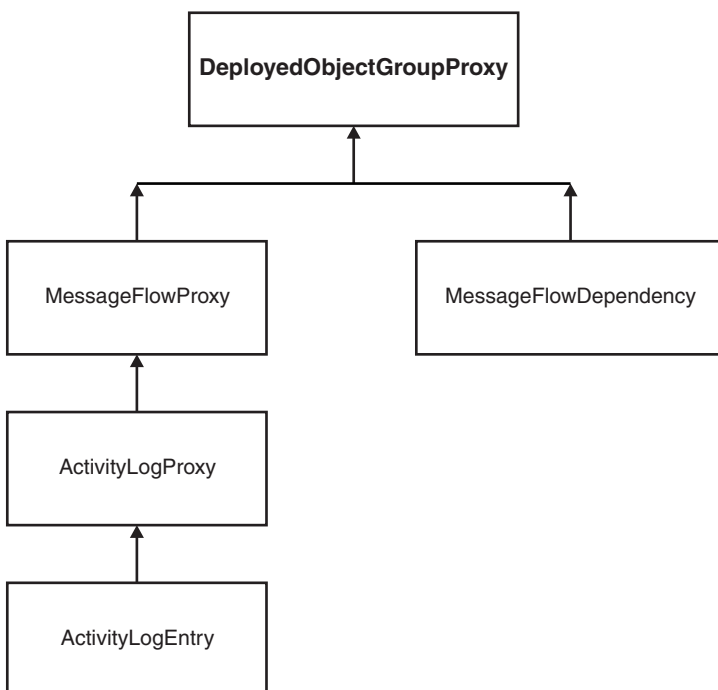
In the following example, a handle is requested to the `BrokerProxy` object. The `BrokerProxy` is logically the root of the administered object tree, therefore your application can access all other objects in the broker directly, or indirectly.

The broker directly owns the integration servers, therefore applications can call a method on the `BrokerProxy` object to gain a handle to the `ExecutionGroupProxy` objects. Similarly, the integration server logically contains the set of all message flows, therefore the application can call methods on the `ExecutionGroupProxy` object to access the `MessageFlowProxy` objects.

The hierarchy of these access relationships is shown in the following diagram.



Each object with an asterisk after its name inherits from the DeployedObjectGroupProxy class, and therefore has further child objects as shown in the following diagram.



The following application traverses the administered object hierarchy to discover the run-state of a deployed message flow. The application assumes that message flow MF1 is deployed to EG1 on broker B1; you can substitute these values in the code for other values that are valid in the broker.

```
import com.ibm.broker.config.proxy.*;

public class GetMessageFlowRunState {

    public static void main(String[] args) {

        BrokerProxy b = null;
        try {
            BrokerConnectionParameters bcp =
                new MQBrokerConnectionParameters(
                    "localhost",
                    1414,
                    "");
            b = BrokerProxy.getInstance(bcp);
        } catch (ConfigManagerProxyException cmpex) {
            System.out.println("Error connecting: "+cmpex);
        }

        if (b != null) {
            System.out.println("Connected to broker!");
            displayMessageFlowRunState(b, "EG1", "MF1");
            b.disconnect();
        }
    }

    private static void displayMessageFlowRunState(
        BrokerProxy b,
        String egName,
        String flowName) {

        try {
            ExecutionGroupProxy eg =
                b.getExecutionGroupByName(egName);

            if (eg != null) {
                MessageFlowProxy mf =
                    eg.getMessageFlowByName(flowName);

                if (mf != null) {
                    boolean isRunning = mf.isRunning();
                    System.out.print("Flow "+flowName+" on " +
                        egName+" on "+b.getName()+" is ");

                    if (isRunning) {
                        System.out.println("running");
                    } else {
                        System.out.println("stopped");
                    }
                } else {
                    System.err.println("No such flow "+flowName);
                }
            } else {
                System.err.println("No such exegrp "+egName+"!");
            }
        } catch (ConfigManagerProxyPropertyNotInitializedException
            ex) {
```

```

        System.err.println("Comms problem! "+ex);
    }
}

```

The method `displayMessageFlowRunState()` does most of the work. This method takes the valid `BrokerProxy` handle gained previously, and discovers the run-state of the message flow in the following way:

1. The `BrokerProxy` instance is used to gain a handle to the `ExecutionGroupProxy` object with the name described by the string `egName`
2. If a valid integration server is returned, the `ExecutionGroupProxy` instance is used to gain a handle to the `MessageFlowProxy` object with the name described by the string `flowName`.
3. If a valid message flow is returned, the run-state of the `MessageFlowProxy` object is queried, and the result is displayed.

The application does not have to know the names of objects that it can manipulate. Each administered object contains methods to return sets of objects that it logically owns. The following example demonstrates this technique by looking up the names of all integration servers within the broker.

```

import java.util.Enumeration;
import com.ibm.broker.config.proxy.*;

public class DisplayExecutionGroupNames {

    public static void main(String[] args) {

        BrokerProxy b = null;
        try {
            BrokerConnectionParameters bcp =
                new MQBrokerConnectionParameters(
                    "localhost",
                    1414,
                    "");
            b = BrokerProxy.getInstance(bcp);
        } catch (ConfigManagerProxyException cmpex) {
            System.out.println("Error connecting: "+cmpex);
        }

        if (b != null) {
            System.out.println("Connected to broker!");
            displayExecutionGroupNames(b);
            b.disconnect();
        }
    }

    private static void displayExecutionGroupNames(BrokerProxy b)
    {
        try {
            Enumeration<ExecutionGroupProxy> allEGs = b.getExecutionGroups(null);

            while (allEGs.hasMoreElements()) {
                ExecutionGroupProxy thisEG =
                    allEGs.nextElement();
                System.out.println("Found EG: "+thisEG.getName());
            }
        } catch (ConfigManagerProxyPropertyNotInitializedException
                ex) {

```

```

        System.err.println("Comms problem! "+ex);
    }
}
}

```

The key method is `BrokerProxy.getExecutionGroups(Properties)`. When supplied with a null argument, this method returns an enumeration of all the `ExecutionGroupProxy` objects in the broker. The application uses this method to look at each `ExecutionGroupProxy` in turn, and display its name.

The `Properties` argument of the method `BrokerProxy.getExecutionGroups(Properties)` can be used to exactly specify the characteristics of the integration servers that are sought. The application can use this argument for nearly all the methods that return administered objects, and is a powerful way of filtering those objects with which the application needs to work.

Examples of the characteristics that can be used to filter object lookups are the run-state and short description, as well as more obvious properties such as the name and UUID. To write logic to achieve filtered lookups, you must understand how each administered object stores its information.

The properties of each administered object are stored locally inside the object by using a hash table, where each property is represented as a {key, value} tuple. Each key is the name of an attribute (for example, name) and each value is the value (for example, BROKER1).

Each key name must be expressed by using a constant from the `AttributeConstants` class (`com.ibm.broker.config.proxy`). A complete set of keys and possible values for each administered object is described in the Java documentation for the `AttributeConstant` class, or by using the `Show raw property table` for this object function in the IBM Integration API (CMP) Exerciser sample application. The latter displays the complete list of {key, value} pairs for each administered object.

The `Properties` argument that is supplied to the lookup methods is a set of those {key, value} pairs that must exist in each administered object in the returned enumeration. Consider the following code fragment:

```

Properties p = new Properties();

p.setProperty(AttributeConstants.OBJECT_RUNSTATE_PROPERTY,
              AttributeConstants.OBJECT_RUNSTATE_RUNNING);

Enumeration<MessageFlowProxy> mf = executionGroup.getMessageFlows(p);

```

Providing that the variable `executionGroup` is a valid `ExecutionGroupProxy` object, the returned enumeration contains only active message flows (that is, `OBJECT_RUN_STATE_PROPERTY` equal to `OBJECT_RUNSTATE_RUNNING`).

When property filtering is applied to a method that returns a single administered object rather than an enumeration of objects, only the first result is returned (which is non-deterministic if more than one match applies). Therefore the following code:

```

Properties p = new Properties();
p.setProperty(AttributeConstants.NAME_PROPERTY,
              "EG1");
ExecutionGroupProxy eg1 = brokerProxy.getExecutionGroup(p);

```


is an alternative to the following statement:

```
ExecutionGroupProxy eg1 = brokerProxy.getTopicByName("EG1");
```

If multiple {key, value} pairs are added to a property filter, all properties must be present in the child object for an object to match. If you want a method to perform a logical OR, or a logical NOT, on a filter, you must write specific application code for this purpose.

When AdministeredObjects are first instantiated in an application, the CMP asks the broker for the current set of properties for that object. This action happens asynchronously, therefore the first time a property is requested, the CMP might pause while it waits for the information to be supplied by the broker. If the information does not arrive within a certain time (for example, if the broker is not running), a ConfigManagerProxyPropertyNotInitializedException is thrown. Your application can control the maximum time that the CMP waits by using the BrokerProxy.setRetryCharacteristics() method.

Procedure

Related concepts:



The IBM Integration API

The IBM Integration API is a programming interface that your applications can use to control brokers and their resources through a remote interface.

Related tasks:

“Configuring an environment for developing and running CMP applications” on page 72

Prepare the environment in which you want to run your CMP applications.

“Connecting to a broker from a CMP application” on page 79

Connect an application that uses the CMP to a broker, to send requests about its status and its resources.

Related information:



IBM Integration API (CMP)

Deploying resources to a broker from a CMP application

Deploy BAR files to the brokers in your broker network from a CMP application.

About this task

You can also check the result of a deployment by using the CMP.

Example

An example

This example connects to a broker that is running on the local computer. Its queue manager, IB9QMGR, is listening on port 2414. The code deploys a BAR file called MyBAR.bar to an integration server called default that is running on the broker, and displays the result.

```
import com.ibm.broker.config.proxy.*;
public class DeployBAR {

    public static void main(String[] args) {
        BrokerConnectionParameters bcp =
```

```

        new MQBrokerConnectionParameters("localhost", 2414, "IB9QMGR");
    try {
        BrokerProxy b = BrokerProxy.getInstance(bcp);
        ExecutionGroupProxy eg = b.getExecutionGroupByName("default");
        DeployResult dr = eg.deploy("MyBAR.bar", true, 30000);
        System.out.println("Result = "+dr.getCompletionCode());
    } catch (Exception e) {
        e.printStackTrace();
    }
}
}
}

```

IBM Integration API (CMP) Exerciser

About this task

You can also use the IBM Integration API (CMP) Exerciser to deploy files and check the results. For example:

Procedure

1. Connect to the broker by clicking **File > Connect to Broker**. This action opens the Connect to Broker dialog.
2. Enter the relevant connection parameters in the dialog. A hierarchical representation of the broker and its resources is displayed.
3. Complete one or more of the following operations:
 - Click an object in the tree to display the attributes of that object.
 - Right-click an object in the tree to call CMP methods that manipulate that object. For example, right-clicking a broker opens a menu that includes the items **Start user trace**.
 - Use the log pane at the bottom of the screen to view useful information that relates to the operation in progress.

Related concepts:


 Packaging and deployment overview

Deployment is the process of transferring data to an integration server on a broker so that it can take effect in the broker. Message flows and associated resources are packaged in broker archive (BAR) files for deployment.

 The IBM Integration API

The IBM Integration API is a programming interface that your applications can use to control brokers and their resources through a remote interface.

Related tasks:

 Packaging and deploying

Package resources that you create in the IBM Integration Toolkit, such as message flows, and deploy them to integration servers on brokers.

“Checking the results of deployment in a CMP application” on page 89

When you deploy from a CMP application, you can check the results of that action.

“Developing applications that use the IBM Integration API” on page 58

Develop Java applications that use the IBM Integration API (also known as the CMP) to communicate with, deploy to, and manage brokers and their associated resources.

“IBM Integration API (CMP) trace” on page 867

Enable or disable service trace for the IBM Integration API (also known as the

CMP).

“Resolving problems when developing CMP applications” on page 828
Use the advice given here to help you to resolve problems that can arise when developing IBM Integration API (also known as the CMP) applications.

Related information:

 [IBM Integration API \(CMP\)](#)

Checking the results of deployment in a CMP application

When you deploy from a CMP application, you can check the results of that action.

About this task

Code your application to test the results of the deploy actions that it takes. You can use code like the following snippet:

```
DeployResult dr = eg.deploy("MyBAR.bar", true, 30000);
System.out.println("Overall result = "+dr.getCompletionCode());

// Display log messages
Enumeration logEntries = dr.getLogEntries();
while (logEntries.hasMoreElements()) {
    LogEntry le = (LogEntry)logEntries.nextElement();
    System.out.println("General message: " + le.getDetail());
}
```

The deploy method blocks other processes until the broker has responded to the deployment request. When the method returns, the `DeployResult` object represents the outcome of the deployment at the time when the method returned; the object is not updated by the CMP.

If the deployment message could not be sent to the broker, a `ConfigManagerProxyLoggedException` exception is thrown at the time of the deployment. If the broker receives the deployment message, log messages for the overall deployment are displayed, followed by completion codes specific to each broker that is affected by the deployment. The completion code, shown in the following table, is one of the static instances from the `CompletionCodeType` class.

Completion code	Description
pending	The deployment is held in a batch and is not sent until you call <code>BrokerProxy.sendUpdates()</code> .
submitted	The deploy message was sent to the broker, but no response was received before the timeout period expired.
success	The broker has successfully completed the deployment.
failure	The broker has generated one or more errors during deployment. You can call the <code>getLogEntries()</code> method of the <code>DeployResult</code> class to get more information about the deployment failure. This method returns an enumeration of available <code>LogEntry</code> objects.

Related concepts:

 [Packaging and deployment overview](#)

Deployment is the process of transferring data to an integration server on a broker so that it can take effect in the broker. Message flows and associated resources are packaged in broker archive (BAR) files for deployment.

 [The IBM Integration API](#)

The IBM Integration API is a programming interface that your applications can use to control brokers and their resources through a remote interface.

Related tasks:



Packaging and deploying

Package resources that you create in the IBM Integration Toolkit, such as message flows, and deploy them to integration servers on brokers.

“Viewing Administration log information” on page 190

You can view Administration log information by using either the IBM Integration Explorer, or the CMP.

“IBM Integration API (CMP) trace” on page 867

Enable or disable service trace for the IBM Integration API (also known as the CMP).

“Resolving problems when deploying message flows or message sets” on page 752

Use the advice given here to help you to resolve common problems that can arise when you deploy message flows or message sets.

“Resolving problems when developing CMP applications” on page 828

Use the advice given here to help you to resolve problems that can arise when developing IBM Integration API (also known as the CMP) applications.

Related information:



IBM Integration API (CMP)

Setting message flow user-defined properties at run time in a CMP application

Use the CMP to query, discover, and set message flow user-defined properties dynamically at run time. You can use the CMP to set properties with a data type of character.

Before you begin

For user-defined properties on a message flow to be discoverable, the message flow must comply with the following conditions:

- The message flow must contain at least one of the following nodes:
 - JavaCompute
 - Compute
 - Database
 - Filter
 - PHPCompute
- The message flow must define the relevant user-defined property and provide an override value.

About this task

Tip: Use meaningful names and values for the properties that you define, so that you can understand their purpose and intent quickly. For example, a user-defined property named `property01`, with an initial value of `valueA` is not as useful as a property named `RouteToAorB` with an initial value of `RouteA`.

To query, discover, and set user-defined properties on a message flow, use the CMP to issue the following calls. For details about the calls, including the syntax and parameters to use, see the CMP documentation (IBM Integration API).

Procedure

1. Call `MessageFlowProxy.getUserDefinedPropertyNames()` to retrieve a list of all the user-defined properties that were defined by the Message Flow editor on the message flow or subflows.
A string array is returned that contains the property names.
2. Call `MessageFlowProxy.getUserDefinedProperty()` to retrieve the value of the specified user-defined property.
The value of the property is returned as a `Java.lang.String` value.
3. Call `MessageFlowProxy.setUserDefinedProperty()` to set a new value for the specified user-defined property.
The property must exist. You cannot change the data type of the existing user-defined property (`Java.lang.String`); therefore, you must ensure that the new value complies with the existing data type. The value that you set with the `MessageFlowProxy.setUserDefinedProperty()` call is populated to all relevant nodes in the message flow, including nodes in subflows.

Related concepts:

User-defined properties

A user-defined property (UDP) is a property that is defined when you construct a message flow by using the Message Flow editor. This property can be used by the ESQL or Java program inside message flow nodes, such as a Compute node.

User-defined properties in ESQL

Access user-defined properties (UDPs) as variables in your ESQL program by specifying the `EXTERNAL` keyword on a `DECLARE` statement. For example, the ESQL statement `DECLARE today EXTERNAL CHARACTER 'monday'` defines a user-defined property called `today` with an initial value `monday`.

The IBM Integration API


The IBM Integration API is a programming interface that your applications can use to control brokers and their resources through a remote interface.


ESQL variables

An ESQL variable is a data field that is used to help process a message.

Related tasks:

“Developing applications that use the IBM Integration API” on page 58
Develop Java applications that use the IBM Integration API (also known as the CMP) to communicate with, deploy to, and manage brokers and their associated resources.

 **Configuring a message flow at deployment time with user-defined properties**
Use user-defined properties (UDPs) to configure message flows at deployment and run time, without modifying program code. You can give a UDP an initial value when you declare it in your program, or when you use the Message Flow editor to create or modify a message flow.

 **Accessing message flow user-defined properties from a JavaCompute node**
Customize a JavaCompute node to access properties that you have associated with the message flow in which the node is included.

Related reference:

IBM Integration API

Use the IBM Integration API (CMP) Java classes and methods to develop CMP applications.

Configurable message flow properties

When you add a message flow to a broker archive (BAR) file in preparation for deploying it to a broker, you can set additional properties that influence its run time operation. These properties are available for review and update when you select the **Manage and Configure** tab for the broker archive file.

Working with properties of a configurable service of type `UserDefined` at run time in a `JavaCompute` node

Use the CMP in a `JavaCompute` node to query, set, create, and delete properties dynamically at run time in configurable services that you have defined with type `UserDefined`.

Before you begin

Before you start:

Complete the following tasks.

- Read the concept topic, [Configurable services](#).
- [Creating a message flow](#).

About this task

If you have created a `UserDefined` configurable service, and created properties for that service, you can work with those properties in a `JavaCompute` node. For example, you can create a `UserDefined` configurable service to set timeouts for processing HTTP messages.

You can create and delete configurable services in `JavaCompute` nodes, in the IBM Integration Explorer, and by using the `mqsicreateconfigurableservice` and `mqsdeleteconfigurableservice` commands.

Procedure

1. Right-click the `JavaCompute` node and click **Open Java** to create and open a Java file in the Editor view, or open an existing file.
2. Create the Java class for the node in which you want to include CMP methods.
3. Add the CMP JAR file `install_dir/classes/ConfigManagerProxy.jar` to the Java build path for the associated Java project.
4. Import `com.ibm.broker.config.proxy.*` in your code.
5. Add the following static method to the class you have created:

```
BrokerProxy b = BrokerProxy.getLocalInstance();
```

This method returns an instance of the `BrokerProxy` object for the broker to which the message flow (that contains this node) is deployed.
6. To ensure that the `BrokerProxy` object has been populated with data from the broker before you access the configurable service, add the following code:

```
while(!b.hasBeenPopulatedByBroker()) { Thread.sleep(100); }
```
7. Access the appropriate `UserDefined` configurable service:
 - a. If you know the name of the configurable service, use the following code to access it:

```
ConfigurableService myUDCS = b.getConfigurableService("UserDefined", "UD1");
```
 - b. If you want to select from a set of `UserDefined` configurable services, use the following code to get a list of all services of a particular type:

```
ConfigurableService[] UD_set = b.getConfigurableServices("UserDefined");
```

8. Add further code to access and use the specific properties that you are interested in. For example:

- Retrieve the properties that are defined to that service:

```
String[] props = myUDCS.getProperties();
```

- Create a new property:

```
String newprop = 'VerifyRequestTimeout';  
String newval = '15';  
myUDCS.setProperty(newprop, newval);
```

- Delete a property:

```
myUDCS.deleteProperty(newprop);
```

You can also use the `deleteProperties()` method to delete more than one property.

You can delete properties in `UserDefined` configurable services only. If you use this method on a configurable service of a different type, a `ConfigManagerProxyLoggedException` is generated.

9. Deploy the JAR file, and all associated message flows, in a BAR file. You do not have to deploy the `ConfigManagerProxy.jar` file to the target integration server, because the broker can access these classes independently.

Related concepts:



The IBM Integration API

The IBM Integration API is a programming interface that your applications can use to control brokers and their resources through a remote interface.



Using the IBM Integration Explorer to work with configurable services

Configurable services are used to define properties that are related to external services on which the integration node (broker) relies. Use the IBM Integration Explorer to view, add, modify and delete configurable services.



Configurable services

Configurable services are typically runtime properties. You can use them to define properties that are related to external services on which the broker relies; for example, an SMTP server or a JMS provider.

Related tasks:

“Developing applications that use the IBM Integration API” on page 58

Develop Java applications that use the IBM Integration API (also known as the CMP) to communicate with, deploy to, and manage brokers and their associated resources.



Creating a message flow

Create a message flow to specify how to process messages in the broker. You can create one or more message flows and deploy them to one or more brokers.



Configuring message flows to process timeouts

Connect the HTTP Timeout terminal of the HTTPInput or SOAPInput nodes to further nodes to process timeouts.

Related reference:



IBM Integration API

Use the IBM Integration API (CMP) Java classes and methods to develop CMP applications.



JavaCompute node

Use the JavaCompute node to work with messages by using the Java language.



mqsicreateconfigurablecommand command

Use the **mqsicreateconfigurablecommand** command to create an object name for a broker external resource.



mqsideleteconfigurablecommand command

Use the **mqsideleteconfigurablecommand** command to delete a configurable service, such as a JMS provider, JDBC provider, or FTP server, that you have created by using the **mqsicreateconfigurablecommand** command.

Managing brokers in a CMP application

Manage the brokers and their resources from a CMP application.

Before you begin

Before you start

Before you start this task, you must have completed the task “Connecting to a broker from a CMP application” on page 79.

About this task

Use the CMP to change the state of objects in the domain; you can create, delete, modify, and deploy objects stored within it. The following example sets the long description field of the broker:

```
import com.ibm.broker.config.proxy.*;

public class SetLongDescription {

    public static void main(String[] args) {

        BrokerProxy b = null;
        try {
            BrokerConnectionParameters bcp =
                new MQBrokerConnectionParameters(
                    "localhost",
                    1414,
                    "");
            b = BrokerProxy.getInstance(bcp);
            b.setLongDescription("this is my broker");
            b.disconnect();
        } catch (ConfigManagerProxyException cmpex) {
            System.out.println("Error connecting: "+cmpex);
        }
    }
}
```

The broker processes requests to change properties from the CMP asynchronously, therefore if your application calls `getLongDescription()` immediately following the call to `setLongDescription()`, the response might return the old value of the property. For more information, see “Checking the results of broker management with the most recent completion code in a CMP application” on page 97.

What to do next

Next:

Most state-changing CMP methods return control immediately without informing the calling application of the outcome of the request. To discover this information, see “Checking the results of broker management in a CMP application” on page 96.

Related concepts:



The IBM Integration API

The IBM Integration API is a programming interface that your applications can use to control brokers and their resources through a remote interface.

Related tasks:

“Configuring an environment for developing and running CMP applications” on page 72

Prepare the environment in which you want to run your CMP applications.

“Connecting to a broker from a CMP application” on page 79

Connect an application that uses the CMP to a broker, to send requests about its status and its resources.

“Managing brokers from JavaCompute nodes” on page 102

You can use the CMP to manage brokers and their associated resources from JavaCompute nodes in deployed message flows.

Related information:



IBM Integration API (CMP)

Creating objects in a CMP application

Create new objects associated with a broker.

About this task

The following example adds an integration server called EG2 to the connected broker.

```
import com.ibm.broker.config.proxy.*;

public class AddExecutionGroup {

    public static void main(String[] args) {

        BrokerProxy b = null;
        try {
            BrokerConnectionParameters bcp =
                new MQBrokerConnectionParameters(
                    "localhost",
                    1414,
                    "");
            b = BrokerProxy.getInstance(bcp);
            ExecutionGroupProxy e = b.createExecutionGroup("EG2");
            b.disconnect();
        } catch (ConfigManagerProxyException cmplex) {
            System.out.println("Error connecting: "+cmplex);
        }
    }
}
```

Because requests are processed asynchronously by the broker, the `ExecutionGroupProxy` object that is returned from the `createExecutionGroup()` method is a skeleton object when it is returned to your application, because it refers to an object that might not yet exist in the broker. The application can manipulate the object as if it existed on the broker, although the actual creation of the underlying object might not happen for some time.

If the request to create the object described by the skeleton fails, all requests that use the skeleton also fail. Therefore, if integration server EG2 cannot be created, all subsequent requests that concern the skeleton object fail. However, unless the application explicitly checks for errors, it works in the same way as it does in the successful case, because no exception is thrown unless, because of a communication problem, a message cannot be sent to the broker.

See “Checking the results of broker management in a CMP application” for further information about how to detect problems such as these.

Related concepts:



The IBM Integration API

The IBM Integration API is a programming interface that your applications can use to control brokers and their resources through a remote interface.

Related tasks:

“Configuring an environment for developing and running CMP applications” on page 72

Prepare the environment in which you want to run your CMP applications.

“Checking the results of broker management in a CMP application”

When you have made a change to the broker, use one of the supplied methods to check if the change has been successful.

Related reference:



`mqsicreatebroker` command

Use the `mqsicreatebroker` command to create a broker and its associated resources.

Related information:



IBM Integration API (CMP)

Checking the results of broker management in a CMP application

When you have made a change to the broker, use one of the supplied methods to check if the change has been successful.

About this task

Choose one of three methods in the CMP to determine the outcome of requests to create, delete, modify, and deploy resources:

- If you have initiated a deployment method, you can use the return code from the deployment API; this technique is shown in “Checking the results of deployment in a CMP application” on page 89.
- You can query an object’s most recent completion code; this option is shown in “Checking the results of broker management with the most recent completion code in a CMP application” on page 97.
- You can use the administered object notification mechanism; by using this approach, you can code specific routines to handle the responses and take

appropriate action, and improve the efficiency of your program. See “Checking the results of broker management with object notification in a CMP application” on page 98.

- If you prefer to make property changes synchronously, call the method `BrokerProxy.setSynchronous()` before making your changes.

If subsequent property change methods give a successful return, the request has been successfully completed by the broker. If the changes are rejected by the broker, or time out (they cannot be completed successfully by the broker before the timeout period expires), the methods throw either a `ConfigManagerProxyRequestFailure` or a `ConfigManagerProxyRequestTimeout` exception.

For more information on synchronous property changes, see the description of the `BrokerProxy.setSynchronous()` method in the CMP Javadoc information.

Related concepts:



The IBM Integration API

The IBM Integration API is a programming interface that your applications can use to control brokers and their resources through a remote interface.

Related tasks:

“Configuring an environment for developing and running CMP applications” on page 72

Prepare the environment in which you want to run your CMP applications.

“Checking the results of broker management with the most recent completion code in a CMP application”

Use an object's most recent completion code to determine the outcome of a request that your application made against that object.

“Checking the results of broker management with object notification in a CMP application” on page 98

Use object notification to determine the outcome of a request that your application made against the object.

Related information:



IBM Integration API (CMP)

Checking the results of broker management with the most recent completion code in a CMP application:

Use an object's most recent completion code to determine the outcome of a request that your application made against that object.

About this task

Most state-changing methods in the CMP do not provide return code that indicates the success or failure of a specific action. For these methods, you must write different code to discover the outcome of the action. Assuming that administered objects are not shared across threads, the following code fragment can be used to discover the outcome of a request to modify a broker's `LongDescription`, where `b` is an instance of a `BrokerProxy` object:

```
GregorianCalendar oldCCTime =
    b.getTimeOfLastCompletionCode();
b.setLongDescription(newDesc);
GregorianCalendar newCCTime = oldCCTime;
while ((newCCTime == null) || (newCCTime.equals(oldCCTime))) {
    newCCTime = b.getTimeOfLastCompletionCode();
}
```

```

    Thread.sleep(1000);
}
CompletionCodeType ccType = b.getLastCompletionCode();
if (ccType == CompletionCodeType.success) {
    // etc.
}

```

In this example, the application initially determines when an action on the broker was last completed, using the `getTimeOfLastCompletionCode()` method. This method returns the time that the topology last received a completion code or, if no return codes have been received, a null value. The application updates the broker's `LongDescription`, then continually monitors the topology, waiting for the results of the `setLongDescription()` command to be returned to the CMP. When the results are returned, control breaks out of the `while` loop and the last completion code is determined.

As well as being unsuitable for a multi-threaded application, this algorithm for determining the outcome of commands is inefficient, because it causes the CMP application to wait while the broker processes the request.

For a more efficient application, and one that is suitable for a multi-threaded environment, code the alternative approach that uses administered object notifications; see “Checking the results of broker management with object notification in a CMP application.”

If you prefer, you can make property changes synchronously by using the `methodBrokerProxy.setSynchronous()` method. When you make synchronous property changes, methods such as `setLongDescription()` do not return until the change has been processed by the broker. For more information on synchronous property changes, see the description of the `BrokerProxy.setSynchronous()` method in the CMP Javadoc information.

Related concepts:



The IBM Integration API

The IBM Integration API is a programming interface that your applications can use to control brokers and their resources through a remote interface.

Related tasks:

“Checking the results of broker management in a CMP application” on page 96

When you have made a change to the broker, use one of the supplied methods to check if the change has been successful.

“Checking the results of broker management with object notification in a CMP application”

Use object notification to determine the outcome of a request that your application made against the object.

Related information:



IBM Integration API (CMP)

Checking the results of broker management with object notification in a CMP application:

Use object notification to determine the outcome of a request that your application made against the object.

About this task

The CMP can notify applications whenever commands complete, or whenever changes occur to administered objects. By making use of the OBSERVER design pattern, your CMP application can define a handle to a user-supplied object that has a specific method that is called if an object is modified or deleted, or whenever a response to a previously submitted action is returned from the broker.

The user-supplied code must implement the `AdministeredObjectListener` interface. This interface defines methods that are invoked by the CMP when an event occurs on an administered object to which the listener is registered. The following methods are defined:

processModify(...)

`processModify(...)` is called when the administered object to which the listener is registered has one or more of its attributes modified by the broker. The following information is included in the notification through the use of the `processModify()` method arguments:

1. A handle to the `AdministeredObject` to which the notification refers.
2. A list of strings that contain the key names that have been changed.
3. A list of strings that describe new subcomponents that have been created for the object; for example, new integration servers in a broker.
4. A list of strings that describe subcomponents that have been removed from the object.

The format of the strings passed to the final two parameters is an internal representation of the administered object. You can turn this representation into an administered object type by using the `getManagedSubcomponentFromStringRepresentation()` method.

Consider the following additional information:

1. Strings are passed within these lists to enhance performance; the CMP does not use resource instantiating administered objects, unless they are specifically requested by the calling application.
2. The first time you call the `processModify()` method for a listener, the `changed attributes` parameter can include a complete set of attribute names for the object, if the application is using a batch method, or if the CMP is experiencing communication problems with the broker.

processDelete(...)

`processDelete(...)` is called if the object with which the listener is registered is completely removed from the broker. Supplied to `processDelete(...)` is one parameter – a handle to the administered object that has been deleted; when this method returns, the administered object handle might no longer be valid. At about the same time that a `processDelete(...)` event occurs, a `processModify(...)` event is sent to listeners of the deleted object's parent, to announce a change in the parent's list of subcomponents.

processActionResponse(...)

`processActionResponse(...)` is the event that informs the application that a previous action submitted by that application is complete. Only one `processActionResponse(...)` event is received for each state-changing operation issued by the CMP application. This event contains the following items of information:

1. A handle to the administered object for which a request was submitted.

2. The completion code of the request.
3. A set of zero, or more, informational (BIP) messages associated with the result.
4. A set of (key, value) pairs that describes the submitted request in more detail. Check the documentation for information about how to parse these pairs.

To register a listener, each administered object has a `registerListener()` method that is used to tell the CMP to call the supplied code whenever an event occurs on that object. You can register the same `AdministeredObjectListener` for notifications from multiple administered objects. You can also register multiple `AdministeredObjectListeners` against the same administered object.

The following example demonstrates this technique by registering a listener on the broker object, and displaying a message whenever it is modified:

```
import com.ibm.broker.config.proxy.*;
import com.ibm.broker.config.proxy.CompletionCodeType;
import java.util.List;
import java.util.ListIterator;
import java.util.Properties;

public class MonitorBroker implements AdministeredObjectListener {

    public static void main(String[] args) {

        BrokerProxy b = null;
        try {
            BrokerConnectionParameters bcp =
                new MQBrokerConnectionParameters(
                    "localhost",
                    1414,
                    "");
            b = BrokerProxy.getInstance(bcp);
        } catch (ConfigManagerProxyException cmpex) {
            System.out.println("Error connecting: "+cmpex);
        }

        if (b != null) {
            System.out.println("Connected to broker");
            listenForChanges(b);
            b.disconnect();
        }
    }

    private static void listenForChanges(AdministeredObject obj)
    {
        if (obj != null) {
            obj.registerListener(new MonitorBroker());
            while(true) {
                // thread could do something else here instead
                try {
                    Thread.sleep(10000);
                } catch (InterruptedException ex) {
                    // ignore
                }
            }
        }
    }

    public void processActionResponse(AdministeredObject obj,
                                     CompletionCodeType cc,
                                     List bipMessages,
                                     Properties refProperties) {
        // Event ignored in this example
    }
}
```

```

}

public void processDelete(AdministeredObject deletedObject) {
    // Event ignored in this example
}

public void processModify(AdministeredObject affectedObject,
                          List changedAttributes,
                          List newChildren,
                          List removedChildren) {
    try {
        System.out.println(affectedObject+" has changed:");
        ListIterator e = changedAttributes.listIterator();
        while (e.hasNext()) {
            String changedAttribute = (String) e.next();
            System.out.println("Changed: "+changedAttribute);
        }
        ListIterator e2 = newChildren.listIterator();
        while (e2.hasNext()) {
            String newChildStr = (String) e2.next();
            AdministeredObject newChild =
                affectedObject.getManagedSubcomponentFromStringRepresentation(newChildStr);
            System.out.println("New child: "+newChild);
        }
        ListIterator e3 = removedChildren.listIterator();
        while (e3.hasNext()) {
            String remChildStr = (String) e3.next();
            AdministeredObject removedChild =
                affectedObject.getManagedSubcomponentFromStringRepresentation(remChildStr);
            System.out.println("Removed child: "+removedChild);
        }
    } catch (ConfigManagerProxyPropertyNotInitializedException ex) {
        ex.printStackTrace();
    }
}
}

```

The `listenForChanges()` method attempts to register an instance of the `MonitorBroker` class for notifications of broker changes. If successful, the main thread pauses indefinitely to prevent the application from ending when the method returns. When the listener is registered, whenever the broker changes (for example, if an integration server is added), the `processModify()` method is called. This method displays details of each notification on the screen.

You must register separate listeners for each administered object on which you want to receive notifications. You can use the same listener instance for multiple administered objects.

You can stop receiving notifications in three ways:

- `AdministeredObject.deregisterListener(AdministeredObjectListener)`
- `ConfigManagerProxy.deregisterListeners()`
- `ConfigManagerProxy.disconnect()`

The first method de-registers a single listener from a single administered object; the other two methods deregister all listeners connected with that `BrokerProxy` instance. In addition, the final method shows that all listeners are implicitly removed when connection to the broker is stopped.

You can also implement the `AdvancedAdministeredObjectListener` interface which, when registered, yields additional information to applications.

Related concepts:



The IBM Integration API

The IBM Integration API is a programming interface that your applications can use to control brokers and their resources through a remote interface.

Related tasks:

“Checking the results of broker management in a CMP application” on page 96

When you have made a change to the broker, use one of the supplied methods to check if the change has been successful.

“Checking the results of broker management with the most recent completion code in a CMP application” on page 97

Use an object's most recent completion code to determine the outcome of a request that your application made against that object.

Related information:



IBM Integration API (CMP)

Managing brokers from JavaCompute nodes

You can use the CMP to manage brokers and their associated resources from JavaCompute nodes in deployed message flows.

Before you begin

Before you start

Before starting this step, you must create a JavaCompute node in a message flow.

About this task

Use CMP methods and classes in your JavaCompute node to explore and manage brokers and other resources.

Procedure

1. Create the Java class for the node in which you want to include CMP methods.
2. Add the CMP JAR file `install_dir/classes/ConfigManagerProxy.jar` to the Java build path for the associated Java project.
3. Add the following static method to the class:

```
BrokerProxy thisBroker = BrokerProxy.getLocalInstance();
```

This method returns an instance of the BrokerProxy object for the broker to which the message flow (that contains this node) is deployed.

4. To work with an integration server on this broker, add the following static method to your code:

```
ExecutionGroupProxy thisEG = ExecutionGroupProxy.getLocalInstance();
```

This method returns an instance of the ExecutionGroupProxy object for the integration server to which the message flow is deployed.

5. If you want to connect to a different broker that you have created on the computer to which your node and message flow are deployed, you can use a variant of this class:

```
BrokerProxy secondBroker = BrokerProxy.getLocalInstance(string)
```

Specify the name of the alternative local broker as the value of the variable `string`. Your code can manage this second broker, and its associated resources, by using the BrokerProxy object that is returned by this call.

6. Include additional CMP methods in your Java code to run the operations that you want against the broker or integration server by using the objects obtained in previous steps. You can follow the guidance that is provided in other topics in this section for further information and examples that show how to use CMP methods in CMP applications.

If you include methods that affect the message flow in which your CMP application is running, it might not be able to receive all notifications that these operations have successfully completed. Stopping, deleting, and redeploying the message flow are examples in this category; consider carefully the consequences of using these methods.

7. Deploy the JAR file, and all associated message flows, in a BAR file. You do not have to deploy the `ConfigManagerProxy.jar` file to the target integration server, because the broker can access these classes independently.

Related concepts:



The IBM Integration API

The IBM Integration API is a programming interface that your applications can use to control brokers and their resources through a remote interface.

Related tasks:

“Configuring an environment for developing and running CMP applications” on page 72

Prepare the environment in which you want to run your CMP applications.



Creating Java code for a JavaCompute node

Use these instructions to associate Java code with your JavaCompute node.

Related reference:



JavaCompute node

Use the JavaCompute node to work with messages by using the Java language.

Related information:



IBM Integration API (CMP)

Working with resource statistics in a CMP application

Start, stop, and review status of resource statistics collection in your CMP applications.

Before you begin

Before you start:

- Read the concept topic about resource statistics.

About this task

You can create CMP applications to examine and control the collection of resource statistics.

Checking what resource types can return statistics

```

////////////////////////////////////
// Sample CMP API code that connects to a local broker
// called 'testbrk' and writes out available
// resource types on the broker that have the
// ability to emit resource-level statistics.
BrokerProxy b = null;
try {

```

```

        b = BrokerProxy.getLocalInstance("testbrk");
String[] resourceNames = b.getResourceTypeNames();
for (String thisResource : resourceNames) {
    System.out.println(thisResource);
}
} catch (ConfigManagerProxyLoggedException e) {
    e.printStackTrace();
} catch (ConfigManagerProxyPropertyNotInitializedException e) {
    e.printStackTrace();
}
}

```

Checking for resource names associated with a specific resource type

```

////////////////////////////////////
// Sample CMP API code that connects to a local broker
// called 'testbrk' and writes out resource property
// names reported for a specific resource type.
BrokerProxy b = null;
try {
    b = BrokerProxy.getLocalInstance("testbrk");
String[] resourcePropertyNames =
    b.getResourceTypeStatisticsPropertyNames("JVM");
for (String thisResourceProperty : resourcePropertyNames) {
    System.out.println(thisResourceProperty);
}
} catch (ConfigManagerProxyLoggedException e) {
    e.printStackTrace();
} catch (ConfigManagerProxyPropertyNotInitializedException e) {
    e.printStackTrace();
}
}

```

Starting statistics collection

```

////////////////////////////////////
// Sample CMP API code that connects to a local broker
// called 'testbrk' and gets a reference to the execution
// group called 'default'. It then enables resource
// statistics for all the integration server's resource types.
BrokerProxy b = null;
try {
    b = BrokerProxy.getLocalInstance("testbrk");
ExecutionGroupProxy eg = b.getExecutionGroupByName("default");
if (eg != null) {
    eg.setResourceStatisticsEnabled(null, true);
}
} catch (ConfigManagerProxyLoggedException e) {
    e.printStackTrace();
} catch (ConfigManagerProxyPropertyNotInitializedException e) {
    e.printStackTrace();
}
}

```

Stopping statistics collection

```

////////////////////////////////////
// Sample CMP API code that connects to a local broker
// called 'testbrk' and gets a reference to the execution
// group called 'default'. It then disables resource
// statistics for all the integration server's resource types.
BrokerProxy b = null;
try {
    b = BrokerProxy.getLocalInstance("testbrk");
ExecutionGroupProxy eg = b.getExecutionGroupByName("default");
if (eg != null) {
    eg.setResourceStatisticsEnabled(null, false);
}
}
}

```

```

    }
    } catch (ConfigManagerProxyLoggedException e) {
        e.printStackTrace();
    } catch (ConfigManagerProxyPropertyNotInitializedException e) {
        e.printStackTrace();
    }
}

```

Viewing statistics collection status

```

////////////////////////////////////
// Sample CMP API code that connects to a local broker
// called 'testbrk' and gets a reference to the execution
// group called 'default'. It then writes out if resource
// statistics is enabled.
BrokerProxy b = null;
try {
    b = BrokerProxy.getLocalInstance("testbrk");
    ExecutionGroupProxy eg = b.getExecutionGroupByName("default");
    if (eg != null) {
        System.out.println(eg.getResourceStatisticsEnabled(null));
    }
} catch (ConfigManagerProxyLoggedException e) {
    e.printStackTrace();
} catch (ConfigManagerProxyPropertyNotInitializedException e) {
    e.printStackTrace();
}

```

Related concepts:

“Resource statistics” on page 530

Resource statistics are collected by a broker to record performance and operating details of resources that are used by integration servers.

Related tasks:

“Developing applications that use the IBM Integration API” on page 58

Develop Java applications that use the IBM Integration API (also known as the CMP) to communicate with, deploy to, and manage brokers and their associated resources.

“Analyzing resource performance” on page 529

You can collect statistics to assess the performance of resources used by integration servers.

Related reference:



Resource statistics data

Learn about the measurements for which data is returned when you activate resource statistics collection.

Working with Activity logs in a CMP application

View Activity log entries for message flows and resource types in your CMP applications.

Before you begin

Before you start:

- Read the concept topic “Activity log overview” on page 624.

About this task

You can create CMP applications to examine and analyze your Activity logs.

Checking which resource types can generate Activity logs

```
/*
 * Sample CMP API code that connects to a local broker
 * called 'testbrk' and lists the available
 * resource types on the integration server 'exgp' that can
 * generate Activity
 * logs.
 */
try {
    BrokerProxy b = BrokerProxy.getLocalInstance("testbrk");
    if (b != null) {
        ExecutionGroupProxy e = b.getExecutionGroupByName("exgp");
        if (e != null) {
            Properties rms = new Properties();
            rms.setProperty(AttributeConstants.ACTIVITYLOG_SUPPORTED_PROPERTY, AttributeConstants.TRUE);
            Enumeration <ResourceManagerProxy> rmps = e.getResourceManagers(rms);
            while (rmps.hasMoreElements()) {
                ResourceManagerProxy rmp = rmps.nextElement();
                String name = rmp.getName();
                System.out.println(name);
            }
        }
    }
} catch (ConfigManagerProxyException ex) {
    ex.printStackTrace();
}
```

Retrieving Activity log entries for a particular resource type

```
/*
 * Sample CMP API code that connects to a local broker
 * called 'testbrk' and retrieves and prints out Activity
 * log
 * entries for resource type 'JMS' on integration server 'exgp'.
 */
try {
    BrokerProxy b = BrokerProxy.getLocalInstance("testbrk");
    if (b != null) {
        ExecutionGroupProxy e = b.getExecutionGroupByName("exgp");
        if (e != null) {
            ResourceManagerProxy rmp = e.getResourceManagerByName("JMS");
            if (rmp != null) {
                ActivityLogProxy al = rmp.getActivityLog();
                if (al != null) {
                    for (int i = 1; i <= al.getSize(); i++) {
                        ActivityLogEntry ale = al.getLogEntry(i);
                        System.out.println(ale);
                        System.out.println(ale.getMessageNumber());
                    }
                }
            }
        }
    }
} catch (ConfigManagerProxyException ex) {
    ex.printStackTrace();
}
```

Retrieving Activity log entries for a particular message flow

```
/*
 * Sample CMP API code that connects to a local broker
 * called 'testbrk' and gets references to the execution
 * group called 'default', the application belonging to this integration server called 'app',
 * and the message flow in the application called 'msgflow'.
 * It prints out the Activity
 * log entries for message flow 'msgflow'.
 */
try {
    BrokerProxy b = BrokerProxy.getLocalInstance("testbrk");
    if (b != null) {
        ExecutionGroupProxy e = b.getExecutionGroupByName("default");
        if (e != null) {
            ApplicationProxy ap = e.getApplicationByName("app");
            if (ap != null) {
                MessageFlowProxy mf = ap.getMessageFlowByName("msgflow");
            }
        }
    }
}
```



```

String output = dcp.getDataCaptureEntryAsXml(1);
System.out.println("XML output: "+output);

int messNo = 0;

Enumeration <DataCaptureEntry> dceE = dcp.elements();
while (dceE.hasMoreElements()) {
    DataCaptureEntry dce = dceE.nextElement();
    System.out.print("\nMessage: "+messNo++);

    Properties props = dce.getAllProperties();
    String[] columns = dce.getPropertyNames();
    for (int i = 0; i < columns.length; i++) {
        System.out.print("\nProperty: "+columns[i]+ " Value: "+props.getProperty(columns[i]));
    }
}
} catch (ConfigManagerProxyLoggedException e) {
    e.printStackTrace();
} catch (ConfigManagerProxyPropertyNotInitializedException e) {
    e.printStackTrace();
} finally {
    if(bp != null) {
        bp.disconnect();
    }
}
}

```

The following example shows how to replay data.

```

BrokerProxy brokerProxy = null;
try {
    // DataCaptureStore containing message to replay
    String dataCaptureStore = "MyStore";

    // ID of message to replay
    String replayMsgId = "0123456789ABCDEF";

    // DataDestination to replay message to
    String dataDestination = "ReplayDestination";

    // EG to replay
    String replayEG = "MyExecutionGroupName";

    Properties dataCaptureProps = new Properties();

    // Set the message to replay
    dataCaptureProps.setProperty(DataCaptureEntry.PROPERTY_WMBMSG_KEY, replayMsgId);

    // Set the destination to replay to
    dataCaptureProps.setProperty(AttributeConstants.DATACAPTURE_REPLAY, dataDestination);

    DataCaptureEntry dataCaptureEntry = new DataCaptureEntry(dataCaptureProps );

    brokerProxy = WebAdminBrokerProxy.getLocalInstance("IB9NODE");
    ExecutionGroupProxy egProxy = brokerProxy.getExecutionGroupByName(replayEG);

    // Submit request to EG to actually do replay
    DataCaptureProxy dataCaptureProxy = egProxy.getDataCapture(dataCaptureStore, dataCaptureEntry);
    dataCaptureProxy.hasBeenPopulatedByBroker(true);

    String responseBody = dataCaptureProxy.getDataCaptureEntryAsXml(1);

} catch (Exception e) {
    // TODO: Handle Exception
} finally {

```

```
if (brokerProxy != null) {  
    brokerProxy.disconnect();  
}  
}
```

Related concepts:

“Record and replay” on page 597

For audit or problem determination purposes, you can record data to a database, then view it, and replay it.

Related tasks:

“Developing applications that use the IBM Integration API” on page 58

Develop Java applications that use the IBM Integration API (also known as the CMP) to communicate with, deploy to, and manage brokers and their associated resources.

Related information:



IBM Integration API (CMP)

Submitting batch requests from a CMP application

Use the CMP to group multiple requests that are destined for the same broker, and submit them as a single unit of work.

About this task

To start a batch, your application must call the `beginUpdates()` method on the `BrokerProxy` handle. The CMP delays submitting any state-changing requests to the broker until it is told a batch of requests is ready to be sent.

The `sendUpdates()` method tells the CMP to submit as a batch all requests received since the last `beginUpdates()` call. The `clearUpdates()` method can be used to discard a batch without submitting it to the broker. The application can check if a batch is currently in progress by using the `isBatching()` method. Only one batch for an CMP handle can be in progress at any one time.

One advantage of using a batch method is that it provides an assurance that no other applications can have messages processed by the broker during the batch. When a broker receives a batch of requests, it processes each request in the batch in the order it was added to the batch (FIFO), and requests from no other CMP application are processed until the entire batch is completed.

Consider the following sequence of commands:

```
ExecutionGroupProxy e = b.createExecutionGroup("EG2");  
e.deploy("mybar.bar");
```

Without using a batch method, the application cannot guarantee the success of these actions. For example, even if each command would otherwise succeed, a second (possibly remote) application might delete the integration server EG2 after it has been created by the first application, but before the other command is processed.

If the sequence is extended to use a batch method, the broker is now guaranteed to process all the commands together, therefore no other application can disrupt the logic intended by the application.

```
b.startUpdates();
ExecutionGroupProxy e = b.createExecutionGroup("EG2");
e.deploy("mybar.bar");
b.sendUpdates();
```

Another advantage of using a batch method is performance. The CMP typically sends one WebSphere MQ message to the broker for each request.

In a situation that requires lots of requests to be sent in quick succession, the use of a batch has a significant effect on performance, reducing both time taken to process the requests, and the memory used. For example, your application might create a number of integration servers on a single broker. Each batch of requests is sent in a single WebSphere MQ message, therefore reducing the processing that is required for each method.

Batch mode does not provide transactional (commit and backout) capability; some requests in a batch might succeed and others fail. If the broker processes a request in a batch that fails, it continues to process the next request in the batch until it has attempted all requests in the batch.

Related concepts:



The IBM Integration API

The IBM Integration API is a programming interface that your applications can use to control brokers and their resources through a remote interface.

Related tasks:

“Configuring an environment for developing and running CMP applications” on page 72

Prepare the environment in which you want to run your CMP applications.

“Connecting to a broker from a CMP application” on page 79

Connect an application that uses the CMP to a broker, to send requests about its status and its resources.

Related information:



IBM Integration API (CMP)

Examples of IBM Integration API code

Use the following examples of IBM Integration API code for common tasks to help you write your own Java code to develop message flow applications.

About this task

The following list shows examples of Java code that you can use to complete common tasks when you are developing message flow applications:

Procedure

- “Loading an existing message flow into memory” on page 111
- “Renaming a node” on page 112
- “Adding a node and a subflow node” on page 113
- “Setting the position of a node” on page 114
- “Copying a node” on page 115
- “Removing a node” on page 116
- “Adding connections between nodes” on page 117
- “Adding and connecting user-defined nodes” on page 118

- “Removing connections between nodes” on page 120
- “Creating or changing user-defined properties” on page 121
- “Creating ESQL modules” on page 122
- “Renaming a message flow” on page 123
- “Updating filter tables on Route nodes” on page 124
- “Saving a message flow file from memory” on page 125

Related concepts:

 User-defined patterns

A *user-defined pattern* extends the function of IBM Integration Bus so that you are able to create patterns that you can reuse within your organization.

Related tasks:

 Creating a pattern authoring project

Create a pattern authoring project and choose an exemplar for the project.

 Extending a user-defined pattern

Extend a user-defined pattern to customize its behavior so that it is easier to use or to provide guidance to pattern users.

 Modifying pattern instances by using the IBM Integration API

Add Java code to a code plug-in project to modify a pattern instance when the pattern instance is generated by a pattern user.

 Testing Java code

After writing Java code to modify pattern instances, test the code to check that it works correctly.

Related reference:

 IBM Integration API

Use the IBM Integration API to write Java code to modify user-defined pattern instances or to develop message flow applications.

Loading an existing message flow into memory

Use the IBM Integration API when developing message flow applications to load a message flow into memory. You can then access the message flow in your Java code.

About this task

You must load a message flow into memory to use it with the IBM Integration API methods within your Java code. To load a message flow into memory, create a File object and use the read() method of the FlowRendererMSGFLOW, which makes the message flow available for your Java code. The read() method takes the message flow project containing the required message flow file and the relative path to the message flow file from this project.

Procedure

The following example shows how to load a message flow that is in the mqsi directory:

```
import java.io.File;
import java.io.IOException;
import com.ibm.broker.MessageBrokerAPIException;
```

```

import com.ibm.broker.config.appdev.MessageFlow;
import com.ibm.broker.config.appdev.FlowRendererMSGFLOW;

public class LoadMessageFlow {
    public static void main(String[] args) {
        File msgFlow = new File("../mqsi/main.msgflow");
        try {
            MessageFlow mf1 = FlowRendererMSGFLOW.read(msgFlow);
        } catch (IOException e) {
            // Add your own code here
            e.printStackTrace();
        } catch (MessageBrokerAPIException e) {
            // Add your own code here
            e.printStackTrace();
        }
    }
}

```

Results

Result:

You can now refer to the instance of the MessageFlow object in your Java code.

Related tasks:

“Examples of IBM Integration API code” on page 110

Use the following examples of IBM Integration API code for common tasks to help you write your own Java code to develop message flow applications.

“Saving a message flow file from memory” on page 125

Use the IBM Integration API when developing message flow applications to save a message flow from memory.

Renaming a node

Use the IBM Integration API when developing message flow applications to rename a node.

About this task

To rename a node, you must first access the required node. You can access the node by using the existing name of the node and the `getNodeByName()` method. You can then rename the node by using the `setNodeName()` method, which takes the new node name as a parameter. For example:

Example

```

File msgFlow = new File("main.msgflow");
MessageFlow mf1 = FlowRendererMSGFLOW.read(msgFlow);
Node mqinNode = mf1.getNodeByName("My Input Node");
mqinNode.setNodeName("New Input Node");

```

Related tasks:

“Examples of IBM Integration API code” on page 110

Use the following examples of IBM Integration API code for common tasks to help you write your own Java code to develop message flow applications.

“Adding a node and a subflow node” on page 113

Use the IBM Integration API when developing message flow applications to add a new node or subflow node to a message flow.

“Renaming a message flow” on page 123

Use the IBM Integration API when developing message flow applications to rename message flows.



Testing Java code

After writing Java code to modify pattern instances, test the code to check that it works correctly.

Related reference:



IBM Integration API

Use the IBM Integration API to write Java code to modify user-defined pattern instances or to develop message flow applications.

Adding a node and a subflow node

Use the IBM Integration API when developing message flow applications to add a new node or subflow node to a message flow.

About this task

You can add a new built-in node, or a new subflow node to a message flow.

- When you add a subflow node by using the IBM Integration API, you must link the subflow node with the subflow message flow by using the `setSubFlow()` method of the subflow node object. For example, if you have assigned your subflow message flow to message flow instance *sub1* and you have assigned your subflow node to subflow node instance *sfNode*, you must use the following statement to link the subflow node with the subflow message flow:

```
sfNode.setSubFlow(sub1);
```

- If you want to set a node property on a subflow node, use the `addNodeProperty()` method to set the property before you link the node to the subflow by using the `setSubFlow()` method. If you use `addNodeProperty()` after `setSubFlow()`, the node property might not be stored.

Procedure

- The following example shows you how to add a new built-in node:

```
File msgFlow = new File("main.msgflow");
MessageFlow mf1 = FlowRendererMSGFLOW.read(msgFlow);
MQInputNode mqinNode = new MQInputNode();
mqinNode.setNodeName("My Input Node");
mqinNode.setQueueName("INPUTQ");
mf1.addNode(mqinNode);
```

- The following example shows you how to add a new subflow node to a message flow:

1. A new subflow node is created and assigned to object *sfNode*.
2. The subflow node name is set to *My Sub Flow Node*.
3. The subflow node is linked to the subflow message flow by using the `setSubFlow()` method.
4. The new subflow node is added to the message flow held in object *mf1*.

- The subflow can be stored in a `.msgflow` format:

```
File msgFlow = new File("main.msgflow");
MessageFlow mf1 = FlowRendererMSGFLOW.read(msgFlow);
File subFlow = new File("subflow.msgflow");
MessageFlow sub1 = FlowRendererMSGFLOW.read(subFlow);
```

```

SubFlowNode sfNode = new SubFlowNode();
sfNode.setNodeName("My Sub Flow Node");
sfNode.setSubFlow(sub1);
mf1.addNode(sfNode);

```

- The subflow can be stored in a .subflow format:

```

File msgFlow = new File("main.msgflow");
MessageFlow mf1 = FlowRendererMSGFLOW.read(msgFlow);
File subFlow = new File("subflow.subflow");
MessageFlow sub1 = FlowRendererMSGFLOW.read(subFlow);
SubFlowNode sfNode = new SubFlowNode();
sfNode.setNodeName("My Sub Flow Node");
sfNode.setSubFlow(sub1);
mf1.addNode(sfNode);

```

Related tasks:

“Examples of IBM Integration API code” on page 110

Use the following examples of IBM Integration API code for common tasks to help you write your own Java code to develop message flow applications.

“Renaming a node” on page 112

Use the IBM Integration API when developing message flow applications to rename a node.

“Setting the position of a node”

Use the IBM Integration API when developing message flow applications to set the position of a node on the canvas in the Application Development view.

“Copying a node” on page 115

Use the IBM Integration API when developing message flow applications to copy a node or subflow node to a message flow.

“Removing a node” on page 116

Use the IBM Integration API when developing message flow applications to remove a node from a message flow.

“Adding connections between nodes” on page 117

Use the IBM Integration API when developing message flow applications to add connections between nodes.

“Adding and connecting user-defined nodes” on page 118

Use the IBM Integration API when developing message flow applications to add user-defined nodes and to connect user-defined nodes to other nodes.

“Removing connections between nodes” on page 120

Use the IBM Integration API when developing message flow applications to remove connections between nodes.

Related reference:



IBM Integration API

Use the IBM Integration API to write Java code to modify user-defined pattern instances or to develop message flow applications.

Setting the position of a node

Use the IBM Integration API when developing message flow applications to set the position of a node on the canvas in the Application Development view.

About this task

You can set the position of a node on the canvas by using the setLocation() method of the node object. The following example sets the position of a new MQOutput node to coordinates x=300 pixels, y=100 pixels:

Example

```
File msgFlow = new File("main.msgflow");
MessageFlow mf1 = FlowRendererMSGFLOW.read(msgFlow);
MQOutputNode mqoutNode = new MQOutputNode();
mqoutNode.setLocation(300, 100);
```

Related tasks:

“Examples of IBM Integration API code” on page 110

Use the following examples of IBM Integration API code for common tasks to help you write your own Java code to develop message flow applications.

“Adding a node and a subflow node” on page 113

Use the IBM Integration API when developing message flow applications to add a new node or subflow node to a message flow.

“Copying a node”

Use the IBM Integration API when developing message flow applications to copy a node or subflow node to a message flow.

“Removing a node” on page 116

Use the IBM Integration API when developing message flow applications to remove a node from a message flow.

“Renaming a message flow” on page 123

Use the IBM Integration API when developing message flow applications to rename message flows.

Related reference:



IBM Integration API

Use the IBM Integration API to write Java code to modify user-defined pattern instances or to develop message flow applications.

Copying a node

Use the IBM Integration API when developing message flow applications to copy a node or subflow node to a message flow.

About this task

You can copy a built-in or subflow node by using the `clone()` method. In the following example, a new `MQInput` node `mqinNode` is created and properties on the node are set. A new `MQInput` node `mqinNode1` is then created by copying `mqinNode` by using the `clone()` method. When the node is copied, the node properties are also copied:

Example

```
File msgFlow = new File("main.msgflow");
MessageFlow mf1 = FlowRendererMSGFLOW.read(msgFlow);
MQInputNode mqinNode = new MQInputNode();
mqinNode.setNodeName("My Input Node");
mqinNode.setQueueName("INPUTQ");
MQInputNode mqinNode1 = (MQInputNode) mqinNode.clone();
mqinNode1.setNodeName("Copy of My Input Node");
mf1.addNode(mqinNode1);
```

Related tasks:

“Examples of IBM Integration API code” on page 110

Use the following examples of IBM Integration API code for common tasks to help you write your own Java code to develop message flow applications.

“Renaming a node” on page 112

Use the IBM Integration API when developing message flow applications to

rename a node.

“Adding a node and a subflow node” on page 113

Use the IBM Integration API when developing message flow applications to add a new node or subflow node to a message flow.

“Setting the position of a node” on page 114

Use the IBM Integration API when developing message flow applications to set the position of a node on the canvas in the Application Development view.

“Removing a node”

Use the IBM Integration API when developing message flow applications to remove a node from a message flow.

“Adding connections between nodes” on page 117

Use the IBM Integration API when developing message flow applications to add connections between nodes.

“Adding and connecting user-defined nodes” on page 118

Use the IBM Integration API when developing message flow applications to add user-defined nodes and to connect user-defined nodes to other nodes.

“Removing connections between nodes” on page 120

Use the IBM Integration API when developing message flow applications to remove connections between nodes.

Related reference:



IBM Integration API

Use the IBM Integration API to write Java code to modify user-defined pattern instances or to develop message flow applications.

Removing a node

Use the IBM Integration API when developing message flow applications to remove a node from a message flow.

About this task

To remove a node you must first get the required node from the message flow object. In the following example, the `getNodeByName()` method is used to get the required node from message flow object `mf1`. The node is then removed by using the `removeNode()` method. When a node is removed, any connections to or from the node are also removed:

Example

```
File msgFlow = new File("main.msgflow");
MessageFlow mf1 = FlowRendererMSGFLOW.read(msgFlow);
Node mqinNode = mf1.getNodeByName("My Input Node");
mf1.removeNode(mqinNode);
```

Related tasks:

“Examples of IBM Integration API code” on page 110

Use the following examples of IBM Integration API code for common tasks to help you write your own Java code to develop message flow applications.

“Adding a node and a subflow node” on page 113

Use the IBM Integration API when developing message flow applications to add a new node or subflow node to a message flow.

“Removing connections between nodes” on page 120

Use the IBM Integration API when developing message flow applications to remove connections between nodes.

Related reference:



IBM Integration API

Use the IBM Integration API to write Java code to modify user-defined pattern instances or to develop message flow applications.

Adding connections between nodes

Use the IBM Integration API when developing message flow applications to add connections between nodes.

About this task

You can connect both subflow and built-in nodes. The first example shows you how to connect two built-in nodes. The second example shows you how to connect a built-in node to a subflow node.

To use the terminals of a subflow node, you must link the subflow node with the subflow message flow by using the `setSubFlow()` method of the subflow node object. For example, if you have assigned your subflow message flow to message flow instance `sub1` and you have assigned your subflow node to subflow node instance `sfNode`, you must use the following statement to link the subflow node with the subflow message flow:

```
sfNode.setSubFlow(sub1);
```

Procedure

- The following example shows you how to connect two built-in nodes:
 1. A `MQInput` node and a `Collector` node are created.
 2. The `getInputTerminal()` method is used to create a dynamic Input terminal called `NEWIN` on the `Collector` node.
 3. The Input terminal is connected to the Output terminal of the `MQInput` node by using the `connect()` method of the message flow object `mf1`.

```

File msgFlow = new File("main.msgflow");
MessageFlow mf1 = FlowRendererMSGFLOW.read(msgFlow);
MQInputNode mqinNode = new MQInputNode();
mqinNode.setNodeName("My Input Node");
mqinNode.setQueueName("INPUTQ");
CollectorNode colNode = new CollectorNode();
colNode.getInputTerminal("NEWIN");
mf1.connect(mqinNode.OUTPUT_TERMINAL_OUT, colNode.getInputTerminal("NEWIN"));

```

To use a static terminal on a node you must use the appropriate constant defined for it in the IBM Integration API. These constants are listed in the IBM Integration API reference, see IBM Integration API.

- The following example shows you how to connect a subflow node to a built-in node. You must load the subflow message flow and link it to a subflow node. You must use the `getInputTerminal()`, `getInputTerminals()`, `getOutputTerminal()` or `getOutputTerminals()` method to access the terminal on the subflow node to which you want to connect. The example code completes the following steps:
 1. The main message flow, `main.msgflow`, and a `Compute` node in the main message flow are loaded into memory.
 2. The subflow message flow, `subflow.msgflow`, and the subflow node in the main message flow are loaded into memory.
 3. The `setSubFlow()` method of the subflow node is used to link the subflow message flow `sub1` to the subflow node `sfNode`.

4. The `getOutputTerminal()` method is used to get the Process terminal of the subflow node. The `connect()` method of the message flow object is used to connect this terminal to the Input terminal of the Compute node.

```
File msgflow = new File("main.msgflow");
MessageFlow mfl = FlowRendererMSGFLOW.read(msgflow);
ComputeNode computeNode = (ComputeNode)mfl.getNodeByName("My Compute Node");
File subflow = new File("subflow.msgflow");
MessageFlow subfl = FlowRendererMSGFLOW.read(subflow);
SubFlowNode sflNode = (SubFlowNode)mfl1.getNodeByName("My SubFlow Node");
sflNode.setSubFlow(subfl);
mfl1.connect(sflNode.getOutputTerminal("Process"), computeNode.INPUT_TERMINAL_IN);
```

Related tasks:

“Examples of IBM Integration API code” on page 110

Use the following examples of IBM Integration API code for common tasks to help you write your own Java code to develop message flow applications.

“Adding a node and a subflow node” on page 113

Use the IBM Integration API when developing message flow applications to add a new node or subflow node to a message flow.

“Setting the position of a node” on page 114

Use the IBM Integration API when developing message flow applications to set the position of a node on the canvas in the Application Development view.

“Adding and connecting user-defined nodes”

Use the IBM Integration API when developing message flow applications to add user-defined nodes and to connect user-defined nodes to other nodes.

“Removing connections between nodes” on page 120

Use the IBM Integration API when developing message flow applications to remove connections between nodes.

“Renaming a message flow” on page 123

Use the IBM Integration API when developing message flow applications to rename message flows.

Related reference:



IBM Integration API

Use the IBM Integration API to write Java code to modify user-defined pattern instances or to develop message flow applications.

Adding and connecting user-defined nodes

Use the IBM Integration API when developing message flow applications to add user-defined nodes and to connect user-defined nodes to other nodes.

About this task

The code required to add and connect user-defined node is different to the code required for built-in nodes and subflow nodes.

When you write Java code for user-defined nodes you must be aware of the following information:

- User-defined nodes are supported by instances of the `GenericNode` class. To add user-defined nodes to message flows, create instances of `GenericNode` and add them to the message flow instance.
- To retrieve existing instances of a user-defined node, call `getNodeByName()` and cast the returned object to a `GenericNode` object.
- The terminals defined on your user-defined nodes are not automatically available in the API. If you create an instance of a `GenericNode` class, it does not have any input or output terminals listed. The methods `getInputTerminals()` and `getOutputTerminals()` return empty lists.
- To get an input terminal for a `GenericNode`, call `getInputTerminal()` and pass the terminal name that exists on the generic node. This method returns the input

terminal and makes it available in the message flow object that contains your generic node. After you have used `getInputTerminal()` with a known terminal name, this input terminal is returned if `getInputTerminals()` is used.

- To get an output terminal for a `GenericNode`, call `getOutputTerminal()` and pass the terminal name that exists on the generic node. This method returns the output terminal and makes it available in the message flow object that contains your generic node. After you have used `getOutputTerminal()` with a known terminal name, this output terminal is returned if `getOutputTerminals()` is used.

Example

The following example shows how you can add a user-defined node to a message flow and connect it to a built-in node:

1. An `MQInput` node is created and added to the message flow.
2. A user-defined node is created by using the `GenericNode` class and is added to the message flow object.
3. The static output terminal of the `MQInput` is assigned to the variable `outputTerminal`.
4. The input terminal of the user-defined node is assigned to the variable `inputTerminal` by using the `getInputTerminal()` method with the known terminal name `In`.
5. The nodes are connected by using the `connect()` method.
6. The final section of code shows that the input node is now available for use in the message flow, by using the `getInputTerminals()` method of the user-defined node instance.

```
File msgFlow = new File("main.msgflow");
MessageFlow mf1 = FlowRendererMSGFLOW.read(msgFlow);

MQInputNode mqinNode = new MQInputNode();
mqinNode.setNodeName("My Input Node");
mqinNode.setQueueName("IN");
mf1.addNode(mqinNode);

GenericNode myNode = new GenericNode("MyUserDefinedNode");
myNode.setNodeName("MyNode");
mf1.addNode(myNode);

OutputTerminal outputTerminal = mqinNode.OUTPUT_TERMINAL_OUT;
InputTerminal inputTerminal = myNode.getInputTerminal("In");
mf1.connect(outputTerminal, inputTerminal);

InputTerminal[] inputTerminals = myNode.getInputTerminals();
System.out.println("Input terminals on my node:");
for (int i = 0; i < inputTerminals.length; i++) {
    InputTerminal inputTerminal = inputTerminals[i];
    System.out.println(inputTerminal.getName());
}
```

Related tasks:

“Examples of IBM Integration API code” on page 110

Use the following examples of IBM Integration API code for common tasks to help you write your own Java code to develop message flow applications.

“Renaming a node” on page 112

Use the IBM Integration API when developing message flow applications to rename a node.

“Adding a node and a subflow node” on page 113

Use the IBM Integration API when developing message flow applications to add a new node or subflow node to a message flow.

“Setting the position of a node” on page 114

Use the IBM Integration API when developing message flow applications to set the position of a node on the canvas in the Application Development view.

“Copying a node” on page 115

Use the IBM Integration API when developing message flow applications to copy a node or subflow node to a message flow.

“Removing a node” on page 116

Use the IBM Integration API when developing message flow applications to remove a node from a message flow.

“Adding connections between nodes” on page 117

Use the IBM Integration API when developing message flow applications to add connections between nodes.

“Removing connections between nodes”

Use the IBM Integration API when developing message flow applications to remove connections between nodes.

“Creating or changing user-defined properties” on page 121

Use the IBM Integration API when developing message flow applications to create or change user-defined properties (UDPs).

Related reference:



IBM Integration API

Use the IBM Integration API to write Java code to modify user-defined pattern instances or to develop message flow applications.

Removing connections between nodes

Use the IBM Integration API when developing message flow applications to remove connections between nodes.

About this task

You can disconnect two nodes by using the `disconnect()` method of the message flow object. You must provide this method with the names of the terminal instances that you want to disconnect. For example:

Example

```
File msgFlow = new File("main.msgflow");
MessageFlow mf1 = FlowRendererMSGFLOW.read(msgFlow);
MQInputNode mqinNode = (MQInputNode)mf1.getNodeByName("My Input Node");
MQOutputNode mqoutNode = (MQOutputNode)mf1.getNodeByName("My Output Node");
mf1.disconnect(mqinNode.OUTPUT_TERMINAL_OUT, mqoutNode.INPUT_TERMINAL_IN);
```

Related tasks:

“Examples of IBM Integration API code” on page 110

Use the following examples of IBM Integration API code for common tasks to help you write your own Java code to develop message flow applications.

“Adding a node and a subflow node” on page 113

Use the IBM Integration API when developing message flow applications to add a new node or subflow node to a message flow.

“Adding and connecting user-defined nodes” on page 118

Use the IBM Integration API when developing message flow applications to add user-defined nodes and to connect user-defined nodes to other nodes.

Testing Java code

After writing Java code to modify pattern instances, test the code to check that it works correctly.

Related reference:

IBM Integration API

Use the IBM Integration API to write Java code to modify user-defined pattern instances or to develop message flow applications.

Creating or changing user-defined properties

Use the IBM Integration API when developing message flow applications to create or change user-defined properties (UDPs).

About this task

You can create new UDPs and add them to the message flow, or you can discover existing UDPs and modify them.

Procedure

- The following example shows you how to create a UDP and add it to a message flow:
 1. A UDP called Property1 is created in parameter group Group1. The data type of the UDP is defined as a string and the UDP is given the default value Hello World!
 2. The UDP is then added to the message flow by using the `addFlowProperty()` method.

```
File msgFlow = new File("main.msgflow");
MessageFlow mf1 = FlowRendererMSGFLOW.read(msgFlow);
UserDefinedProperty udp = new UserDefinedProperty("Group1", "Property1", UserDefinedProperty.Usage.MANDATORY, UserDefinedProperty.Type.STRING, "Hello World!");
mf1.addFlowProperty(udp);
```

- In the following example, the existing UDPs in a message flow are discovered by using the `getFlowProperties()` method on the message flow. The `setName()` method is then used to set the name of the first UDP to Property3:

```
File msgFlow = new File("main.msgflow");
MessageFlow mf1 = FlowRendererMSGFLOW.read(msgFlow);
Vector<FlowProperty> flowProperties = mf1.getFlowProperties();
flowProperties.get(0).setName("Property3");
```

Related tasks:

“Examples of IBM Integration API code” on page 110

Use the following examples of IBM Integration API code for common tasks to help you write your own Java code to develop message flow applications.

“Copying a node” on page 115

Use the IBM Integration API when developing message flow applications to copy a node or subflow node to a message flow.

“Removing a node” on page 116

Use the IBM Integration API when developing message flow applications to remove a node from a message flow.

Related reference:

IBM Integration API

Use the IBM Integration API to write Java code to modify user-defined pattern instances or to develop message flow applications.

Creating ESQL modules

Use the IBM Integration API when developing message flow applications to create ESQL modules. You can then associate ESQL modules with nodes that use ESQL. For example, if you create a Compute node, you can write Java code to create an ESQL module and then associate that module with the node.

About this task

Examples of nodes that use ESQL are Compute, Database, DatabaseInput, and Filter nodes. If you are using a non-default broker schema, you must set the schema by using the `setBrokerSchema()` method.

Procedure

- The following example shows you how to create an ESQL module in the `mqs i` schema. The module is then assigned to a new Compute node by using the `setComputeExpression()` method:

```
File msgFlow = new File("main.msgflow");
MessageFlow mf1 = FlowRendererMSGFLOW.read(msgFlow);
ESQLModule module = new ESQLModule();
module.setBrokerSchema("mqs i");
module.setEsqMain("MyESQLMain");
ComputeNode compNode = new ComputeNode();
compNode.setNodeName("My Compute Node");
compNode.setComputeExpression(module);
mf1.addNode(compNode);
```

- The following example shows you how to create an ESQL module in the default schema. The `setBrokerSchema()` method is not required.

```
File msgFlow = new File("main.msgflow");
MessageFlow mf1 = FlowRendererMSGFLOW.read(msgFlow);
ESQLModule module = new ESQLModule();
module.setEsqMain("MyESQLMain");
ComputeNode compNode = new ComputeNode();
compNode.setNodeName("My Compute Node");
compNode.setComputeExpression(module);
mf1.addNode(compNode);
```

- The following example shows you how to discover an ESQL module from within an ESQL file by using the `getEsqModules()` method. You can then use the ESQL module to set the compute expression on a Compute node by using the `setComputeExpression()` method.

```
File msgFlow = new File("main.msgflow");
MessageFlow mf1 = FlowRendererMSGFLOW.read(msgFlow);
File esql = new File("FileBatchProcessingSample_Branch.esql");
ESQLFile esqlFile = new ESQLFile(esql);
Vector<ESQLModule> esqlModules = esqlFile.getEsqModules();
ComputeNode compNode = new ComputeNode();
compNode.setNodeName("My Compute Node");
compNode.setComputeExpression(esqlModules.get(0));
mf1.addNode(compNode);
```

Related tasks:

“Examples of IBM Integration API code” on page 110

Use the following examples of IBM Integration API code for common tasks to help you write your own Java code to develop message flow applications.

“Loading an existing message flow into memory” on page 111

Use the IBM Integration API when developing message flow applications to load a message flow into memory. You can then access the message flow in your Java

code.

“Renaming a node” on page 112

Use the IBM Integration API when developing message flow applications to rename a node.

“Adding a node and a subflow node” on page 113

Use the IBM Integration API when developing message flow applications to add a new node or subflow node to a message flow.

“Setting the position of a node” on page 114

Use the IBM Integration API when developing message flow applications to set the position of a node on the canvas in the Application Development view.

“Copying a node” on page 115

Use the IBM Integration API when developing message flow applications to copy a node or subflow node to a message flow.

“Removing a node” on page 116

Use the IBM Integration API when developing message flow applications to remove a node from a message flow.

“Adding connections between nodes” on page 117

Use the IBM Integration API when developing message flow applications to add connections between nodes.

“Adding and connecting user-defined nodes” on page 118

Use the IBM Integration API when developing message flow applications to add user-defined nodes and to connect user-defined nodes to other nodes.

“Removing connections between nodes” on page 120

Use the IBM Integration API when developing message flow applications to remove connections between nodes.

“Creating or changing user-defined properties” on page 121

Use the IBM Integration API when developing message flow applications to create or change user-defined properties (UDPs).

“Renaming a message flow”

Use the IBM Integration API when developing message flow applications to rename message flows.

“Updating filter tables on Route nodes” on page 124

Use the IBM Integration API when developing message flow applications to update filter tables on Route nodes.

Related reference:



IBM Integration API

Use the IBM Integration API to write Java code to modify user-defined pattern instances or to develop message flow applications.

Renaming a message flow

Use the IBM Integration API when developing message flow applications to rename message flows.

About this task

You can write code to rename a message flow by using the `setName()` method. In the following example, a message flow file called `main.msgflow` is renamed to `mainGenerated.msgflow`:

Example

```
File msgFlow = new File("main.msgflow");
MessageFlow mf1 = FlowRendererMSGFLOW.read(msgFlow);
mf1.setName(mf1.getName()+"Generated");
```

Related tasks:

“Examples of IBM Integration API code” on page 110

Use the following examples of IBM Integration API code for common tasks to help you write your own Java code to develop message flow applications.

“Loading an existing message flow into memory” on page 111

Use the IBM Integration API when developing message flow applications to load a message flow into memory. You can then access the message flow in your Java code.

Related reference:



IBM Integration API

Use the IBM Integration API to write Java code to modify user-defined pattern instances or to develop message flow applications.

Updating filter tables on Route nodes

Use the IBM Integration API when developing message flow applications to update filter tables on Route nodes.

About this task

You can add and update rows on the filter table of a Route node.

Adding a new row

The following example shows you how to add a new row to a filter table by using the `createRow()` method:

1. The message flow and the Route node are loaded into memory.
2. The filter table of the Route node is loaded into memory by using the `getFilterTable()` method of the `RouteNode` object.
3. A new filter table row is created by using the `createRow()` method.
4. The value of the filter pattern property on this new row is set to `value="123"` by using the `setFilterPattern()` method.
5. The routing output terminal property is set to `NEWOUT` by using the `setRoutingOutputTerminal()` method.
6. The new row is then added to the filter table by using the `addRow()` method.

```
File msgFlow = new File("main.msgflow");
MessageFlow mf1 = FlowRendererMSGFLOW.read(msgFlow);
RouteNode routeNode = (RouteNode)mf1.getNodeByName("My Route Node");
RouteNode.FilterTable filterTable = (RouteNode.FilterTable)routeNode.getFilterTable();
RouteNode.FilterTableRow newRow = filterTable.createRow();
newRow.setFilterPattern("value=\"123\"");
newRow.setRoutingOutputTerminal("NEWOUT");
filterTable.addRow(newRow);
```

Updating a row

The following example shows you how to update rows on the filter table of a Route node.

1. The message flow, Route node, and filter table of the Route node are loaded into memory.
2. The rows of the filter table are loaded into memory by using the `getRows()` method.

- The filter pattern property of the first row of the filter table is set to `value2="456"`.
- The routing output terminal property of the first row of the filter table is set to `NEWOUT2`.

```
File msgFlow = new File("main.msgflow");
MessageFlow mf1 = FlowRendererMSGFLOW.read(msgFlow);
RouteNode routeNode = (RouteNode)mf1.getNodeByName("My Route Node");
RouteNode.FilterTable filterTable = (RouteNode.FilterTable)routeNode.getFilterTable();
Vector<RouteNode.FilterTableRow> filterTableRows = filterTable.getRows();
filterTableRows.get(0).setFilterPattern("value2=\"456\"");
filterTableRows.get(0).setRoutingOutputTerminal("NEWOUT2");
```

Related tasks:

“Examples of IBM Integration API code” on page 110

Use the following examples of IBM Integration API code for common tasks to help you write your own Java code to develop message flow applications.

“Loading an existing message flow into memory” on page 111

Use the IBM Integration API when developing message flow applications to load a message flow into memory. You can then access the message flow in your Java code.

Related reference:



IBM Integration API

Use the IBM Integration API to write Java code to modify user-defined pattern instances or to develop message flow applications.

Saving a message flow file from memory

Use the IBM Integration API when developing message flow applications to save a message flow from memory.

About this task

Save a message flow to a message flow file to preserve the additions and modifications that you have made. To save a message flow from memory, use the `write()` method of the `FlowRendererMSGFLOW` object. The `write()` method writes all the changes made using the IBM Integration API to the specified message flow file.

The schema for a message flow is stored in the message flow and cannot be modified by using the IBM Integration API. When you save your message flow file by using the IBM Integration API, the file is saved to a directory that has the same name as the schema. If the directory does not exist, it is created. For example, if your message flow is part of the `mqsi` schema, when you save the message flow file it is saved to the `mqsi` directory.

Procedure

The following example shows how to save a message flow in the specified directory by using the `write()` method. On Windows, if the message flow is part of the `mqsi` schema, the message flow file is saved to `C:\MB\mqsi\main.msgflow`.

```
File msgFlow = new File("main.msgflow");
MessageFlow mf1 = FlowRendererMSGFLOW.read(msgFlow);
FlowRendererMSGFLOW.write(mf1, "C:\MB");
```

Related tasks:

“Examples of IBM Integration API code” on page 110

Use the following examples of IBM Integration API code for common tasks to help you write your own Java code to develop message flow applications.

“Loading an existing message flow into memory” on page 111

Use the IBM Integration API when developing message flow applications to load a message flow into memory. You can then access the message flow in your Java code.

Related reference:

 IBM Integration API

Use the IBM Integration API to write Java code to modify user-defined pattern instances or to develop message flow applications.

Starting, stopping, or quiescing TCP/IP connections

Use the IBM Integration API to start, stop, or quiesce server connections that use a TCP/IP configurable service.

About this task

You can call the TCP/IP resource manager to start, stop, or quiesce a TCP/IP connection manager.

Procedure

- The following example shows how to start a connection manager:

```
ResourceManagerProxy rm = EG.getResourceManagerByName("TCPIP");
Properties prop1 = new Properties(); prop1.setProperty("port","1445");
rm.execute("start", prop1,"Server",null);
```


- The following example shows how to stop a connection manager. Stopping a connection manager forcibly stops all connections, and no new connections are made.

```
ResourceManagerProxy rm = EG.getResourceManagerByName("TCPIP");
Properties prop1 = new Properties(); prop1.setProperty("port","1445");
rm.execute("stop", prop1,"Server",null);
```

- The following example shows how to quiesce a connection manager. Quiescing stops new connections from being made. You can set a timeout value for the quiesce by using the **quiesceTimeoutSec** property. If the quiesce is not completed before the value that is set on **quiesceTimeoutSec**, then the connections are forcibly closed.

```
ResourceManagerProxy rm = EG.getResourceManagerByName("TCPIP");
Properties prop1 = new Properties(); prop1.setProperty("port","1445");
prop1.setProperty("quiesceTimeoutSec","30");
rm.execute("quiesce",
prop1,"Server",null);
```

Related concepts:

 Connection management

TCP/IP connections are requested by the client connection manager and accepted by the server connection manager.

Related tasks:

“Examples of IBM Integration API code” on page 110

Use the following examples of IBM Integration API code for common tasks to help you write your own Java code to develop message flow applications.

Related reference:

 IBM Integration API

Use the IBM Integration API (CMP) Java classes and methods to develop CMP applications.

Administering brokers using the web user interface

You can use the IBM Integration Bus web user interface to administer broker resources.

Before you begin

Before you start:

Read the following topics:

- Chapter 1, “Administering brokers and broker resources,” on page 1
- “Role-based security” on page 220

About this task

The IBM Integration Bus web user interface enables web users to access broker resources through an HTTP client, and provides broker administrators with an alternative to the IBM Integration Explorer for administering broker resources.

By granting permissions to the roles that are associated with web users, broker administrators can allow those users to start and stop integration servers, applications, and message flows from the web user interface.

Support is provided for most major browsers; for more information, see the IBM Support site.

Complete the following steps to configure your web user interface server, create the required web user accounts, and log on to the web user interface:

Procedure

1. Configure a web user interface server. For information on how to do this, see “Configuring the web user interface server” on page 128.
2. Use the `mqswebuseradmin` command to create web user accounts for your web users. For information on how to do this, see “Managing web user accounts” on page 226.
3. Log on to the web user interface server through the web user interface. For information on how to do this, see “Accessing the web user interface” on page 133.
4. You can now administer your broker resources through the web user interface.

Results

The web user interface is enabled by default for all new brokers when they are created. However, you can disable and enable the web user interface at any time; for more information, see “Enabling and disabling the web user interface” on page 131.

If broker administration security is enabled, you can restrict web users' access to data and broker resources. For more information, see “Controlling access to data and resources in the web user interface” on page 239.

Related concepts:

“Role-based security” on page 220

You can control access to broker resources by associating web users with roles.



The IBM Integration API

The IBM Integration API is a programming interface that your applications can use to control brokers and their resources through a remote interface.

Related tasks:

“Accessing the web user interface” on page 133

You can access broker resources by logging on to the web user interface.

“Managing web user accounts” on page 226

You can control a web user's access to broker resources by associating the web user ID with a role, which has security permissions assigned to it.

“Enabling and disabling the web user interface” on page 131

You can enable and disable the web user interface by using the **mqsichangeproperties** command.

“Recording, viewing, and replaying data” on page 595

Record data that is processed by a message flow, or emitted from WebSphere Application Server. View and replay the data.

“Enabling security for record and replay” on page 598

You can restrict the users who can view and replay data for a broker by enabling security.

“Controlling access to data and resources in the web user interface” on page 239

Broker administrators can control web users' access to data and broker resources by assigning permissions to users based on their role.

Chapter 1, “Administering brokers and broker resources,” on page 1

Administering brokers and associated integration node resources includes the tasks that you perform frequently to activate and manage those resources. Choose the method you prefer to administer your brokers and associated resources.

“Starting collection of accounting and statistics data in the web user interface” on page 487

You can use the web user interface to start collecting snapshot accounting and statistics data for your message flows. You can then view the accounting and statistics data in the web user interface.

“Viewing accounting and statistics data in the web user interface” on page 499

You can use the web user interface to view snapshot accounting and statistics data.

Related reference:



mqswebuseradmin command

Use the **mqswebuseradmin** command to administer user accounts for the web user interface.

Configuring the web user interface server

To enable access to broker resources through the web user interface, configure the IBM Integration Bus web user interface server.

Procedure

To configure a web user interface server, complete the following steps:

1. Configure the properties relating to the HTTP or HTTPS port to be used, and enable the web user interface server, by using the IBM Integration Explorer or the **mqsichangeproperties** command.
 - If you are using the IBM Integration Explorer, complete the following steps:

- a. In the MQ Explorer - Navigator view, navigate to **IBM WebSphere MQ > Brokers**, right-click your broker name, and select **Properties**.
 - b. Select the **WebAdmin** tab.
 - c. Enable the web user interface and set the port to use for your chosen protocol.
 If you are using HTTP, set **Enabled** to true, set **Enabled SSL** to false, and enter a value for the **HTTP Port**.
 If you are using HTTPS, set **Enabled** to true, set **Enabled SSL** to true, and enter a value for the **HTTPS Port**.
 To set additional parameters for the HTTPS protocol, you must use the command-line interface. For more information, see Parameter values for the webadmin component.
 - d. To review your changes, click **Apply**; to close the Properties window, click **OK**. You can also confirm the changes by examining recent Change Notification entries in the Administration Log view.
- If you are using the **mqschangeproperties** command, complete the following steps:

- a. Configure the properties relating to the HTTP or HTTPS port to be used.

- If you are using HTTP, run the following command:

```
mqschangeproperties brokerName
-b webadmin -o HTTPConnector -n port -v portValue
- brokerName is the name of your broker
- portValue is the HTTP port value that you want to use for the web
  user interface
```

- If you are using HTTPS, run the following command:

```
mqschangeproperties brokerName
-b webadmin -o HTTPSConnector -n port,keystoreFile,keystorePass
-v portValue,fileName,password
- brokerName is the name of your broker.
- portValue is the HTTPS port value that you want to use for the web
  user interface.
- fileName is your keystore file.
- password is the password for the keystore file.
```

For more information about the properties that can be set for the HTTPConnector and HTTPSConnector objects, see Parameter values for the webadmin component.

- b. Confirm that the properties are set correctly.
 - If you are using HTTP, run the following command:

```
mqsireportproperties brokerName
-b webadmin -o HTTPConnector -a
```

- If you are using HTTPS, run the following command:

```
mqsireportproperties brokerName
-b webadmin -o HTTPSConnector -a
```

This command produces a response similar to this example (for the HTTPConnector):

```
HTTPConnector
  uuid='HTTPConnector'
  address=''
  port='4144'
  maxPostSize=''
```

```

acceptCount=''
compressableMimeTypes=''
compression=''
connectionLinger=''
connectionTimeout=''
maxHTTPHeaderSize=''
maxKeepAliveRequests=''
maxThreads=''
minSpareThreads=''
noCompressionUserAgents='
restrictedUserAgents=''
socketBuffer=''
tcpNoDelay=''
enableLookups='false'

```

c. Enable the web user interface server for the broker:

- To enable the web user interface, and to use HTTP as the communication protocol between the broker and the web user interface server, run the following command on an IBM Integration Bus command line, where *brokerName* is the name of your broker:

```

mqsichangeproperties brokerName
-b webadmin -o server -n enabled,enableSSL -v true,false

```

- To enable the web user interface, and to use HTTPS as the protocol for communication between the broker and the web user interface server, run the following command:

```

mqsichangeproperties brokerName
-b webadmin -o server -n enabled,enableSSL -v true,true

```

For more information about the properties of the web user interface server, see Parameter values for the webadmin component.

d. Confirm that the web user interface component is enabled by running the **mqsireportproperties** command, as shown in the following example.

```

mqsireportproperties brokerName -b webadmin -o server -a

```

This command produces a response similar to this example:

```

server=''
  uuid='server'
  enabled='true'
  enableSSL='true'

```

2. Optional: The time that is taken to load data in the web user interface can be reduced by compressing the data. Compression is turned on by default for new brokers. For migrated brokers, compression is turned off by default.

You can enable and disable compression by using the **mqsichangeproperties** command. When you enable compression, you must use the **mqsichangeproperties** command to turn on compression and set the file types to be compressed.

a. Turn on compression, as shown in this example:

```

mqsichangeproperties brokerName -b webadmin -o HTTPConnector -n compression -v on

```

b. Set the file types that are to be compressed, as shown in this example:

```

mqsichangeproperties brokerName -b webadmin -o HTTPConnector -n compressableMimeTypes
-v \"text/html,text/css,application/javascript,image/gif,image/png,application/json\"

```

c. Use the **mqsireportproperties** command to check that compression has been set correctly, as shown in this example:

```

mqsireportproperties brokerName -b webadmin -o HTTPConnector -r

```

Check the output from the **mqsireportproperties** command to verify that the settings are correct. The output should be similar to that shown in the following example:

```
...  
compressableMimeTypes='text/html,text/css,application/javascript,image/gif,image/png,application/json'  
compression='on'  
...
```

If you are using SSL, change the properties for HTTPSCConnector rather than HTTPConnector.

3. To ensure that the changes take effect, restart the broker. For more information, see “Starting and stopping a broker” on page 19.
4. Create a web user account by using the **mqswebuseradmin** command. For more information, see “Managing web user accounts” on page 226.

What to do next

When you have configured a web user interface server, you can log on to the web user interface by following the steps described in “Accessing the web user interface” on page 133.

Related concepts:

“Role-based security” on page 220

You can control access to broker resources by associating web users with roles.

Related tasks:

“Viewing recorded data” on page 616

You can view a list of recorded messages, or details of individual messages.

“Accessing the web user interface” on page 133

You can access broker resources by logging on to the web user interface.

“Managing web user accounts” on page 226

You can control a web user's access to broker resources by associating the web user ID with a role, which has security permissions assigned to it.


Related reference:

 **mqsissetdbparms** command

Use the **mqsissetdbparms** command to associate a specific user ID and password (or SSH identity file) with one or more resources that are accessed by the broker.

 **mqsichangeproperties** command

Use the **mqsichangeproperties** command to modify broker properties and properties of broker resources.

 Parameter values for the webadmin component

Select the objects and properties associated with the webadmin component that you want to change.

 **mqswebuseradmin** command

Use the **mqswebuseradmin** command to administer user accounts for the web user interface.

Enabling and disabling the web user interface

You can enable and disable the web user interface by using the **mqsichangeproperties** command.

Before you begin

Before you start:

- Read the following topics:
 - “Administering brokers using the web user interface” on page 127
 - “Managing web user accounts” on page 226
- Ensure that a web user interface server has been configured, as described in “Configuring the web user interface server” on page 128.

About this task

All new brokers have the web user interface assigned to port 4414 when they are created, and the port is enabled by default. As a result, the web user interface is enabled automatically when a new broker is created.

You can disable the web user interface by using the **mqsichangeproperties** command to change the enabled property of the webadmin component to false; by default, it is set to true. For example:

```
mqsichangeproperties IB9NODE -b webadmin -o server -n enabled -v false
```

You can change this property either when the broker is stopped or while it is running. If you make changes while the broker is running, they take effect when the broker is restarted.

You can also change the port number for the web user interface by using the **mqsichangeproperties** command; for example:

```
mqsichangeproperties IB9NODE -b webadmin -o HTTP[S]Connector -n port -v new port
```

You can change this property only when the broker is running, and the change takes effect when the broker is restarted. Alternatively, you can use the IBM Integration Explorer to enable or disable the web user interface, and to change the port number to which it is assigned.

For brokers that were created in WebSphere Message Broker Version 8.0.0.0, the web user interface is enabled in later versions only if you enabled it in Version 8.0.0.0. If you created brokers in Version 8.0.0.0 and did not enable the web user interface, it remains disabled for those brokers in subsequent versions. The web user interface is not enabled automatically for brokers that are migrated from WebSphere Message Broker Version 6.1 or Version 7 to later versions of the product.

Related concepts:

“Role-based security” on page 220

You can control access to broker resources by associating web users with roles.



The IBM Integration API

The IBM Integration API is a programming interface that your applications can use to control brokers and their resources through a remote interface.

Related tasks:

“Managing web user accounts” on page 226

You can control a web user's access to broker resources by associating the web user ID with a role, which has security permissions assigned to it.

“Recording, viewing, and replaying data” on page 595

Record data that is processed by a message flow, or emitted from WebSphere

Application Server. View and replay the data.

“Enabling security for record and replay” on page 598

You can restrict the users who can view and replay data for a broker by enabling security.

Chapter 1, “Administering brokers and broker resources,” on page 1

Administering brokers and associated integration node resources includes the tasks that you perform frequently to activate and manage those resources. Choose the method you prefer to administer your brokers and associated resources.

“Administering brokers using the web user interface” on page 127

You can use the IBM Integration Bus web user interface to administer broker resources.

“Configuring the web user interface server” on page 128

To enable access to broker resources through the web user interface, configure the IBM Integration Bus web user interface server.

Related reference:



Parameter values for the webadmin component

Select the objects and properties associated with the webadmin component that you want to change.



mqsichangeproperties command

Use the **mqsichangeproperties** command to modify broker properties and properties of broker resources.

Accessing the web user interface

You can access broker resources by logging on to the web user interface.

Before you begin

Before you start:

- Read the following topics:
 - “Administering brokers using the web user interface” on page 127
 - “Role-based security” on page 220
 - “Managing web user accounts” on page 226
- Ensure that a web user interface server has been configured, as described in “Configuring the web user interface server” on page 128.

About this task

All new brokers that are created in WebSphere Message Broker Version 8.0.0.1 or later have the web user interface assigned to port 4414, which is enabled by default. As a result, the web user interface is enabled automatically when a new broker is created.

You can disable and enable the web user interface by using the **mqsichangeproperties** command to change the enabled property of the webadmin component to false or true. For more information, see “Enabling and disabling the web user interface” on page 131.

When you have configured a web user interface server to be used by IBM Integration Bus, open the web user interface in a web browser:

Procedure

1. Use the following URL:

protocol://serverAddress:port/

where:

- *protocol* has the value `http` or `https`, depending on whether you are using an HTTP or an HTTPS connector object
- *serverAddress* identifies the web user interface server address specified for the HTTP or HTTPS connector object; for example `127.0.0.1`
- *port* identifies the port that you specified for the HTTP or HTTPS connector object in the preceding steps (by default the port is 4414)

For example:

`http://127.0.0.1:4415`

2. Enter your user ID and password and then click **Log in** to log on to the system. Users have permissions granted to them according to their role, so administrators and web users can have different access to broker resources based on their role. The web interface is tailored to your role, so you see only the options that are available to you based on the permissions that have been assigned to your role.

For more information about roles, see “Role-based security” on page 220. For information about creating user accounts, see “Managing web user accounts” on page 226.

If administration security is not enabled, you can access the web user interface as the *default* user without logging on, but you still have access to all data and broker resources.

3. A window opens, in which you can view and administer your broker resources. The Navigator view is displayed on the left side of the window (unless it has been hidden), and the content that is displayed on the right side of the window varies according to the resource that has been selected in the Navigator view.

Related concepts:

“Role-based security” on page 220

You can control access to broker resources by associating web users with roles.



The IBM Integration API

The IBM Integration API is a programming interface that your applications can use to control brokers and their resources through a remote interface.

Related tasks:

“Managing web user accounts” on page 226

You can control a web user's access to broker resources by associating the web user ID with a role, which has security permissions assigned to it.

“Recording, viewing, and replaying data” on page 595

Record data that is processed by a message flow, or emitted from WebSphere Application Server. View and replay the data.

“Enabling security for record and replay” on page 598

You can restrict the users who can view and replay data for a broker by enabling security.

Chapter 1, “Administering brokers and broker resources,” on page 1

Administering brokers and associated integration node resources includes the tasks that you perform frequently to activate and manage those resources. Choose the method you prefer to administer your brokers and associated resources.

“Administering brokers using the web user interface” on page 127

You can use the IBM Integration Bus web user interface to administer broker

resources.

“Configuring the web user interface server” on page 128

To enable access to broker resources through the web user interface, configure the IBM Integration Bus web user interface server.

“Enabling and disabling the web user interface” on page 131

You can enable and disable the web user interface by using the **mqsichangeproperties** command.

“Viewing accounting and statistics data in the web user interface” on page 499

You can use the web user interface to view snapshot accounting and statistics data.

Customizing the data viewer

Broker administrators can customize the appearance of the data shown in the data capture store in the web user interface.

Before you begin

Before you start:

- Read the following topics:
 - “Administering brokers using the web user interface” on page 127
 - “Role-based security” on page 220
- Ensure that your web administration server has been configured; for more information, see “Configuring the web user interface server” on page 128
- Sign in to the web administration server, as described in “Accessing the web user interface” on page 133

About this task

As a broker administrator, you can control the way in which the contents of the data capture store are displayed to all web user interface users. You can define the columns that are displayed, and you can customize the width, order, and names of those columns. The changes that you make apply only to the data capture stores that you have customized; other data capture stores retain their default appearance.

Related concepts:

“Role-based security” on page 220

You can control access to broker resources by associating web users with roles.



The IBM Integration API

The IBM Integration API is a programming interface that your applications can use to control brokers and their resources through a remote interface.

Related tasks:

“Administering brokers using the web user interface” on page 127

You can use the IBM Integration Bus web user interface to administer broker resources.

“Managing web user accounts” on page 226

You can control a web user's access to broker resources by associating the web user ID with a role, which has security permissions assigned to it.

“Configuring the web user interface server” on page 128

To enable access to broker resources through the web user interface, configure the IBM Integration Bus web user interface server.

“Accessing the web user interface” on page 133

You can access broker resources by logging on to the web user interface.

“Recording, viewing, and replaying data” on page 595

Record data that is processed by a message flow, or emitted from WebSphere Application Server. View and replay the data.

Administering integration nodes from WebSphere Application Server

You can view IBM Integration resources from the WebSphere Application Server administrative console.

Before you begin

Before you start:

- Read the topic Chapter 1, “Administering brokers and broker resources,” on page 1
- Install the IBM Integration feature into WebSphere Application Server; see Installing IBM Integration Administration for WebSphere Application Server
- Perform the required configuration steps; see Configuring IBM Integration Administration for WebSphere Application Server

About this task

IBM Integration Administration for WebSphere Application Server enables administrators to administer integration node resources from the WebSphere Application Server administrative console.

As a WebSphere Application Server administrator, you can use this feature to:

- Connect to existing integration nodes
- Test connections to integration nodes
- View, start, and stop integration servers, message flows, services, and applications
- View libraries and their properties

You can get help on the IBM Integration Bus collection pages by clicking **Help** on any page.

Complete the following steps to view your IBM Integration resources in the WebSphere Application Server administrative console:

Procedure

1. If your WebSphere Application Server server is not running, start it by running the following command from the *install_root\bin* or *install_root/bin* directory:
 - On Windows:
`startServer server1 -profileName profile`
 - On other operating systems:
`startServer.sh server1 -profileName profileName`
 - *install_root* is the WebSphere Application Server installation root
 - *server1* is the name of your WebSphere Application Server server
 - *profile* is the name of your profile
2. Open the WebSphere Application Server administrative console.

- For example, to open the administrative console on Windows, click **Start > All Programs > IBM WebSphere > IBM WebSphere Application Server - Version > Profiles > profile > Administrative console**
3. To see the collection pages for IBM Integration Bus, click **IBM Integration Bus** in the list of collections.
 4. To see the Integration nodes page, click **IBM Integration Bus > Integration nodes**.

What to do next

You are now ready to create a connection to an existing integration node. Use the instructions on the page and in the related Help topic. To access the Help topic, click **Help** (labeled with a question mark) on the Integration nodes page.

Related tasks:

Chapter 1, “Administering brokers and broker resources,” on page 1
Administering brokers and associated integration node resources includes the tasks that you perform frequently to activate and manage those resources. Choose the method you prefer to administer your brokers and associated resources.

Managing resources used by brokers

Manage the resources used by brokers.

About this task

- “Listing database connections that the broker holds”
- “Quiescing a database” on page 138
- “Using a JDBC connection pool to manage database resources used by an integration server” on page 139
- Modifying a configurable service
- Modifying an IBM defined configurable service
- Configuring properties for TCP/IP

Related tasks:

“Analyzing resource performance” on page 529

You can collect statistics to assess the performance of resources used by integration servers.

Listing database connections that the broker holds

The broker does not provide an interface that you can use to list the connections that it has to a database. You must use the facilities of the database suppliers to list connections.

For further information about how to list database connections, refer to the documentation that is provided by your database vendor.

Related concepts:



Database connections

Databases contain business data that is written and accessed by deployed message flows. You must create connections from the broker to the database by using ODBC or JDBC.

Related tasks:

Working with databases

Create and configure databases to use with your message flows.

Related reference:

Supported databases

You can optionally configure databases to contain data that is accessed by your message flows. Databases from IBM and other suppliers are supported at specific versions on supported operating systems.




“Quiescing a database”

The broker and database exhibit specific behaviors when you quiesce the database.

WebSphere MQ connections

The number of WebSphere MQ connections a broker requires to its queue manager depends on the actions of the message flows that access the WebSphere MQ resource.

Related information:

-  [DB2 V9.1 product documentation \(distributed systems\)](#)
-  [DB2 V9.5 product documentation \(distributed systems\)](#)
-  [DB2 Information Center \(z/OS\)](#)

Quiescing a database

The broker and database exhibit specific behaviors when you quiesce the database.

If you access databases from one or more message flows, your database administrator might occasionally want to issue the quiesce instruction on a database. This action is a function of the database, not of the broker.

The following three assumptions are made for the database that you are quiescing:

- The database can be quiesced (not all databases support this function).
- New connections to the database are blocked by the database when it is quiescing.
- Message flows that access the database eventually become idle.


The following list shows the behavior that is expected while a database is quiescing:

- Run the command that quiesces the database. When this command starts, the connections that are in use remain in use, but no new connections to the database are allowed.
- Messages that are being processed by message flows, which use existing connections to the database, continue to use their connections until the connections become idle. Therefore if messages continue to be received by the message flow, it might be a long time before the quiesce occurs. To ensure that messages are no longer processed, stop the message flow. Stopping the message flow stops messages being processed, and releases the database connections that the flow was using. This action ensures that the database connections that the flow holds become idle.
- Database connections for the message flow become idle. This situation causes the broker to release the connections to the databases that the message flow is

using. When all connections to the database from the broker, and from any other applications that are using the database, are released, the database can complete its quiesce function.

For more information, see Database connections.

Related concepts:

 Database connections

Databases contain business data that is written and accessed by deployed message flows. You must create connections from the broker to the database by using ODBC or JDBC.

Related tasks:

 Enabling ODBC connections to the databases

Set up the resources and environment that the broker requires for Open Database Connectivity (ODBC) connections to databases on distributed systems.

 Enabling JDBC connections to the databases

Configure connections to a database through a JDBCProvider configurable service.

Related reference:

 Supported databases

You can optionally configure databases to contain data that is accessed by your message flows. Databases from IBM and other suppliers are supported at specific versions on supported operating systems.

“Listing database connections that the broker holds” on page 137

The broker does not provide an interface that you can use to list the connections that it has to a database. You must use the facilities of the database suppliers to list connections.

 WebSphere MQ connections

The number of WebSphere MQ connections a broker requires to its queue manager depends on the actions of the message flows that access the WebSphere MQ resource.

Using a JDBC connection pool to manage database resources used by an integration server

Use broker JDBC provider resources to configure the use of thread pools independently of message flow and input node thread pools.

About this task

IBM Integration Bus manages JDBC connections in the following ways:

- Non-pooled connections:
 - IBM Integration Bus creates a JDBC connection on demand for each message flow instance that requires one.
 - Each JDBC connection is associated with the message flow instance for which it was created. This association is maintained until the connection is closed.
 - Each JDBC connection that is idle for 60 seconds is closed, and is no longer associated with a message flow instance.
 - After a JDBC connection that was associated with a message flow instance is closed, if the same message flow instance requires a JDBC connection, IBM Integration Bus creates a new JDBC connection on demand.

- Pooled connections:
 - When a message flow instance requires a JDBC connection, IBM Integration Bus assigns an unused connection from the pool.
 - If all pooled JDBC connections are being used, and the maximum pool size has not been reached, IBM Integration Bus creates a new pooled JDBC connection. The maximum pool size is specified in the `maxConnectionPoolSize` property of the `JDBCProviders` configurable service.
 - Each pooled JDBC connection remains associated with a message flow instance only for the processing of one input message.
 - When a message flow instance completes the processing of an input message, the association with a JDBC connection is removed, and the JDBC connection is returned to the pool.
 - Each pooled JDBC connection that is idle for 15 minutes is closed, and is removed from the pool.
 - Pooled JDBC connections are not applicable to the `DatabaseRetrieve` and `DatabaseRoute` nodes.

Using a JDBC connection pool enable you to scale database access independently of the number of message flow threads.

You can create a JDBC connection pool by setting the `maxConnectionPoolSize` property of the `JDBCProviders` configurable service to a non-zero integer value. The `maxConnectionPoolSize` property acts at the integration server level to specify the maximum number of JDBC connection threads that can be used. A value of zero defaults to the standard WebSphere Message Broker Version 8.0 behavior, where one JDBC connection is created for each message flow thread.

All message flows within an integration server that use the same `JDBCProviders` configurable service also share a connection pool. You can monitor the behavior of a JDBC connection pool by using broker resource statistics

The `maxConnectionPoolSize` property is applicable to JDBC connections obtained using the `getJDBCType4Connection()` API of the `JavaCompute` node, and to database operations in graphical data maps that are called by the `Mapping` node.

Note: The `maxConnectionPoolSize` property does not apply to the JDBC connections used by the `DatabaseRetrieve` or `DatabaseRoute` nodes.

Related tasks:

“Viewing resource statistics data in the IBM Integration Explorer” on page 537
Use the IBM Integration Explorer to view resource statistics data for your integration servers in the `Broker Resources` and `Broker Resources Graph` views.

“Viewing the status of resource statistics collection in the IBM Integration Explorer” on page 541

Use the IBM Integration Explorer to view the status of resource statistics collection in the `Broker Resources` view.

“Managing resources used by brokers” on page 137
Manage the resources used by brokers.

Related reference:



Resource statistics data

Learn about the measurements for which data is returned when you activate resource statistics collection.



Configurable services properties

The supplied configurable services, and the configurable services that you create, are defined by their names and properties. You can use the supplied services.



JDBCProviders configurable service

Select the objects and properties that you want to change for the JDBCProviders configurable service.

Managing data caching

Store data that you want to reuse by using the embedded global cache or an external WebSphere eXtreme Scale grid.

About this task

The topics in this section describe the data caching capability that is provided by WebSphere eXtreme Scale. For information about other caching solutions, see the following topics:

- Environment tree structure
- Long-lived variables

WebSphere eXtreme Scale provides IBM Integration Bus with data caching capability.

Find out more about the embedded global cache and external WebSphere eXtreme Scale grids in the following topics:

- “Data caching overview” on page 142
- “Embedded global cache” on page 150
- “WebSphere eXtreme Scale grids” on page 153
- “Global cache scenario: Storing state for integrations” on page 144
- “Global cache scenario: Caching static data” on page 147
- “Data caching terminology” on page 155
- “Configuring the embedded global cache” on page 162
- “Configuring the embedded global cache by using commands” on page 164
- “Configuring the embedded global cache by using IBM Integration Explorer” on page 167
- “Configuring the global cache for multiple brokers” on page 170
- “Configuring the global cache for multi-instance brokers” on page 173
- “Connecting to a WebSphere eXtreme Scale grid” on page 179
- “Enabling SSL for external WebSphere eXtreme Scale grids” on page 181
- “Monitoring the global cache” on page 183

Related tasks:



Accessing the global cache with a JavaCompute node

You can use a JavaCompute node to interact with a map in a global cache or an external WebSphere eXtreme Scale grid.

Related reference:



Parameter values for the cachemanager component

Select the objects and properties that are associated with the global cache that you want to change.



JavaCompute node

Use the JavaCompute node to work with messages by using the Java language.

[WebSphere eXtreme Scale product documentation](#)

[WebSphere eXtreme Scale product web page](#)

Data caching overview

WebSphere eXtreme Scale provides IBM Integration Bus with data caching capability.

The topics in this section describe the data caching capability that is provided by WebSphere eXtreme Scale. For information about other caching solutions, see the following topics:

- Environment tree structure
- Long-lived variables

A global cache is a repository for data that you want to reuse. For example, you can use a global cache in WebSphere MQ message flows to store correlation information for use beyond a specific node, instance of a message flow, integration server, or broker. The cache facilitates sharing of data across processes (both in the same broker, and across brokers) and eliminates the need for an alternative solution, such as a database. You can use one node to store data in the global cache, then a second node (in the same message flow or a separate flow), can retrieve that data from the global cache.

You can use a message flow node to interact with the global cache. Interactions with the cache happen outside the message flow transaction, and are committed immediately. If an exception is thrown downstream of the node that interacts with the cache, the cache interactions are not rolled back.

For illustrations of how you can use a global cache, see the following scenarios:

- “Global cache scenario: Storing state for integrations” on page 144
- “Global cache scenario: Caching static data” on page 147

For an example of how to use the global cache, see the Coordinated Request Reply sample. You can view information about samples only when you use the information center that is integrated with the IBM Integration Toolkit or the online information center. You can run samples only when you use the information center that is integrated with the IBM Integration Toolkit.

The following demonstrations of the global cache are also available:

- IBM Education Assistant module: Global cache - single broker
- IBM Education Assistant module: Global cache - multiple brokers

You can use the global cache that is supplied with IBM Integration Bus, and you can configure IBM Integration Bus to connect to an external WebSphere eXtreme Scale grid. You can work with multiple remote grids, and the embedded grid, at the same time.

For a detailed description of the elements involved in a global cache, see the following information:

- “Embedded global cache” on page 150
- “WebSphere eXtreme Scale grids” on page 153

Interacting with the global cache or external grid

You can interact with the global cache or external grid by using a JavaCompute node. The node can put data into a map, retrieve data, and create a map if one does not exist. For more information, see [Accessing the global cache with a JavaCompute node](#).

Monitoring the global cache or external grid

You can monitor the global cache or external grid by using the following tools:

- The activity log
- Resource statistics

You can administer the embedded cache by using the `mqsicacheadmin` command.

For more information, see [“Monitoring the global cache”](#) on page 183.

Related concepts:

[“Data caching terminology”](#) on page 155

The global cache is embedded in the broker. You can also connect to an external WebSphere eXtreme Scale grid.

[“Global cache scenario: Storing state for integrations”](#) on page 144

You can use a global cache to store state for integrations. With a global cache, each broker can handle replies, even when the request was processed by another broker.

[“Global cache scenario: Caching static data”](#) on page 147

You can use a global cache to store static data. The use of a global cache facilitates horizontal scaling in situations where a cache is used to minimize network interactions to a back end system. With a global cache, you can increase the number of clients while maintaining a predictable response time for each client.

Related tasks:

[“Configuring the embedded global cache”](#) on page 162

Configure properties of the embedded global cache by using commands, IBM Integration Explorer, an XML policy file, or the IBM Integration API.

[“Managing data caching”](#) on page 141

Store data that you want to reuse by using the embedded global cache or an external WebSphere eXtreme Scale grid.



[Accessing the global cache with a JavaCompute node](#)

You can use a JavaCompute node to interact with a map in a global cache or an external WebSphere eXtreme Scale grid.

Related reference:



[Parameter values for the cachemanager component](#)

Select the objects and properties that are associated with the global cache that you want to change.



[JavaCompute node](#)

Use the JavaCompute node to work with messages by using the Java language.

[WebSphere eXtreme Scale product documentation](#)

[WebSphere eXtreme Scale product web page](#)

Global cache scenario: Storing state for integrations:

You can use a global cache to store state for integrations. With a global cache, each broker can handle replies, even when the request was processed by another broker.

When IBM Integration Bus is used to integrate asynchronous systems, the broker records information about the requester to correlate the replies correctly, as shown in the following diagram.

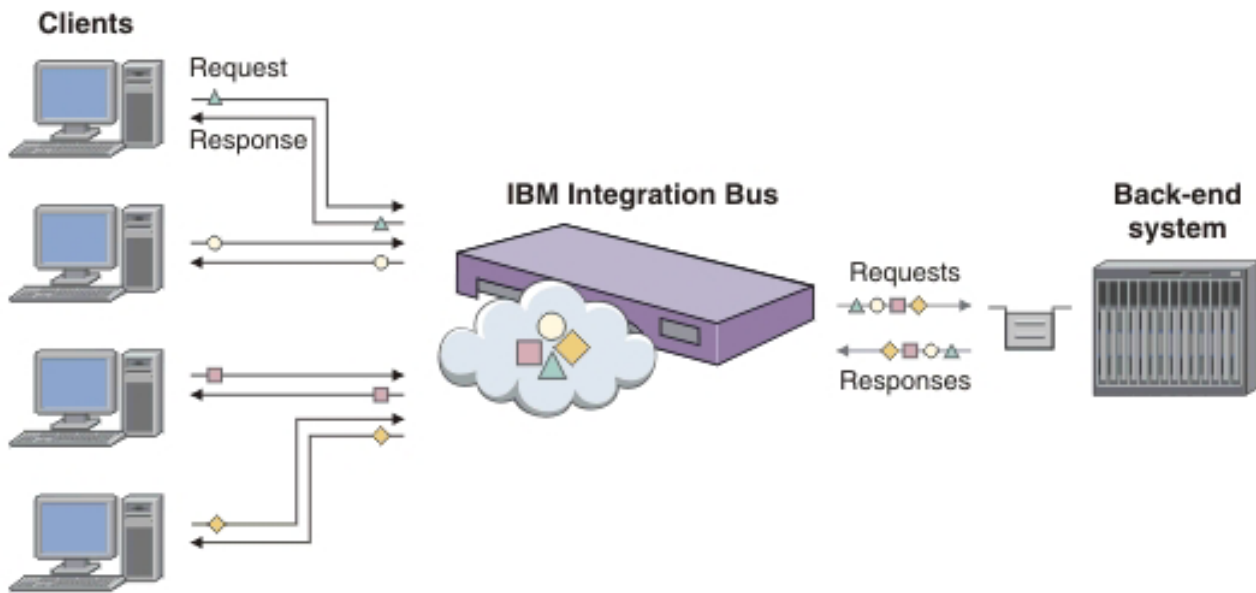


Figure 4. Graphic showing the placement of the global cache in a multi-broker environment.

When this integration is deployed to multiple brokers for workload balancing, the reply will not necessarily return through the same broker as the original request.

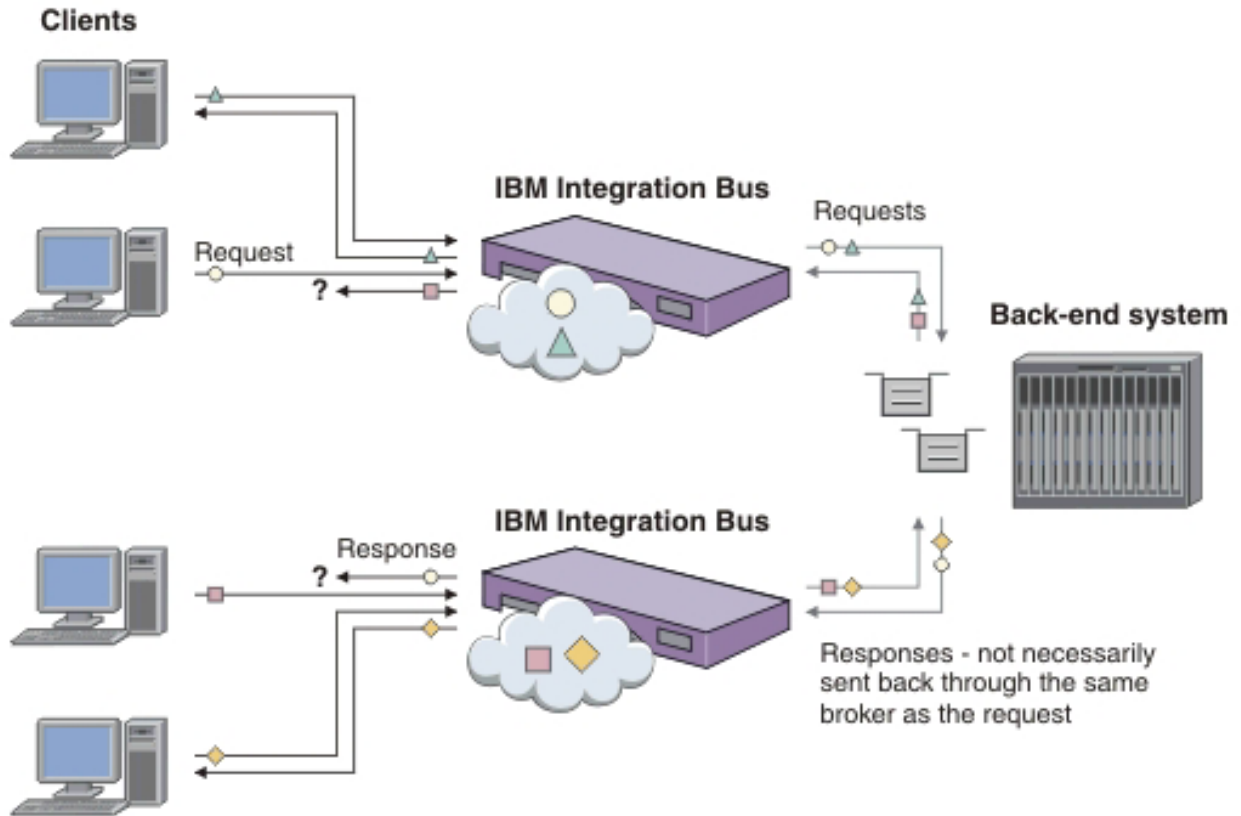


Figure 5. Graphic showing the placement of the global cache in a multi-broker environment.

With a global cache, each broker can handle replies, even when the request was processed by another broker.

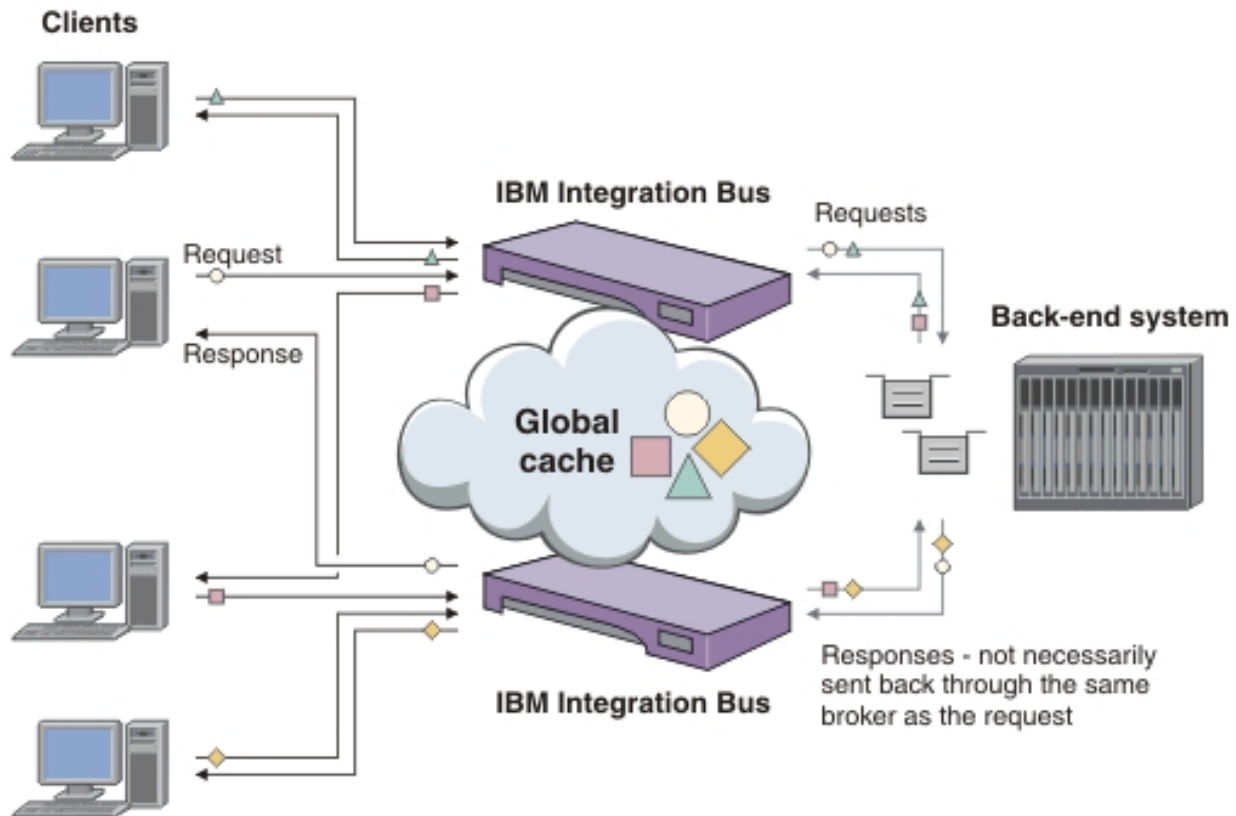


Figure 6. Graphic showing the placement of the global cache in a multi-broker environment.

Related concepts:

“Data caching overview” on page 142

WebSphere eXtreme Scale provides IBM Integration Bus with data caching capability.

“Data caching terminology” on page 155

The global cache is embedded in the broker. You can also connect to an external WebSphere eXtreme Scale grid.

“Global cache scenario: Caching static data” on page 147

You can use a global cache to store static data. The use of a global cache facilitates horizontal scaling in situations where a cache is used to minimize network interactions to a back end system. With a global cache, you can increase the number of clients while maintaining a predictable response time for each client.

Related tasks:

“Configuring the embedded global cache” on page 162

Configure properties of the embedded global cache by using commands, IBM Integration Explorer, an XML policy file, or the IBM Integration API.

“Managing data caching” on page 141


Store data that you want to reuse by using the embedded global cache or an external WebSphere eXtreme Scale grid.




Accessing the global cache with a JavaCompute node

You can use a JavaCompute node to interact with a map in a global cache or an external WebSphere eXtreme Scale grid.

Related reference:

 Parameter values for the cachemanager component
Select the objects and properties that are associated with the global cache that you want to change.

 JavaCompute node
Use the JavaCompute node to work with messages by using the Java language.

[WebSphere eXtreme Scale product documentation](#)

[WebSphere eXtreme Scale product web page](#)

Global cache scenario: Caching static data:

You can use a global cache to store static data. The use of a global cache facilitates horizontal scaling in situations where a cache is used to minimize network interactions to a back end system. With a global cache, you can increase the number of clients while maintaining a predictable response time for each client.

When IBM Integration Bus acts as a façade to a back-end database, it must provide short response times to the client, even though the back-end database has high latency.

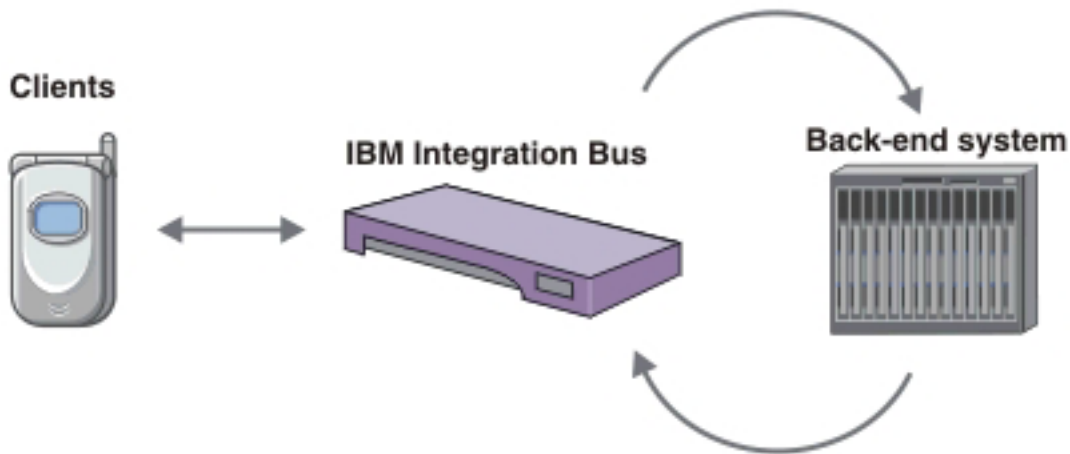


Figure 7. Graphic showing the placement of the global cache in a multiple client scenario.

IBM Integration Bus can provide short response times by caching results in memory (for example, ESQL shared variables).

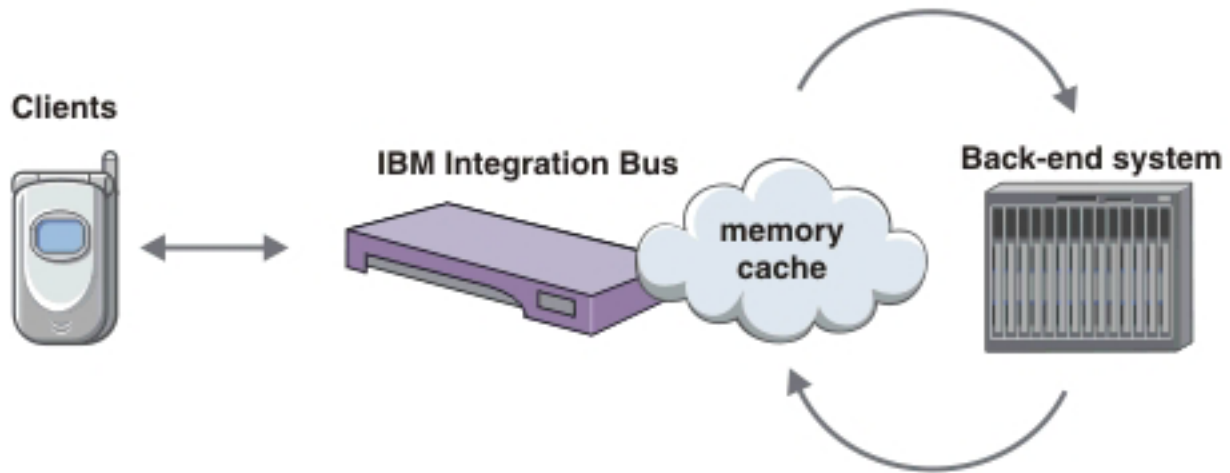


Figure 8. Graphic showing the placement of the global cache in a multiple client scenario.

However, this configuration does not scale well horizontally. When the number of clients increases, the number of brokers and integration servers can be increased to accommodate the clients, but each broker has to maintain a separate in-memory cache.

Number of clients might increase rapidly.

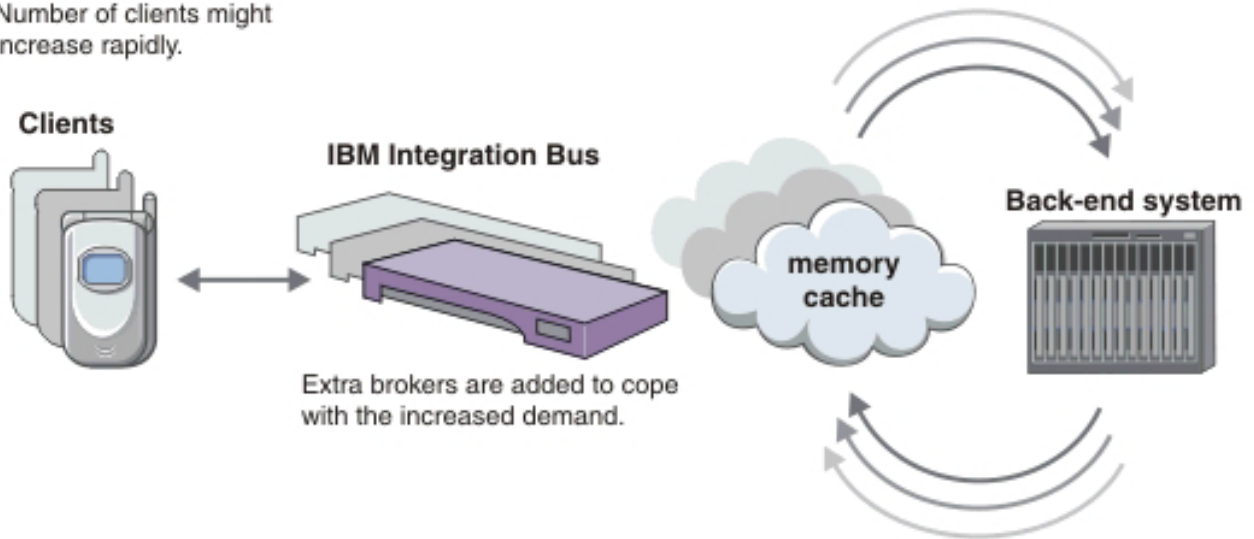


Figure 9. Graphic showing the placement of the global cache in a multiple client scenario.

With a global cache, the number of clients can increase while a predictable response time is maintained for each client.

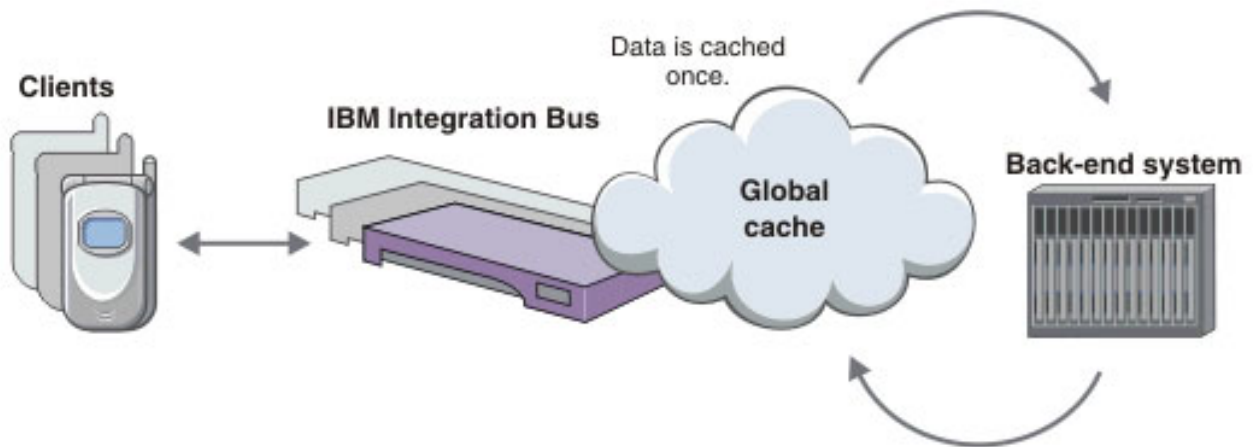


Figure 10. Graphic showing the placement of the global cache in a multiple client scenario.

Related concepts:

“Data caching overview” on page 142

WebSphere eXtreme Scale provides IBM Integration Bus with data caching capability.

“Data caching terminology” on page 155

The global cache is embedded in the broker. You can also connect to an external WebSphere eXtreme Scale grid.

“Global cache scenario: Storing state for integrations” on page 144

You can use a global cache to store state for integrations. With a global cache, each broker can handle replies, even when the request was processed by another broker.

Related tasks:

“Configuring the embedded global cache” on page 162

Configure properties of the embedded global cache by using commands, IBM Integration Explorer, an XML policy file, or the IBM Integration API.

“Managing data caching” on page 141

Store data that you want to reuse by using the embedded global cache or an external WebSphere eXtreme Scale grid.



Accessing the global cache with a JavaCompute node

You can use a JavaCompute node to interact with a map in a global cache or an external WebSphere eXtreme Scale grid.

Related reference:



Parameter values for the cachemanager component

Select the objects and properties that are associated with the global cache that you want to change.



JavaCompute node

Use the JavaCompute node to work with messages by using the Java language.

WebSphere eXtreme Scale product documentation

WebSphere eXtreme Scale product web page

Embedded global cache:

Use the global cache that is supplied with IBM Integration Bus to store data that you want to reuse.

The global cache is embedded in the broker. By default, the cache is disabled; to use the cache, select an appropriate cache policy. The default cache policy creates a default topology of cache components in a single broker. The alternatives to the default topology are to have no policy, so that you can control your own topology by setting cache properties on the integration servers, or to use an XML policy file to enable the cache across multiple brokers.

The default topology

By default, one integration server in the broker hosts a catalog server. The catalog server controls placement of data and monitors the health of container servers. Up to three other integration servers in that broker host container servers. A container server is a component that is embedded in the integration server that holds a subset of the cache data. The catalog server and container servers are placed in integration servers dynamically when the broker starts. All integration servers can communicate with the global cache, regardless of whether they are hosting catalog servers, container servers, or neither. Each integration server contains a cache manager, which manages the cache components that are embedded in that integration server. For a description of the cache components, see “Data caching terminology” on page 155.

Placement of the catalog server is reset when the broker is restarted. By default, the broker sets a range of ports to use, but you can specify a particular range of ports. If you stop the integration server that contains the catalog server, the global cache becomes unavailable. If you stop and restart integration servers that host container servers, the container servers are reassigned dynamically. Existing cache data is preserved and rebalanced automatically, unless you stop all container servers. Integration servers are configured in the following order:

- The first integration server to start is a catalog server and a container server. This integration server uses the first four ports from the range.
- The second integration server to start is a container server only; it uses the three ports in the range (ports 5 - 7).
- The third and fourth integration servers to start are container servers only, and use ports 8 - 10 and 11 - 13 from the range.
- If further integration server are started, they do not host global cache components. However, all integration servers can communicate with the global cache as clients.
- If you stop the integration server that contains the catalog server, the global cache becomes unavailable.
- If you stop the second, third, or fourth integration server, the container server of the stopped integration server is assigned to the next integration server that starts.

If you restart the integration server that hosts the catalog server, it can no longer communicate with the container servers in other integration servers. Although these container servers are still running, they are no longer part of the cache, and your data is lost. Therefore, you must also restart the integration servers that host the container servers. Alternatively, restart the broker to reset all cache components.

When you use the default topology, the cache properties for individual integration servers are read only; an error is issued if you try to change them. When you use the default topology, you can specify the range of ports that the cache manager uses, and the listener host that is used by the cache components. If your computer has more than one host name, setting the listener host ensures that the cache components use the correct host name. To set specific properties for the integration server, you must first change the cache policy property to none. The integration server properties that were set most recently by the broker-level policy are retained as a starting point for customization.

Multi-broker cache topologies

To share data across brokers, or enhance the availability of the cache, you must create a policy file. The policy file is an XML file that the cache manager uses to connect the caches of multiple brokers. Set the cache policy to the fully qualified name of the policy file. Use the policy file to specify the names of the brokers that will share the cache data, the number of catalog servers in each broker, and the port range and listener host for each broker to use. You can also use a policy file to configure a multi-instance broker to host container servers. Some sample policy files are provided in the installation directory. You can copy and customize these policy files for your own system. The sample XML files describe the following configurations:

- A single broker that hosts two catalog servers; if one catalog server fails, the global cache switches to the other one.
- Two brokers that share a catalog server that is hosted by the first broker.
- Two brokers that each host a catalog server; if one catalog server fails, the global cache switches to the catalog server in the other broker.
- Three brokers in a high availability scenario. Two brokers each host a catalog server, and a multi-instance broker hosts two container servers. If the active instance of the multi-instance broker fails, the global cache switches to the container servers in the standby instance.

For more information, see “Configuring the global cache for multiple brokers” on page 170 and “Configuring the global cache for multi-instance brokers” on page 173.

Customizing the embedded topology

You can set properties explicitly for each integration server. For example, you might want to specify particular integration servers to host the catalog and container servers so that you can tune broker performance. When the cache policy is set to none, the cache manager in each integration server uses the values that you set. The integration server properties that were set most recently by the broker-level policy are retained as a starting point for customization.

If you stop the integration server that contains the catalog server, the cache becomes unavailable and the data in the cache is lost. Therefore, if you switch off the default topology, ensure that you place the catalog server appropriately. If you restart the integration server that hosts the catalog server, it can no longer communicate with the container servers in other integration servers. Although these container servers are still running, they are no longer part of the cache, and your data is lost. Therefore, you must also restart the integration servers that host the container servers. Alternatively, restart the broker to reset all cache components.

When you use multiple catalog servers, you can improve performance by taking the following steps:

- Provide other integration servers that host container servers only, rather than having only integration servers that host both catalog and container servers.
- Start and stop integration servers in sequence, rather than using the **mqsistart** or **mqsistop** commands to start or stop all integration servers at once. For example, start the integration servers that host catalog servers before you start the integration servers that host only container servers.

When you use a global cache that spans multiple brokers, ensure that all WebSphere eXtreme Scale servers that are clustered in one embedded grid use the same domain name. Only servers with the same domain name can participate in the same grid. WebSphere eXtreme Scale clients use the domain name to identify and distinguish between embedded grids. If you do not specify a domain name in the integration server or broker-level policy file, the broker creates a name that is based on the server names of the catalog servers.

By default, each server starts with a domain name that is derived by the broker. In previous versions of IBM Integration Bus, the domain name for all servers in all embedded caches was an empty string. Servers in different domains cannot collaborate in the same grid. Therefore, for a cache that spans multiple brokers, migrate these brokers at the same time.

You can configure the global cache by using IBM Integration Bus commands, IBM Integration Explorer, or the IBM Integration API. For more information, see “Configuring the embedded global cache” on page 162.

You can disable all cache components in the broker by setting the cache policy property to disabled. The cache policy is set to disabled by default.

Interaction with the global cache

You can use JavaCompute nodes to store and retrieve data in a map in the global cache. When you get a global map from an external grid, the `getGlobalMap` method makes a connection to the grid if one does not exist. You can use your own Java classes with the global cache. To use your own Java classes, put the JAR files that contain the Java classes in the shared classes directory at system level, broker level, or integration server level. You must make the Java classes available to all brokers and integration servers that participate in the global cache. For more information, see [Accessing the global cache with a JavaCompute node](#).

When you get an `MbGlobalMap` object, you can also specify how long the data remains in the global cache before it is removed automatically. This time is known as the *time to live* and is counted from when that map entry is last updated. The value applies to all cache entries that are created by using that `MbGlobalMap` object in that instance of the message flow. Data that is already in the map that you specify, or that is created by another `MbGlobalMap` object, is not affected by the time to live value. You can create multiple `MbGlobalMap` objects in different flows, integration servers, or brokers, all resolving to the same map in the global cache, but with different time to live values.

By default, the time to live is set to zero so that the data is never removed. To set a specific time to live, create a session policy, which you can reference from the `MbGlobalMap` object. For detailed instructions, see [Removing data from the global cache](#).

Related concepts:

“Data caching terminology” on page 155

The global cache is embedded in the broker. You can also connect to an external WebSphere eXtreme Scale grid.

“WebSphere eXtreme Scale grids”

Use one or more external WebSphere eXtreme Scale grids to store data that you want to reuse.

“Global cache scenario: Storing state for integrations” on page 144

You can use a global cache to store state for integrations. With a global cache, each broker can handle replies, even when the request was processed by another broker.

“Global cache scenario: Caching static data” on page 147

You can use a global cache to store static data. The use of a global cache facilitates horizontal scaling in situations where a cache is used to minimize network interactions to a back end system. With a global cache, you can increase the number of clients while maintaining a predictable response time for each client.

Related tasks:

“Configuring the embedded global cache” on page 162

Configure properties of the embedded global cache by using commands, IBM Integration Explorer, an XML policy file, or the IBM Integration API.

“Managing data caching” on page 141

Store data that you want to reuse by using the embedded global cache or an external WebSphere eXtreme Scale grid.

“Connecting to a WebSphere eXtreme Scale grid” on page 179

You can store data in a WebSphere eXtreme Scale grid for use by other message flows.



Accessing the global cache with a JavaCompute node

You can use a JavaCompute node to interact with a map in a global cache or an external WebSphere eXtreme Scale grid.

Related reference:

Parameter values for the cachemanager component

Select the objects and properties that are associated with the global cache that you want to change.



JavaCompute node

Use the JavaCompute node to work with messages by using the Java language.

[WebSphere eXtreme Scale product documentation](#)

[WebSphere eXtreme Scale product web page](#)

WebSphere eXtreme Scale grids:

Use one or more external WebSphere eXtreme Scale grids to store data that you want to reuse.

WebSphere eXtreme Scale provides a scalable, in-memory data grid. The data grid dynamically caches, partitions, replicates, and manages data across multiple servers. The catalog servers and container servers for the IBM Integration Bus global cache collaborate to act as a WebSphere eXtreme Scale grid. For more information about the grid that is embedded in the broker, see “Embedded global cache” on page 150.

In addition to the grid that is available (as the embedded global cache) within IBM Integration Bus, you can integrate with WebSphere eXtreme Scale grids that are running elsewhere. You can work with multiple external grids, and the embedded grid, at the same time. For a diagram that shows how IBM Integration Bus message flows interact with an external WebSphere eXtreme Scale grid, and for definitions of the terms that are associated with a grid, see “Data caching terminology” on page 155. For more information about grids, see WebSphere eXtreme Scale product documentation.

You might have an existing WebSphere eXtreme Scale environment that uses a topology such as one of the following examples:

- A DataPower® XC10 device
- Embedded WebSphere eXtreme Scale servers in WebSphere Application Server network deployment installations
- A stand-alone WebSphere eXtreme Scale software grid

When you are connecting to an external grid, IBM Integration Bus hosts the WebSphere eXtreme Scale client, but the cache is hosted on an external grid. Therefore, a separate WebSphere eXtreme Scale installation or appliance is required, such as a DataPower XC10 device. An installation or appliance can host more than one grid.

To connect to an external grid, you need the following information:

- The name of the grid
- The host name and port of each catalog server for the grid
- Optional: the object grid file that can be used to override WebSphere eXtreme Scale properties

By using a configurable service to specify the parameters, you can connect to an external WebSphere eXtreme Scale grid. IBM Integration Bus message flows can then access and modify data that is stored in the external grid. If you are connecting to a secure grid, you can create a security identity by using the **mqsisetdbparms** command. To connect to an external grid, follow the instructions in “Connecting to a WebSphere eXtreme Scale grid” on page 179.

You can enable SSL for client connections to external WebSphere eXtreme Scale grids by setting up a public key infrastructure, then enabling SSL for an integration server. For more details, see “Enabling SSL for external WebSphere eXtreme Scale grids” on page 181.

You can use JavaCompute nodes to store and retrieve data in a map in the external grid. When you get a global map from an external grid, the `getGlobalMap` method makes a connection to the grid if one does not exist. For instructions, see Accessing the global cache with a JavaCompute node.

You can monitor the external grid by viewing the activity log and resource statistics. For more information, see “Monitoring the global cache” on page 183.

To enable new function that is available in fix packs, you use the **-f** parameter on the **mqsichangebroker** command. Running this command also enables the specification of domain names. In previous versions of IBM Integration Bus, the domain name for all WebSphere eXtreme Scale servers in all embedded caches was an empty string. Servers in different domains cannot collaborate in the same grid. Therefore, for a cache that spans more than one broker, you must enable the new capability for these brokers at the same time.

Related concepts:

“Data caching terminology”

The global cache is embedded in the broker. You can also connect to an external WebSphere eXtreme Scale grid.

“Global cache scenario: Storing state for integrations” on page 144

You can use a global cache to store state for integrations. With a global cache, each broker can handle replies, even when the request was processed by another broker.

“Global cache scenario: Caching static data” on page 147

You can use a global cache to store static data. The use of a global cache facilitates horizontal scaling in situations where a cache is used to minimize network interactions to a back end system. With a global cache, you can increase the number of clients while maintaining a predictable response time for each client.

Related tasks:

“Configuring the embedded global cache” on page 162

Configure properties of the embedded global cache by using commands, IBM Integration Explorer, an XML policy file, or the IBM Integration API.

“Connecting to a WebSphere eXtreme Scale grid” on page 179

You can store data in a WebSphere eXtreme Scale grid for use by other message flows.

“Managing data caching” on page 141

Store data that you want to reuse by using the embedded global cache or an external WebSphere eXtreme Scale grid.



Accessing the global cache with a JavaCompute node

You can use a JavaCompute node to interact with a map in a global cache or an external WebSphere eXtreme Scale grid.

Related reference:



Parameter values for the cachemanager component

Select the objects and properties that are associated with the global cache that you want to change.



JavaCompute node

Use the JavaCompute node to work with messages by using the Java language.



mqsi changebroker command

Use the **mqsi changebroker** command to change one or more of the configuration parameters of the broker.

WebSphere eXtreme Scale product documentation

WebSphere eXtreme Scale product web page

Data caching terminology:

The global cache is embedded in the broker. You can also connect to an external WebSphere eXtreme Scale grid.

The embedded cache has a default single-broker topology and can be used by message flows running in any integration server on the broker, without any configuration. However, you can switch off the default topology by selecting a broker policy of none, and set properties explicitly for each integration server.

The following diagram shows the embedded global cache in a broker that contains six integration servers. Four integration servers host components for the global

cache, but message flows in all six integration servers can use the cache.

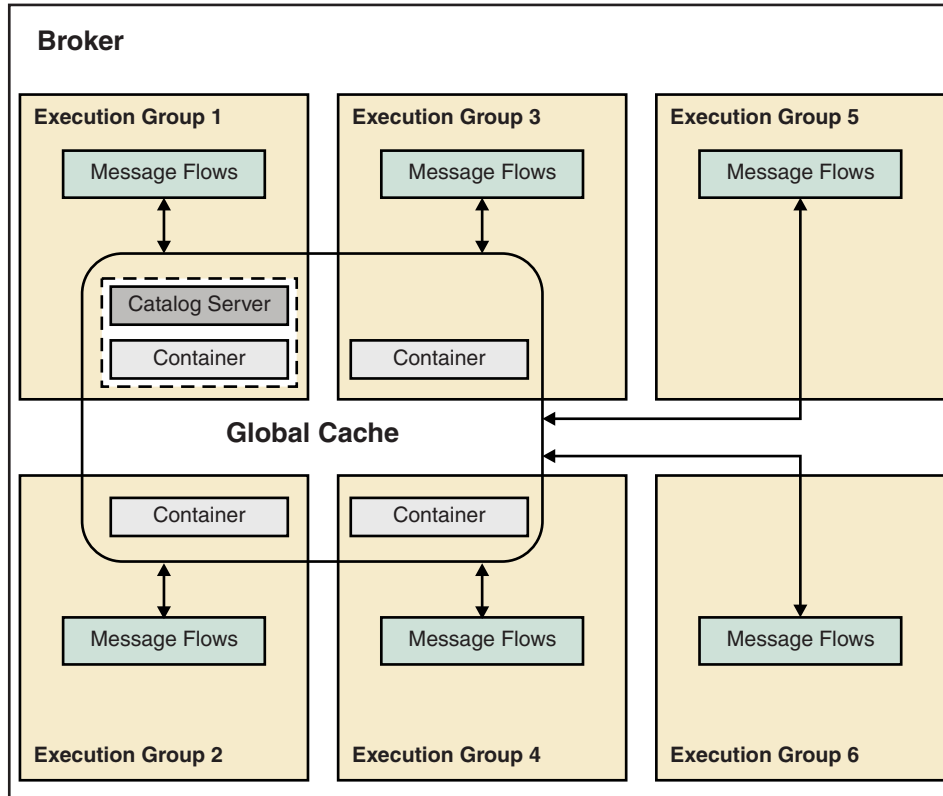


Figure 11. Diagram showing how integration server 1 hosts a catalog server and a container server, and integration servers 2, 3, and 4 host container servers only. Integration servers 5 and 6 do not host any cache components, but their message flows can still communicate with the cache.

The following diagram shows a multi-broker embedded cache, which is configured by the broker policy file `policy_two_brokers_ha.xml`.

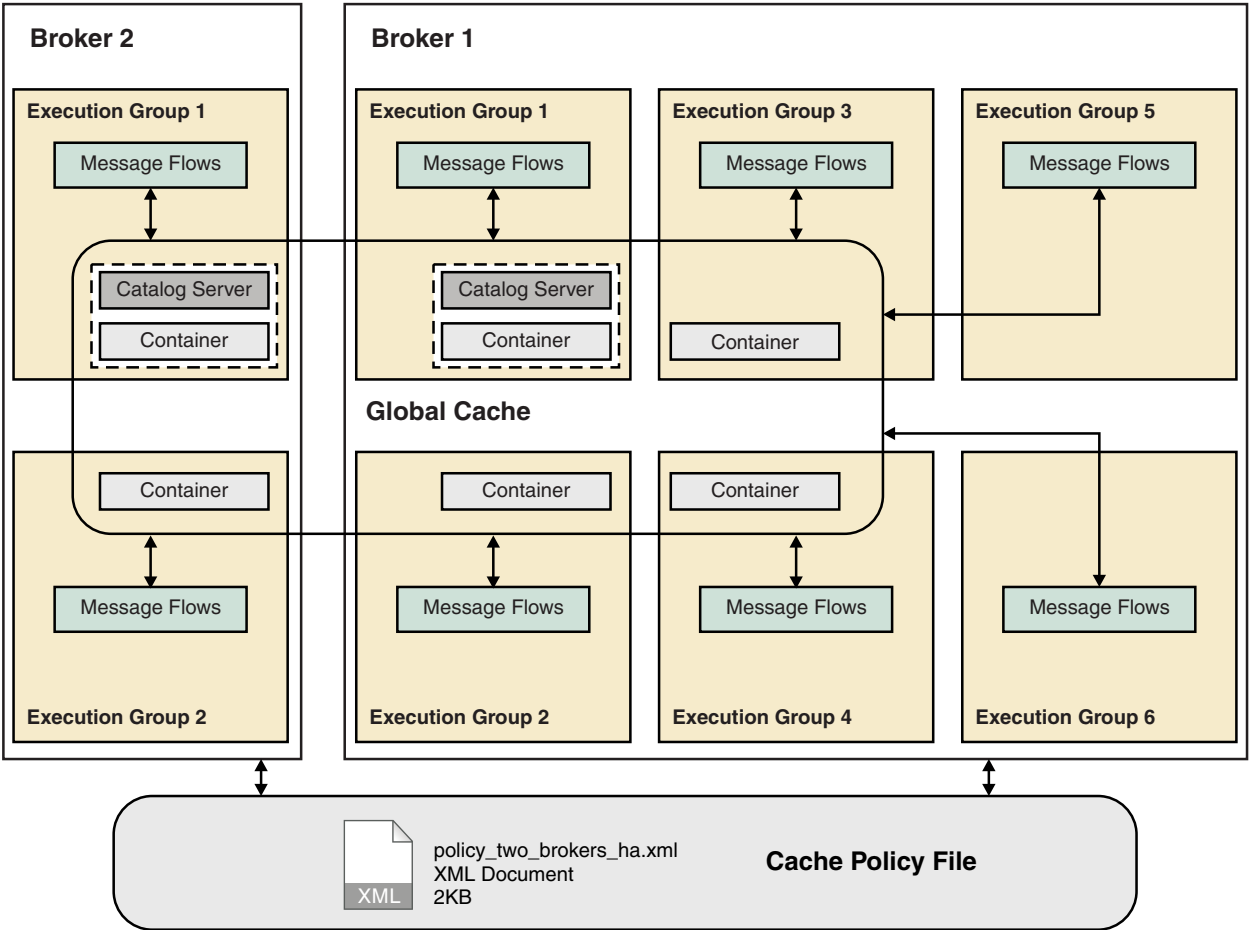


Figure 12. Diagram showing two brokers that are participating in an embedded cache. Integration server 1 of broker 1 contains a catalog server and a container server. Integration server 1 of broker 2 also contains a catalog server and container server. Integration servers 2, 3, and 4 of broker 1 contain container server. Integration server 2 of broker 2 also contains a container server.

The following diagram shows how IBM Integration Bus can connect to both an embedded cache and an external WebSphere eXtreme Scale grid. A configurable service is used to connect to the external grid.

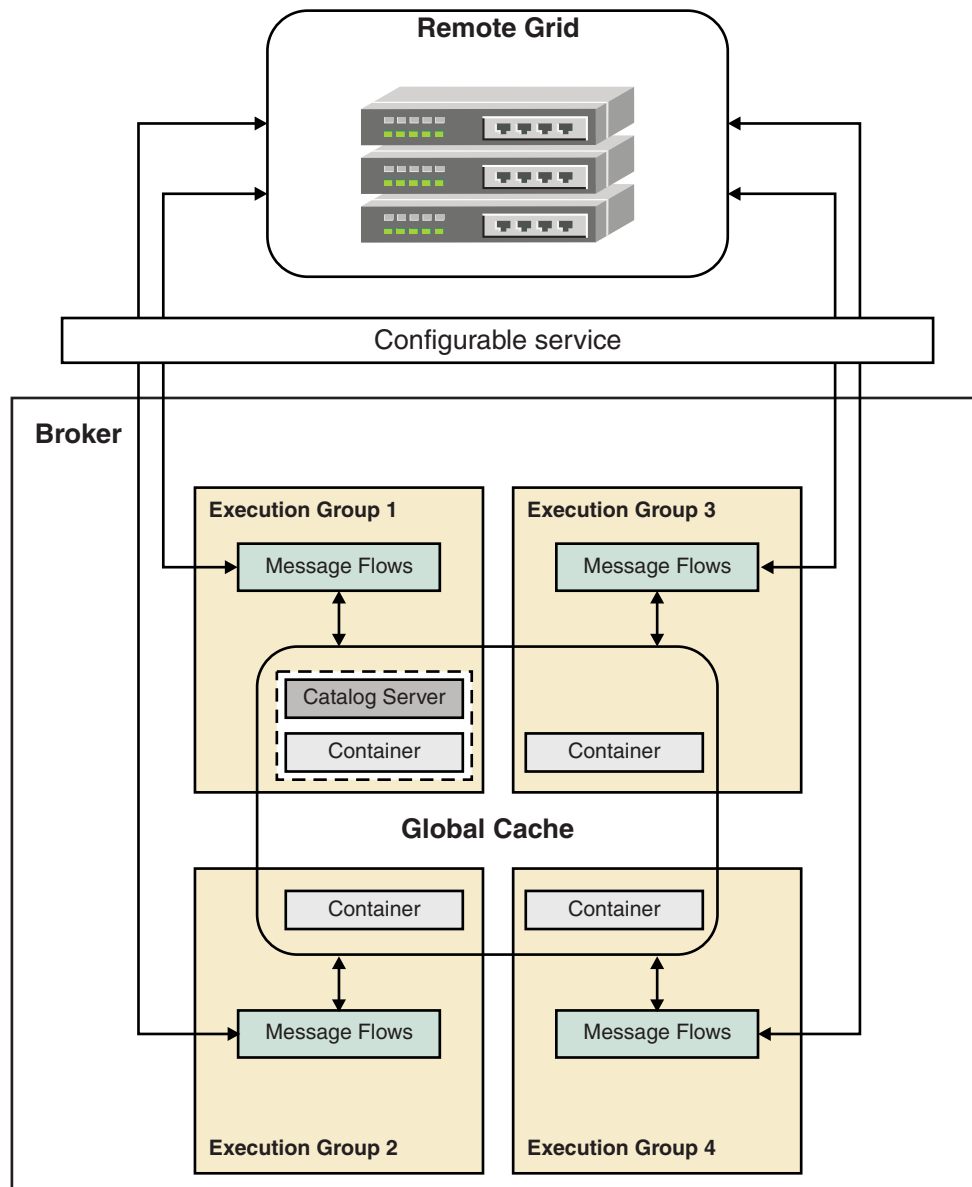


Figure 13. Diagram showing how IBM Integration Bus can connect to an embedded cache and a WebSphere eXtreme Scale grid at the same time. Integration server 1 in the broker contains a catalog server and a container server. Integration servers 2, 3, and 4 each host a container server. Double-ended arrows link the message flows in each integration server to the embedded cache and to a remote WebSphere eXtreme Scale grid. Between the message flows and remote grid is a box that represents the configurable service that is used to connect to the external grid.

The following components are involved in the global cache.

Broker-level properties

By default, the global cache is turned off, and the cache policy is set to disabled. To use the global cache, select a broker-level cache policy by using the **cachePolicy** parameter.

IBM Integration Bus has a default cache policy that creates a default topology of cache components in a single broker. The default topology puts catalog servers and container servers in integration servers dynamically so that the cache is available for use by all integration servers in the broker. Broker-level properties are available to specify a range of ports, and a

listener host for the default topology. The broker sets a range of ports to use, but you can specify a particular range of ports by using the **cachePortRange** parameter. You can use the **listenerHost** parameter to specify the listener host that is used by the cache components. If your computer has more than one host name, setting the listener host ensures that the cache components use the correct host name.

If you set the cache policy to none, you must set the integration server properties explicitly. The properties that were set most recently by the broker-level policy are used as a starting point. Therefore, if you set the cache policy to default first, then switch to none, the default topology properties are retained.

You can configure the global cache to span multiple brokers by setting the cache policy to the fully qualified name of an XML policy file. This policy file lists the brokers that share the cache, and for each broker specifies the listener host, port range, and the number of catalog servers hosted. You can use the policy file to set up a single broker that hosts two catalog servers. If one catalog server is stopped, the broker switches to the other catalog server, ensuring that no cache data is lost. You can also use the policy file to configure a multi-instance broker to host more than one container server. If the active instance of the multi-instance broker fails, the global cache switches to the container server in the standby instance.

If you set the cache policy to disabled, all cache components in the broker are disabled. The disabled policy is the default setting.

For more information, see “Configuring the embedded global cache” on page 162 and Parameter values for the cachemanager component.

Cache manager

The cache manager is the integration server resource that manages the cache components that are embedded in that integration server.

In the default topology, one integration server in the broker hosts a catalog server, and up to three other integration servers in that broker host container servers. All integration servers can communicate with the global cache, regardless of whether they are hosting catalog servers, container servers, or neither. Each integration server contains a cache manager, which manages the cache components that are embedded in that integration server. When you turn off the default topology, configure the integration servers by setting the parameter values for the cachemanager component.

For more information, see “Configuring the embedded global cache” on page 162 and Parameter values for the cachemanager component.

Container servers

A container server is a component that is embedded in the integration server that holds a subset of the cache data. Between them, all container servers in the global cache host all of the cache data at least once. If more than one container exists, the default cache policy ensures that all data is replicated at least once. In this way, the global cache can cope with the loss of container servers without losing data.

You can host more than one container server in a multi-instance broker. If the active instance of the multi-instance broker fails, the global cache switches to the container server in the standby instance.

Catalog servers

The catalog server controls placement of data and monitors the health of containers. You must have at least one catalog server in your global cache.

To avoid losing cache data when a catalog server is lost, use a policy file to specify more than one catalog server for a broker. For example, if you specify two catalog servers for a single broker, if one catalog server fails, the broker switches to the other catalog server. If the cache is shared by two brokers, each of which hosts a catalog server, if one catalog server fails, the brokers switch to the remaining catalog server. Having more than one catalog server can affect startup time until the cache is available. If you have more than one catalog server, you must start at least two of them for the cache to be available. When you configure a cache across multiple brokers with multiple catalog servers, if you need to start one broker before the others then you can configure this broker to host two catalog servers. You cannot host catalog servers in a multi-instance broker.

When you are using multiple catalog servers, you can improve performance by taking the following steps:

- Provide other integration servers that host container servers only, rather than having only integration servers that host both catalog and container servers.
- Start and stop integration servers in sequence, rather than using the **mqsistart** or **mqsistop** commands to start or stop all integration servers at once. For example, start the integration servers that host catalog servers before you start the integration servers that host only container servers.

Domain name

When you are using a global cache that spans multiple brokers, ensure that all WebSphere eXtreme Scale servers that are clustered in one embedded grid use the same domain name. Only servers with the same domain name can participate in the same grid. WebSphere eXtreme Scale clients use the domain name to identify and distinguish between embedded grids. If you do not specify a domain name in the integration server or broker-level policy file, the broker creates a name that is based on the server names of the catalog servers.

By default, each server starts with a domain name that is derived by the broker. In previous versions of IBM Integration Bus, the domain name for all WebSphere eXtreme Scale servers in all embedded caches was an empty string. Servers in different domains cannot collaborate in the same grid. Therefore, for a cache that spans more than one broker, migrate those brokers at the same time.

Grids WebSphere eXtreme Scale provides a scalable, in-memory data grid. The data grid dynamically caches, partitions, replicates, and manages data across multiple servers. The catalog servers and container servers for the IBM Integration Bus global cache collaborate to act as a WebSphere eXtreme Scale grid. For more information about grids, see WebSphere eXtreme Scale product documentation.

Maps Data is stored in maps. A map is a data structure that maps keys to values. One map is the default map, but the global cache can have several maps.

The cache uses WebSphere eXtreme Scale dynamic maps. Any map name is allowed, apart from names that begin with **SYSTEM.BROKER**, which is reserved for use by the broker. The default map is named **SYSTEM.BROKER.DEFAULTMAP**; you can use or clear this map.

ObjectGrid file

An ObjectGrid XML file is used to configure the WebSphere eXtreme Scale client. You can use this file to override WebSphere eXtreme Scale

properties. For more information about configuring clients, see WebSphere eXtreme Scale product documentation.

You can configure WebSphere eXtreme Scale options by using the following tools:

- IBM Integration Bus commands (see “Configuring the embedded global cache by using commands” on page 164)
- IBM Integration Explorer (see “Configuring the embedded global cache by using IBM Integration Explorer” on page 167)
- An XML policy file (see “Configuring the global cache for multiple brokers” on page 170 and “Configuring the global cache for multi-instance brokers” on page 173)
- IBM Integration API (see “Configuring the embedded global cache” on page 162)
- Integration server properties (see Parameter values for the cachemanager component)
- Configurable services (see “Connecting to a WebSphere eXtreme Scale grid” on page 179)

You can use resource statistics and activity trace to monitor the status of the global cache and external grid, and to diagnose problems. You can also administer the embedded global cache by using the `mqsicacheadmin` command.

Related concepts:

“Data caching overview” on page 142

WebSphere eXtreme Scale provides IBM Integration Bus with data caching capability.

“Global cache scenario: Storing state for integrations” on page 144

You can use a global cache to store state for integrations. With a global cache, each broker can handle replies, even when the request was processed by another broker.

“Global cache scenario: Caching static data” on page 147

You can use a global cache to store static data. The use of a global cache facilitates horizontal scaling in situations where a cache is used to minimize network interactions to a back end system. With a global cache, you can increase the number of clients while maintaining a predictable response time for each client.

Related tasks:

“Configuring the embedded global cache” on page 162

Configure properties of the embedded global cache by using commands, IBM Integration Explorer, an XML policy file, or the IBM Integration API.

“Connecting to a WebSphere eXtreme Scale grid” on page 179

You can store data in a WebSphere eXtreme Scale grid for use by other message flows.

“Managing data caching” on page 141

Store data that you want to reuse by using the embedded global cache or an external WebSphere eXtreme Scale grid.



Accessing the global cache with a JavaCompute node

You can use a JavaCompute node to interact with a map in a global cache or an external WebSphere eXtreme Scale grid.

Related reference:



Parameter values for the cachemanager component

Select the objects and properties that are associated with the global cache that you want to change.



JavaCompute node

Use the JavaCompute node to work with messages by using the Java language.



`mqsicacheadmin` command

Use the `mqsicacheadmin` command to provide information about the global cache and complete basic administration.



Global cache Activity log

The global cache activity log provides a high-level overview of how IBM Integration Bus interacts with the global cache or external grid so that you can understand what your message flows are doing.



Resource statistics data: global cache

Learn about the data that is returned for the GlobalCache resource type when you activate resource statistics collection.

[WebSphere eXtreme Scale product documentation](#)

[WebSphere eXtreme Scale product web page](#)

Configuring the embedded global cache

Configure properties of the embedded global cache by using commands, IBM Integration Explorer, an XML policy file, or the IBM Integration API.

Before you begin

Before you start:

- For concept information, see “Data caching overview” on page 142.
- When you are using the global cache, increase the JVM heap size for integration servers that are hosting cache components. For more information, see the *Planning environment capacity* section of the WebSphere eXtreme Scale product documentation.

About this task

By default, the global cache is turned off, and the cache policy is set to disabled. To use the global cache, select a broker-level cache policy by using the `cachePolicy` parameter. The global cache has a default single-broker topology that can be used immediately without any configuration. To use the default topology, change the cache policy property to default. The broker sets a range of ports to use, but you can specify a particular range of ports. You can also specify the listener host that is used by the broker cache components. If your computer has more than one host name, setting the listener host ensures that the cache components use the correct host name.

When you use the default topology, integration server properties are read only; an error is issued if you try to change them. You can switch off the default topology by selecting a broker cache policy of none, and set properties explicitly for each integration server. For example, you might want to specify particular integration servers to host the catalog and container servers so that you can tune broker performance. The integration server properties that were set most recently by the broker-level policy are retained as a starting point for customization.

You can configure the global cache to span multiple brokers by providing an XML policy file. This policy file lists the brokers that share the cache, and for each broker specifies the listener host, port range, and the number of catalog servers hosted. You can use the policy file to set up a single broker that hosts two catalog

servers. If one catalog server is lost, the broker switches to the other catalog server, ensuring that no cache data is lost. You can also use the policy file to configure a multi-instance broker to host more than one container server. If the active instance of the multi-instance broker fails, the global cache switches to the container server in the standby instance.

To disable all cache components in the broker, set the cache policy property to disabled. Integration server properties are removed and cannot be changed. When the cache is enabled, the memory usage of integration servers that host cache components is larger. If this memory usage is an issue, and you do not need to use the global cache, set the cache policy property to disabled.

If you stop the integration server that contains the catalog server, the cache becomes unavailable. Therefore, if you switch off the default topology, ensure that you place the catalog server appropriately. If you restart the integration server that hosts the catalog server, it can no longer communicate with the container servers in other integration servers. Although these container servers are still running, they are no longer part of the cache, and your data is lost. Therefore, you must also restart the integration servers that host the container servers. Alternatively, restart the broker to reset all cache components.

You can configure the properties of the global cache in the following ways:

- By using IBM Integration Bus commands; see “Configuring the embedded global cache by using commands” on page 164.
- By using the IBM Integration Explorer; see “Configuring the embedded global cache by using IBM Integration Explorer” on page 167.
- By using a policy file; see “Configuring the global cache for multiple brokers” on page 170 and “Configuring the global cache for multi-instance brokers” on page 173.
- By using the IBM Integration API.

The cache manager properties are documented in the Javadoc for the IBM Integration API (see IBM Integration API.) You can also access the Javadoc for the IBM Integration API through the Start menu at **Start > All Programs > IBM Integration Bus 9.0.0.0 > IBM Integration Java APIs > IBM Integration API Documentation**.

Related concepts:

“Data caching overview” on page 142

WebSphere eXtreme Scale provides IBM Integration Bus with data caching capability.

“Data caching terminology” on page 155

The global cache is embedded in the broker. You can also connect to an external WebSphere eXtreme Scale grid.

“Global cache scenario: Storing state for integrations” on page 144

You can use a global cache to store state for integrations. With a global cache, each broker can handle replies, even when the request was processed by another broker.

“Global cache scenario: Caching static data” on page 147

You can use a global cache to store static data. The use of a global cache facilitates horizontal scaling in situations where a cache is used to minimize network interactions to a back end system. With a global cache, you can increase the number of clients while maintaining a predictable response time for each client.

Related tasks:

“Managing data caching” on page 141

Store data that you want to reuse by using the embedded global cache or an external WebSphere eXtreme Scale grid.

“Setting the JVM heap size” on page 544

When you start an integration server, it creates a Java virtual machine (JVM) for executing a Java user-defined node.



Accessing the global cache with a JavaCompute node

You can use a JavaCompute node to interact with a map in a global cache or an external WebSphere eXtreme Scale grid.

“Monitoring the global cache” on page 183

You can use the **mqsicacheadmin** command, resource statistics, and the activity log to monitor the global cache. You can use resource statistics and the activity log to monitor external grids.

Related reference:



Parameter values for the cachemanager component

Select the objects and properties that are associated with the global cache that you want to change.



JavaCompute node

Use the JavaCompute node to work with messages by using the Java language.

[WebSphere eXtreme Scale product documentation](#)

[WebSphere eXtreme Scale product web page](#)

Configuring the embedded global cache by using commands:

You can customize the default global cache topology to use a specific port range and listener host, or you can turn off the default topology and specify your own integration server properties. Multiple brokers can share a cache by using a policy file.

Before you begin

Before you start:

For more information about the default global cache topology, see “Data caching overview” on page 142.

About this task

If you are using the default global cache topology, you can use IBM Integration Bus commands to set a port range and listener host for the cache manager to use, and to report the properties for the cache manager. If you want multiple brokers to share data in the cache, or to configure a cache with enhanced availability, you can use commands to specify a policy file. For more information, see “Broker properties” on page 165.

Use IBM Integration Bus commands to turn off the default topology and set integration server properties explicitly (see “Integration server properties” on page 165).

You can also use the **mqsicacheadmin** command to clear data from a map. For more information, see **mqsicacheadmin** command.

Broker properties:

Procedure

- To specify a policy for the cache manager to use, set the **-b (cachePolicy)** parameter on the **mqsicreatebroker** or **mqsichangebroker** command.
The **cachePolicy** parameter specifies the policy to use for the cache manager. You can set this parameter to default, disabled, none, or the fully qualified name of an XML policy file.
 - If you set this property to default, the default global cache topology is used. The following example shows how to enable the default cache topology:
mqsichangebroker brokerName -b default
 - If you set this property to disabled, the global cache components in the broker are disabled. The cache is disabled by default.
 - If you set this parameter to none, you must set the integration server properties explicitly.
 - If you specify the fully qualified name of a policy file, the brokers listed in the policy file are configured to share the data in the global cache. The path name must be absolute, not relative. The following example shows how to use the **mqsichangebroker** command to set the name of a policy file:
mqsichangebroker brokerName -b c:\filepath\policy.xml

For more information, see “Configuring the global cache for multiple brokers” on page 170.

You must stop the broker before you run the **mqsichangebroker** command. Changes take effect when you restart the broker.

- To specify a port range for the cache manager to use, set the **-r (cachePortRange)** parameter on the **mqsicreatebroker** or **mqsichangebroker** command.
The **cachePortRange** parameter specifies a range of ports that the cache manager can use. Set this parameter to generate or to a specific range of ports.
 - If you specify a range of ports, the value of this parameter must be in the format xxxx-yyyy, and the range must contain at least 20 ports.
 - If you specify generate, the broker generates a range of ports that are not being used by another broker on that computer. The broker chooses a range that starts from 2800. If, for example, another broker is using ports 2800 to 2819, the broker generates a range from 2820 to 2839.

You must stop the broker before you run the **mqsichangebroker** command. Changes take effect when you restart the broker.

Use of these parameters is shown in the following example.

```
mqsichangebroker broker_name -b default -r 2809-2825
```

- You can also use the **mqsichangeproperties** command to change broker properties while the broker is running, although you do have to restart the broker for the changes to take effect. To set the policy, port range, and listener host for the broker, set the **-b (componentName)** parameter to cachemanager, and use the **-n** and **-v** parameters to set the relevant properties, as shown in the following example:

```
mqsichangeproperties broker_name -b cachemanager -o CacheManager -n policy,portRange,listenerHost -v default,generate,host_name
```

- To report properties for the cache manager, set the **-b** parameter on the **mqsireportproperties** command to cachemanager, as shown in the following example.

```
mqsireportproperties broker_name -b cachemanager -o CacheManager -r
```

Integration server properties:

Before you begin

Before you start:

Before you can set properties explicitly for an integration server, you must set the cache policy property to none for the broker. You can set the policy by using the **mqscreatebroker**, **mqschangebroker**, or **mqschangeproperties** command, as described in “Broker properties” on page 165. The integration server properties that were set most recently by the broker-level policy are retained as a starting point for customization.

About this task

If you set the cache policy property to none, you must set properties on the integration servers. If you stop the integration server that contains the only catalog server, the cache becomes unavailable. Therefore, if you switch off the default topology, ensure that you place the catalog server appropriately. If you restart the integration server that hosts the catalog server, it can no longer communicate with the container servers in other integration servers. Although these container servers are still running, they are no longer part of the cache, and your data is lost. Therefore, you must also restart the integration servers that host the container servers. Alternatively, restart the broker to reset all cache components.

To use the new properties that you set, restart the integration server.

Procedure

- To set cache manager properties for an integration server, use the **mqschangeproperties** command and specify the object name `ComIbmCacheManager`. For example, to start the catalog service in integration server 1, set the `enableCatalogService` property to true, as shown in the following example.

```
mqschangeproperties broker_name -e execution_group_1 -o ComIbmCacheManager -n enableCatalogService -v true
```

- To provide a specific domain name for the WebSphere eXtreme Scale servers in an integration server, set the `domainName` property, as shown in the following example. Only servers with the same domain name can participate in the same embedded grid. Therefore, ensure that all servers that participate in the same embedded grid use the same domain name. If you do not set a specific domain name, the broker creates a domain name that is based on the server names of the catalog servers.

```
mqschangeproperties broker_name -e execution_group_1 -o ComIbmCacheManager -n domainName -v myDomain
```

- To start a container server in integration server 2, set the `enableContainerService` property to true, as shown in the following example.

```
mqschangeproperties broker_name -e execution_group_2 -o ComIbmCacheManager -n enableContainerService -v true
```

For a list of cache manager properties that you can set for an integration server, see Parameter values for the cachemanager component.

- To specify a list of listener hosts when using multi-instance brokers, enclose the comma-separated list with a pair of backslash characters and quotation marks (`\`):

```
mqschangeproperties broker_name -e execution_group_2 -o ComIbmCacheManager -n listenerHost -v \"host1,host2\"
```

For more information about configuring the global cache for multi-instance brokers, see “Configuring the global cache for multi-instance brokers” on page 173.

- To report cache manager properties for the integration server, set the **-o** parameter on the **mqsireportproperties** command to `ComIbmCacheManager`, and specify the integration server name, as shown in the following example.

```
mqsireportproperties broker_name -e execution_group_name -o ComIbmCacheManager -r
```

Related concepts:

“Data caching overview” on page 142

WebSphere eXtreme Scale provides IBM Integration Bus with data caching capability.

“Data caching terminology” on page 155

The global cache is embedded in the broker. You can also connect to an external WebSphere eXtreme Scale grid.

“Global cache scenario: Storing state for integrations” on page 144

You can use a global cache to store state for integrations. With a global cache, each broker can handle replies, even when the request was processed by another broker.

“Global cache scenario: Caching static data” on page 147

You can use a global cache to store static data. The use of a global cache facilitates horizontal scaling in situations where a cache is used to minimize network interactions to a back end system. With a global cache, you can increase the number of clients while maintaining a predictable response time for each client.

Related tasks:

“Configuring the embedded global cache” on page 162

Configure properties of the embedded global cache by using commands, IBM Integration Explorer, an XML policy file, or the IBM Integration API.

“Managing data caching” on page 141

Store data that you want to reuse by using the embedded global cache or an external WebSphere eXtreme Scale grid.



Accessing the global cache with a JavaCompute node

You can use a JavaCompute node to interact with a map in a global cache or an external WebSphere eXtreme Scale grid.

Related reference:



Parameter values for the cachemanager component

Select the objects and properties that are associated with the global cache that you want to change.



JavaCompute node

Use the JavaCompute node to work with messages by using the Java language.

WebSphere eXtreme Scale product documentation

WebSphere eXtreme Scale product web page

Configuring the embedded global cache by using IBM Integration Explorer:

You can set cache manager properties at integration node (broker) level and integration server level by using IBM Integration Explorer.

Before you begin

Before you start:

For more information about the default global cache topology, see “Data caching overview” on page 142.

About this task

Quick views are provided in IBM Integration Explorer to view the properties for an integration node or an integration server. For example, you can see if an integration server is hosting a catalog server or container server. In the Navigator view, click an integration node or integration server, and see the quick views in the Content view.

If you are using the default global cache topology, you can use IBM Integration Explorer to set a port range for the cache manager to use, and to specify a listener host for the cache components to use (see “Integration node properties”).

You can also use IBM Integration Explorer to turn off the default topology and set integration server properties explicitly (see “Integration server properties”), or disable cache components. If you want to share the data in the global cache between multiple integration nodes, or configure the cache for enhanced availability, you can use IBM Integration Explorer to specify an XML policy file.

Integration node properties:

About this task

To set the cache policy and port range by using IBM Integration Explorer, complete the following steps.

Procedure

1. Open IBM Integration Explorer.
2. Right-click the integration node for which you want to set properties, then click **Properties**. The integration node properties are displayed.
3. Click **Global Cache**. The global cache properties are displayed.
4. Set the cache policy to none, default, disabled, or the fully qualified name of an XML policy file.
 - If you set this property to default, the default global cache topology is used.
 - If you set this parameter to none, you must set the integration server properties explicitly.
 - If you set this parameter to disabled, all cache components in the integration node are disabled. The cache is disabled by default.
 - If you specify the fully qualified name of a policy file, the integration nodes listed in the policy file are configured to share the data in the global cache. The path must be absolute, not relative. For more information, see “Configuring the global cache for multiple brokers” on page 170.
5. Set the port range to generate or to a specific range of ports.
 - If you specify a range of ports, the value of this parameter must be in the format xxx-yyy, and the range must contain at least 20 ports.
 - If you specify generate, the integration nodes generates a range of ports that are not being used by another integration nodes on that computer. The integration node chooses a range that starts from 2800. If, for example, another integration node is using ports 2800 to 2819, the integration node generates a range from 2820 to 2839.
6. Set the host name or IP address that the cache components use to listen.
7. To save your settings, click **OK**.
8. To use the new settings, restart the integration node.

Integration server properties:

Before you begin

Before you start:

1. Before you can set properties explicitly for an integration server, you must set the cache policy property to none for the integration node, as described in “Integration node properties” on page 168. The integration server properties that were set most recently by the integration node-level policy are retained as a starting point for customization.
2. For the change in policy to take effect, you must restart the integration node.

About this task

One integration server in the integration node hosts a catalog server, and up to three other integration servers in that integration node host container servers. If you stop the integration server that contains the only catalog server, the cache becomes unavailable. Therefore, if you switch off the default topology, ensure that you place the catalog server appropriately. If you restart the integration server that hosts the catalog server, it can no longer communicate with the container servers in other integration servers. Although these container servers are still running, they are no longer part of the cache, and your data is lost. Therefore, you must also restart the integration servers that host the container servers. Alternatively, restart the integration node to reset all cache components.

To set cache manager properties for an integration server by using IBM Integration Explorer, complete the following steps.

Procedure

1. Open IBM Integration Explorer.
2. In the Navigator view, expand the Integration Nodes folder, then expand the integration node that contains the integration server for which you want to set properties.
3. Right-click the integration server, then click **Properties**. The integration node properties are displayed.
4. Click **Global Cache**. The global cache properties are displayed.
5. Set the properties as appropriate. For a description of the properties that you can set, see Parameter values for the cachemanager component.
6. To save your settings, click **OK**.
7. To use the new settings, restart the integration server. (Remember that if you stop and restart the integration server that contains the catalog server, the cache becomes unavailable.)

Related concepts:

“Data caching overview” on page 142

WebSphere eXtreme Scale provides IBM Integration Bus with data caching capability.

“Data caching terminology” on page 155

The global cache is embedded in the broker. You can also connect to an external WebSphere eXtreme Scale grid.

“Global cache scenario: Storing state for integrations” on page 144

You can use a global cache to store state for integrations. With a global cache, each broker can handle replies, even when the request was processed by another broker.

“Global cache scenario: Caching static data” on page 147

You can use a global cache to store static data. The use of a global cache facilitates horizontal scaling in situations where a cache is used to minimize network interactions to a back end system. With a global cache, you can increase the number of clients while maintaining a predictable response time for each client.

Related tasks:

“Configuring the embedded global cache” on page 162

Configure properties of the embedded global cache by using commands, IBM Integration Explorer, an XML policy file, or the IBM Integration API.

“Managing data caching” on page 141

Store data that you want to reuse by using the embedded global cache or an external WebSphere eXtreme Scale grid.



Accessing the global cache with a JavaCompute node

You can use a JavaCompute node to interact with a map in a global cache or an external WebSphere eXtreme Scale grid.

Related reference:



Parameter values for the cachemanager component

Select the objects and properties that are associated with the global cache that you want to change.



JavaCompute node

Use the JavaCompute node to work with messages by using the Java language.

WebSphere eXtreme Scale product documentation

WebSphere eXtreme Scale product web page

Configuring the global cache for multiple brokers:

Configure one or more brokers to share data in the global cache by using an XML policy file. Use a policy file to enhance the availability of the cache by configuring a broker to host two catalog servers. Also use a policy file to host container servers in a multi-instance broker for high availability.

Before you begin

Before you start:

For more information about the default global cache topology, see “Data caching overview” on page 142.

About this task

Specify a policy file for your brokers to use by setting the cache policy to the name and file path of the policy file. Sample policy files are provided in the installation directory *install_dir/sample/globalcache*, covering the following configurations:

- A single broker that hosts two catalog servers; if one catalog server fails, the global cache switches to the other one.
- Two brokers that share a catalog server that is hosted by the first broker.
- Two brokers that each host a catalog server; if one catalog server fails, the global cache switches to the catalog server in the other broker.
- Three brokers in a high availability scenario. Two brokers each host a catalog server, and a multi-instance broker hosts two container servers. If the active instance of the multi-instance broker fails, the global cache switches to the container servers in the standby instance.

For specific instructions to configure the global cache for multi-instance brokers, see “Configuring the global cache for multi-instance brokers” on page 173.

Two of the sample policies configure the cache to contain two catalog servers. This configuration means that if one of the catalog servers is stopped, the other catalog server is used, and no cache data is lost. However, having more than one catalog server can affect startup time after the broker is started, until the cache is available. If you have more than one catalog server, you must start at least two of them for the cache to be available. When you configure a cache across multiple brokers with multiple catalog servers, if you need to start one broker before the others then you can configure this broker to host two catalog servers.

When you set the broker-level property to a policy file, the policy file is validated against an XML schema. A copy of the XML schema file is provided at *install_dir/cachesupport/schema*.

Do not edit the sample policy files in their original location; copy them to your own file system first. The original sample policy files might be replaced when you apply maintenance to IBM Integration Bus.

You cannot use the policy file to fix specific cache roles to specific integration servers. Instead, you must use the none policy; see “Embedded global cache” on page 150.

The following steps describe how to configure the global cache for multiple brokers.

Procedure

1. Copy one of the sample policy files from *install_dir/sample/globalcache* to another location on your file system.
You can place a copy of the same policy file on each computer where a broker is running, or you can provide a single copy of the policy file in a shared file system for all brokers to access.
2. Modify the policy file for your system, specifying the appropriate broker names and listener hosts, the port range that the broker is to use, and how many catalog servers the broker hosts. Optionally, you can also specify a domain name for all catalog servers in the embedded cache. If you do not set a domain name, the broker creates one.

Ensure that the policy meets the following criteria:

- You can define 0, 1, or 2 catalog servers for an individual broker, but at least one catalog server must be defined in the policy.
- If two brokers share a host name, you must set a distinct port range for each broker.
- Ensure that the port range for each broker includes at least 20 ports.
- The broker names and listener hosts specified in the policy must match the values defined for the brokers.
- You can define only one domain name in the policy file.
- If specified, the domain name must precede the broker elements in the policy file.
- The policy file must be encoded in UTF-8.
- The policy file must contain valid XML. The policy file is validated against an XML schema when you set the broker-level property. You can also

validate the policy file against the copy of the schema (`policy.xsd`) that is provided in `install_dir/cachesupport/schema`.

- If you are configuring a multi-instance broker, an optional **listenerHost** element is provided as a child of the **broker** element so that you can specify more than one listener host. In this situation, the `listenerHost` attribute of the **broker** element is also optional. However, you must specify either the **listenerHost** element or the `listenerHost` attribute.
- If you are configuring a multi-instance broker, you cannot specify the **listenerHost** element more than once for a broker that is configured to host one or more catalog servers.

When you use an XML policy file, the broker-level **portRange** property is ignored. The port range specified in the XML file overrides the property specified for the broker.

3. Save the policy file.

4. Set the cache policy to the fully qualified name of the policy file.

The path that you specify must be absolute, not relative. If you use a shared drive on Windows, you must use the `\\hostname\directory` path syntax to the shared drive, instead of a mapped drive letter. The IBM Integration Bus user ID that is used to access the `\\hostname\directory` path must have read access to the file system and must use the same password.

You can set the cache policy by using commands (see “Configuring the embedded global cache by using commands” on page 164) or IBM Integration Explorer (see “Configuring the embedded global cache by using IBM Integration Explorer” on page 167).

5. Restart each broker. If your cache is configured for more than one catalog server, then ensure that at least two catalog servers are started.

Results

When each broker restarts, it uses the values in the policy file to determine its cache properties. Each broker contains up to 4 container servers. To find out where container servers are placed, use the **mqsicacheadmin** command to run the **showPlacement** command, as shown in the following example:

```
mqsicacheadmin brokerName -c showPlacement
```

You can also use the **mqsicacheadmin** command to show cache components in a multi-broker cache. For example, the **listHosts** command shows the host names, number of hosts, and number of catalogs in the cache:

```
mqsicacheadmin brokerName -c listHosts
```

Related concepts:

“Data caching overview” on page 142

WebSphere eXtreme Scale provides IBM Integration Bus with data caching capability.

“Data caching terminology” on page 155

The global cache is embedded in the broker. You can also connect to an external WebSphere eXtreme Scale grid.

“Global cache scenario: Storing state for integrations” on page 144

You can use a global cache to store state for integrations. With a global cache, each broker can handle replies, even when the request was processed by another broker.

“Global cache scenario: Caching static data” on page 147

You can use a global cache to store static data. The use of a global cache facilitates horizontal scaling in situations where a cache is used to minimize network interactions to a back end system. With a global cache, you can increase the number of clients while maintaining a predictable response time for each client.

Related tasks:

“Configuring the global cache for multi-instance brokers”

You can configure the global cache to withstand software or hardware failures so that it is available for as much time as possible. Configure a multi-instance broker to host container servers by using an XML policy file.

“Configuring the embedded global cache” on page 162

Configure properties of the embedded global cache by using commands, IBM Integration Explorer, an XML policy file, or the IBM Integration API.

“Managing data caching” on page 141

Store data that you want to reuse by using the embedded global cache or an external WebSphere eXtreme Scale grid.



Accessing the global cache with a JavaCompute node

You can use a JavaCompute node to interact with a map in a global cache or an external WebSphere eXtreme Scale grid.

“Monitoring the global cache” on page 183

You can use the `mqsicacheadmin` command, resource statistics, and the activity log to monitor the global cache. You can use resource statistics and the activity log to monitor external grids.

Related reference:



Parameter values for the cachemanager component

Select the objects and properties that are associated with the global cache that you want to change.



JavaCompute node

Use the JavaCompute node to work with messages by using the Java language.

[WebSphere eXtreme Scale product documentation](#)

[WebSphere eXtreme Scale product web page](#)

Configuring the global cache for multi-instance brokers:

You can configure the global cache to withstand software or hardware failures so that it is available for as much time as possible. Configure a multi-instance broker to host container servers by using an XML policy file.

Before you begin

Before you start:

For more information about the default global cache topology, see “Data caching overview” on page 142.

About this task

You can configure the global cache so that a multi-instance broker hosts up to 4 container servers. If the active broker instance fails, the global cache switches automatically to use the container servers in the standby broker instance.

Consider the following example. Your global cache consists of 2 brokers that host catalog servers and container servers, and a multi-instance broker. The active instance of the multi-instance broker hosts up to 4 container servers. If the active instance of the multi-instance broker fails, the cache will remain operational as long as at least one of the catalog servers is still available. Data is temporarily rebalanced across the remaining container servers in the brokers that host the catalog servers. When the standby instance of the multi-instance broker starts, the container servers rejoin the global cache, and cached data is rebalanced automatically.

A multi-instance broker cannot host a catalog server. Therefore, you cannot configure an integration server to host a catalog server if that integration server is defined with multiple listener hosts.

A sample XML policy file is provided as a starting point for your configuration. The `policy_multi_instance.xml` file configures three brokers in a high availability scenario. Two brokers each host a catalog server, and a multi-instance broker hosts two container servers.

To configure a multi-instance broker, a **listenerHost** element has been introduced as an alternative to the `listenerHost` attribute of the **broker** element. You can use the **listenerHost** element to specify a list of listener hosts. Alternatively, you can set the **listenerHost** property on the integration server to a comma-separated list of listener hosts.

The following steps describe how to configure the global cache for a multi-instance broker.

Procedure

1. Copy the sample policy file, `policy_multi_instance.xml`, from `install_dir/sample/globalcache` to another location on your file system.
Do not edit the sample policy file in its original location; copy it to your own file system first. The original sample policy file might be replaced when you apply maintenance to IBM Integration Bus. You can place a copy of the same policy file on each computer where a broker is running, or you can provide a single copy of the policy file in a shared file system for all brokers to access. No matter how the file is shared between two computers, the policy file must be placed on the same file path on each computer or shared system.
2. Modify the policy file for your system, specifying the appropriate broker names and listener hosts, the port range that the broker is to use, and how many catalog servers the broker hosts. Optionally, you can also specify a domain name for all catalog servers in the embedded cache. If you do not set a domain name, the broker creates one.

Ensure that the policy meets the following criteria:

- You can define 0, 1, or 2 catalog servers for an individual broker, but at least one catalog server must be defined in the policy.
- A multi-instance broker cannot host a catalog server.
- You cannot specify the **listenerHost** element more than once for a broker that is configured to host one or more catalog servers.
- If two brokers share a host name, you must set a distinct port range for each broker.
- Ensure that the port range for each broker includes at least 20 ports.

- The broker names and listener hosts specified in the policy must match the values defined for the brokers.
- You must specify either the **listenerHost** element or the `listenerHost` attribute for each broker.
- You can define only one domain name in the policy file.
- If specified, the domain name must precede the broker elements in the policy file.
- The policy file must be encoded in UTF-8.
- The policy file must contain valid XML. The policy file is validated against an XML schema when you set the broker-level property. You can also validate the policy file against the copy of the schema (`policy.xsd`) that is provided in `install_dir/cachesupport/schema`.

When you use an XML policy file, the broker-level **portRange** property is ignored. The port range specified in the XML file overrides the property specified for the broker.

3. Save the policy file.

4. Set the cache policy to the fully qualified name of the policy file.

The path that you specify must be absolute, not relative. If you use a shared drive on Windows, you must use the `\\hostname\directory` path syntax to the shared drive, instead of a mapped drive letter. The IBM Integration Bus user ID that is used to access the `\\hostname\directory` path must have read access to the file system and must use the same password.

You can set the cache policy by using commands (see “Configuring the embedded global cache by using commands” on page 164) or IBM Integration Explorer (see “Configuring the embedded global cache by using IBM Integration Explorer” on page 167).

5. Restart each broker.

Results

When each broker restarts, it uses the values in the policy file to determine its cache properties. If multiple listener hosts are specified, the global cache tries to bind to each one in turn until it finds one that is available on the system. If the global cache does not find a listener host that is available on the system, it uses the first listener host in the list.

Each broker contains up to 4 container servers. To find out where container servers are placed, use the **mqsicacheadmin** command to run the **showPlacement** command, as shown in the following example:

```
mqsicacheadmin brokerName -c showPlacement
```

You can also use the **mqsicacheadmin** command to show cache components in a multi-broker cache. For example, the **listHosts** command shows the host names, number of hosts, and number of catalogs in the cache:

```
mqsicacheadmin brokerName -c listHosts
```

Related concepts:

“Data caching overview” on page 142

WebSphere eXtreme Scale provides IBM Integration Bus with data caching capability.

“Data caching terminology” on page 155

The global cache is embedded in the broker. You can also connect to an external WebSphere eXtreme Scale grid.

“Global cache scenario: Storing state for integrations” on page 144
You can use a global cache to store state for integrations. With a global cache, each broker can handle replies, even when the request was processed by another broker.

“Global cache scenario: Caching static data” on page 147
You can use a global cache to store static data. The use of a global cache facilitates horizontal scaling in situations where a cache is used to minimize network interactions to a back end system. With a global cache, you can increase the number of clients while maintaining a predictable response time for each client.

Related tasks:

“Configuring the embedded global cache” on page 162
Configure properties of the embedded global cache by using commands, IBM Integration Explorer, an XML policy file, or the IBM Integration API.

“Managing data caching” on page 141
Store data that you want to reuse by using the embedded global cache or an external WebSphere eXtreme Scale grid.



Accessing the global cache with a JavaCompute node
You can use a JavaCompute node to interact with a map in a global cache or an external WebSphere eXtreme Scale grid.

“Monitoring the global cache” on page 183
You can use the `mqsicacheadmin` command, resource statistics, and the activity log to monitor the global cache. You can use resource statistics and the activity log to monitor external grids.

Related reference:



Parameter values for the cachemanager component
Select the objects and properties that are associated with the global cache that you want to change.



JavaCompute node
Use the JavaCompute node to work with messages by using the Java language.
[WebSphere eXtreme Scale product documentation](#)
[WebSphere eXtreme Scale product web page](#)

Connecting to external WebSphere eXtreme Scale grids

About this task

For concept information about WebSphere eXtreme Scale grids, see the following topic:

- “WebSphere eXtreme Scale grids” on page 153

For detailed information about how to connect to an external WebSphere eXtreme Scale grid, and how to enable SSL on those connections, see the following topics:

- “Connecting to a WebSphere eXtreme Scale grid” on page 179
- “Enabling SSL for external WebSphere eXtreme Scale grids” on page 181

Related concepts:

“Data caching terminology” on page 155
The global cache is embedded in the broker. You can also connect to an external WebSphere eXtreme Scale grid.

Related tasks:

“Managing data caching” on page 141

Store data that you want to reuse by using the embedded global cache or an external WebSphere eXtreme Scale grid.

“Monitoring the global cache” on page 183

You can use the `mqsicacheadmin` command, resource statistics, and the activity log to monitor the global cache. You can use resource statistics and the activity log to monitor external grids.



Accessing the global cache with a JavaCompute node

You can use a JavaCompute node to interact with a map in a global cache or an external WebSphere eXtreme Scale grid.

Related reference:



Parameter values for the cachemanager component

Select the objects and properties that are associated with the global cache that you want to change.

WebSphere eXtreme Scale grids:

Use one or more external WebSphere eXtreme Scale grids to store data that you want to reuse.

WebSphere eXtreme Scale provides a scalable, in-memory data grid. The data grid dynamically caches, partitions, replicates, and manages data across multiple servers. The catalog servers and container servers for the IBM Integration Bus global cache collaborate to act as a WebSphere eXtreme Scale grid. For more information about the grid that is embedded in the broker, see “Embedded global cache” on page 150.

In addition to the grid that is available (as the embedded global cache) within IBM Integration Bus, you can integrate with WebSphere eXtreme Scale grids that are running elsewhere. You can work with multiple external grids, and the embedded grid, at the same time. For a diagram that shows how IBM Integration Bus message flows interact with an external WebSphere eXtreme Scale grid, and for definitions of the terms that are associated with a grid, see “Data caching terminology” on page 155. For more information about grids, see WebSphere eXtreme Scale product documentation.

You might have an existing WebSphere eXtreme Scale environment that uses a topology such as one of the following examples:

- A DataPower XC10 device
- Embedded WebSphere eXtreme Scale servers in WebSphere Application Server network deployment installations
- A stand-alone WebSphere eXtreme Scale software grid

When you are connecting to an external grid, IBM Integration Bus hosts the WebSphere eXtreme Scale client, but the cache is hosted on an external grid. Therefore, a separate WebSphere eXtreme Scale installation or appliance is required, such as a DataPower XC10 device. An installation or appliance can host more than one grid.

To connect to an external grid, you need the following information:

- The name of the grid
- The host name and port of each catalog server for the grid
- Optional: the object grid file that can be used to override WebSphere eXtreme Scale properties

By using a configurable service to specify the parameters, you can connect to an external WebSphere eXtreme Scale grid. IBM Integration Bus message flows can then access and modify data that is stored in the external grid. If you are connecting to a secure grid, you can create a security identity by using the **mqsisetdbparms** command. To connect to an external grid, follow the instructions in “Connecting to a WebSphere eXtreme Scale grid” on page 179.

You can enable SSL for client connections to external WebSphere eXtreme Scale grids by setting up a public key infrastructure, then enabling SSL for an integration server. For more details, see “Enabling SSL for external WebSphere eXtreme Scale grids” on page 181.

You can use JavaCompute nodes to store and retrieve data in a map in the external grid. When you get a global map from an external grid, the `getGlobalMap` method makes a connection to the grid if one does not exist. For instructions, see Accessing the global cache with a JavaCompute node.

You can monitor the external grid by viewing the activity log and resource statistics. For more information, see “Monitoring the global cache” on page 183.

To enable new function that is available in fix packs, you use the **-f** parameter on the **mqsichangebroker** command. Running this command also enables the specification of domain names. In previous versions of IBM Integration Bus, the domain name for all WebSphere eXtreme Scale servers in all embedded caches was an empty string. Servers in different domains cannot collaborate in the same grid. Therefore, for a cache that spans more than one broker, you must enable the new capability for these brokers at the same time.

Related concepts:

“Data caching terminology” on page 155

The global cache is embedded in the broker. You can also connect to an external WebSphere eXtreme Scale grid.

“Global cache scenario: Storing state for integrations” on page 144

You can use a global cache to store state for integrations. With a global cache, each broker can handle replies, even when the request was processed by another broker.

“Global cache scenario: Caching static data” on page 147

You can use a global cache to store static data. The use of a global cache facilitates horizontal scaling in situations where a cache is used to minimize network interactions to a back end system. With a global cache, you can increase the number of clients while maintaining a predictable response time for each client.

Related tasks:

“Configuring the embedded global cache” on page 162

Configure properties of the embedded global cache by using commands, IBM Integration Explorer, an XML policy file, or the IBM Integration API.

“Connecting to a WebSphere eXtreme Scale grid” on page 179

You can store data in a WebSphere eXtreme Scale grid for use by other message flows.

“Managing data caching” on page 141

Store data that you want to reuse by using the embedded global cache or an external WebSphere eXtreme Scale grid.



Accessing the global cache with a JavaCompute node

You can use a JavaCompute node to interact with a map in a global cache or an external WebSphere eXtreme Scale grid.

Related reference:



Parameter values for the cachemanager component

Select the objects and properties that are associated with the global cache that you want to change.



JavaCompute node

Use the JavaCompute node to work with messages by using the Java language.



mqsichangebroker command

Use the **mqsichangebroker** command to change one or more of the configuration parameters of the broker.

WebSphere eXtreme Scale product documentation

WebSphere eXtreme Scale product web page

Connecting to a WebSphere eXtreme Scale grid:

You can store data in a WebSphere eXtreme Scale grid for use by other message flows.

Before you begin

Before you start:

- Read the concept information about external grids in “WebSphere eXtreme Scale grids” on page 153.
- Ask your WebSphere eXtreme Scale administrator for the following information about the external grid to which you are connecting:
 - The name of the grid
 - The host name and port of each catalog server for the grid
 - Optional: The object grid file that can be used to override WebSphere eXtreme Scale properties
- To enable SSL for connections to external WebSphere eXtreme Scale grids, follow the instructions in “Enabling SSL for external WebSphere eXtreme Scale grids” on page 181.

About this task

The following steps describe how to connect to a WebSphere eXtreme Scale grid by using a configurable service.

Procedure

1. Optional: If you are connecting to a secure grid, use the **mqsisetdbparms** command to create a security identity, as shown in the following example.
mqsisetdbparms IB9NODE -n wxs::idl -u userId -p password
For more information, see **mqsisetdbparms** command.
2. Create a WXSServer configurable service by using the **mqsicreateconfigurable-service** command or IBM Integration Explorer.
For more information, see **mqsicreateconfigurable-service** command or Creating a new configurable service.
3. Use the values provided by your WebSphere eXtreme Scale administrator to configure the connection to the external grid, as shown in the following example. For an explanation of the parameters of the configurable service, see Configurable services properties.

```
mqsicreateconfigurableservice IB9NODE -c WXSServer -o xc10
-n catalogServiceEndPoints,gridName,overrideObjectGridFile,securityIdentity
-v "server.ibm.com:2809","myGrid","C:\Brokers\WebSphere_eXtreme_Scale\xc10\xc10Client.xml","id1"
```

As you can see from this example, when you specify the security identity, you omit the prefix (wxs::) that you used when creating the security identity with the **mqsisetdbparms** command.

If the object grid file is on a shared drive on Windows, you must use the \\hostname\directory path syntax to the shared drive, instead of a mapped drive letter. The IBM Integration Bus user ID that is used to access the \\hostname\directory path must have read access to the file system and must use the same password that is required to access the shared drive.

What to do next

After you create the WXSServer configurable service, you can interact with the external grid by using a JavaCompute node. (For detailed instructions, see [Accessing the global cache with a JavaCompute node.](#)) When you use a JavaCompute node to get a global map from an external grid, the `getGlobalMap` method makes the connection to the grid if one does not exist. To modify a flow that previously used the embedded cache to connect to an external grid instead, you must update your JavaCompute node by specifying the name of the map on the external grid and the name of the configurable service that is used to connect to the grid. You can work with multiple external grids, and the embedded grid, at the same time.

Statistics are recorded for the interactions between message flows and the external grid. The global cache activity log also provides a high-level overview of how IBM Integration Bus interacts with the external grid. For more information, see [“Monitoring the global cache”](#) on page 183.

Related concepts:

[“Data caching terminology”](#) on page 155

The global cache is embedded in the broker. You can also connect to an external WebSphere eXtreme Scale grid.

Related tasks:

[“Configuring the embedded global cache”](#) on page 162

Configure properties of the embedded global cache by using commands, IBM Integration Explorer, an XML policy file, or the IBM Integration API.

[“Managing data caching”](#) on page 141

Store data that you want to reuse by using the embedded global cache or an external WebSphere eXtreme Scale grid.



[Accessing the global cache with a JavaCompute node](#)

You can use a JavaCompute node to interact with a map in a global cache or an external WebSphere eXtreme Scale grid.

Related reference:



[Configurable services properties](#)

The supplied configurable services, and the configurable services that you create, are defined by their names and properties. You can use the supplied services.



[mqsicreateconfigurableservice](#) command

Use the **mqsicreateconfigurableservice** command to create an object name for a broker external resource.



mqsisetdbparms command

Use the **mqsisetdbparms** command to associate a specific user ID and password (or SSH identity file) with one or more resources that are accessed by the broker.

WebSphere eXtreme Scale product documentation

WebSphere eXtreme Scale product web page

Enabling SSL for external WebSphere eXtreme Scale grids:

Enable SSL for an external WebSphere eXtreme Scale grid by setting up a public key infrastructure, then enabling SSL on the integration server.

Before you begin

Before you start:

Read the concept information in “WebSphere eXtreme Scale grids” on page 153 and “Public key cryptography” on page 206.

About this task

You can enable SSL for client connections to external WebSphere eXtreme Scale grids. You cannot enable SSL for servers in the embedded global cache.

To enable SSL communication, configure the keystore, truststore, passwords, and certificates. To enable server authentication, import the public certificate from the WebSphere eXtreme Scale server into the broker or integration server truststore. If the server requires client authentication, you must also create a private key in the broker or integration server keystore that the WebSphere eXtreme Scale server trusts.

You then set properties on the integration server to enable SSL and specify the required protocol. You can also nominate a particular key to use if you have more than one. SSL connections can be made only from integration servers that are not hosting catalog or container servers.

The following steps describe how to enable SSL for an external WebSphere eXtreme Scale grid.

Procedure

1. Set up a public key infrastructure by following the instructions in “Setting up a public key infrastructure” on page 415. You can set up the public key infrastructure at broker or integration server level.

Connections to external WebSphere eXtreme Scale grids cannot implicitly use public certificates that are located in the JVM cacerts file.

2. To ensure that you are enabling SSL on an integration server that does not host a catalog or container server, use one of these methods:

- Set the broker-level policy to none to specify roles for the integration servers manually. For example, to ensure that an integration server does not host a catalog or container server, run the following **mqsichangeproperties** command:

```
mqsichangeproperties broker_name -e execution_group_1 -o ComIbmCacheManager -n enableCatalogService,enableContainerService -v false,false
```

- Set the broker-level policy to disabled to switch off the embedded global cache and ensure that no integration servers are hosting WebSphere eXtreme Scale server components.

To set the broker-level policy, use one of the following methods:

- From the command line, run the following command, setting the **-b** parameter to none or disabled:
`mqsichangebroker brokerName -b none`
- From the IBM Integration Explorer, right-click the appropriate broker, click **Properties**, then **Global Cache**, then set the cache policy to none or disabled.

3. Optional: If you set the broker-level policy to none, check that the **enableCatalogService** and **enableContainerService** properties are set to false for each integration server for which you are enabling SSL.

Use one of the following methods:

- From the command line, run the following command, then check that both properties are set to false:
`mqsireportproperties brokerName -e integrationServerName -o ComIbmCacheManager -r`
- From the IBM Integration Explorer, select the properties for each integration server, then select **Global Cache**. Confirm that the **Catalog server enabled** and **Container Server enabled** properties are not selected.

4. To enable SSL, set the following properties on the appropriate integration server:

- To enable SSL, set **clientsDefaultToSSL** to true.
- Optional: To specify an SSL protocol, set **sslProtocol** to a value that is recognized by the IBM JSSE2 security provider.
- Optional: If the external grid requires client authentication and you have more than one trusted private key in the broker keystore, set **sslAlias** to the appropriate key.

For more information about these properties, see Parameter values for the cachemanager component.

To set these properties on the integration server, use one of the following methods:

- From the command line, run the **mqsichangeproperties** command, as shown in the following example:
`mqsichangeproperties broker_name -e execution_group_1 -o ComIbmCacheManager -n clientsDefaultToSSL,sslProtocol,sslAlias -v true,SSL_TLS,ProdKey`
- From the IBM Integration Explorer, select the properties for each integration server, then select **Global Cache**. Select **clientsDefaultToSSL** and, if required, set the SSL protocol and SSL key alias.

5. Restart the integration server. For more information, see **mqsireload** command.
6. Connect to the WebSphere eXtreme Scale grid by following the instructions in “Connecting to a WebSphere eXtreme Scale grid” on page 179.

Results

Keystore, truststore, and protocol settings are verified the first time that a connection is made from the integration server (either to the embedded grid, or for the first remote connection). Errors in the configuration are reported as a warning, and SSL connections are then prohibited. For example, a warning is issued if a keystore file is not found, the file is corrupted, or the keystore password is incorrect.

If you enable SSL and try to connect from an integration server that hosts WebSphere eXtreme Scale server components, the connection fails with a detailed exception message, BIP7144, which explains why the connection failed. If an SSL

handshake exception occurs, the message flow fails and the exception message BIP7147 is issued.

Related concepts:

“Public key cryptography” on page 206

All encryption systems rely on the concept of a key. A key is the basis for a transformation, usually mathematical, of an ordinary message into an unreadable message. For centuries, most encryption systems have relied on private key encryption. Public key encryption is the only challenge to private key encryption that has appeared in the last 30 years.

“WebSphere eXtreme Scale grids” on page 153

Use one or more external WebSphere eXtreme Scale grids to store data that you want to reuse.

 **IBM Integration Explorer**

The IBM Integration Explorer is a graphical user interface based on the Eclipse platform for administering your integration nodes (brokers).

Related tasks:

“Setting up a public key infrastructure” on page 415

Configure keystores, truststores, passwords, and certificates to enable SSL communication, and Web Services Security.

“Connecting to a WebSphere eXtreme Scale grid” on page 179

You can store data in a WebSphere eXtreme Scale grid for use by other message flows.

“Configuring the embedded global cache” on page 162

Configure properties of the embedded global cache by using commands, IBM Integration Explorer, an XML policy file, or the IBM Integration API.

Related reference:

 **Parameter values for the cachemanager component**

Select the objects and properties that are associated with the global cache that you want to change.

 **mqsichangebroker** command

Use the **mqsichangebroker** command to change one or more of the configuration parameters of the broker.

 **mqsichangeproperties** command

Use the **mqsichangeproperties** command to modify broker properties and properties of broker resources.

 **mqsireportproperties** command

Use the **mqsireportproperties** command to display properties that relate to a broker, an integration server, or a configurable service.

Monitoring the global cache

You can use the **mqsicacheadmin** command, resource statistics, and the activity log to monitor the global cache. You can use resource statistics and the activity log to monitor external grids.

About this task

The following tools provide information about the global cache and how it is performing.

mqsicacheadmin command

The **mqsicacheadmin** command provides information about the global cache that is embedded in a broker. For example, you can find out the size of a map and list the hosts that are participating in the cache. You can also use this command to clear data from a map. See “Running the **mqsicacheadmin** command.”

Activity log

Activity logs provide a high-level overview of how IBM Integration Bus interacts with external resources, therefore helping you to understand what your message flows are doing. See “Viewing the activity log.”

Resource statistics

Resource statistics are collected by a broker to record performance and operating details of resources that are used by integration servers. See “Collecting resource statistics” on page 185.

Running the mqsicacheadmin command:

About this task

To use the command to find information about the global cache that is embedded in a broker, complete the following steps.

Procedure

1. Before you run the **mqsicacheadmin** command, ensure that the broker is running and that the global cache is available.

The cache is disabled by default. To use the cache, select a broker-level cache policy (see “Configuring the embedded global cache” on page 162).

2. Run the **mqsicacheadmin** command, specifying the appropriate parameters, as described in **mqsicacheadmin** command. Use the **-c** parameter to run a WebSphere eXtreme Scale command.

For example, to list all container servers and their shards in the embedded cache, run the WebSphere eXtreme Scale command, **showPlacement**:

```
mqsicacheadmin myBroker -c showPlacement
```

To clear all data from the map called *myMap* in the broker *myBroker*, run the following command:

```
mqsicacheadmin myBroker -c clearGrid -m myMap
```

The data is cleared immediately; you do not need to restart the broker for this action to take effect.

You do not have to specify a broker name with this command. For example, use the **-cep** parameter to connect to a catalog server and show the routing table for each WebSphere eXtreme Scale shard:

```
mqsicacheadmin -cep server.company.com:2800 -c routetable
```

3. Examine the output and usage information that is returned by the command.

Viewing the activity log:

About this task

To open the activity log view in IBM Integration Explorer, complete the following steps.

Procedure

1. Ensure that the global cache is available.

The cache is disabled by default. To use the cache, select a broker-level cache policy (see “Configuring the embedded global cache” on page 162).

2. In the **WebSphere MQ Explorer - Navigator** view, expand the Integration Nodes folder, then expand the folder for the relevant integration server.
3. To view the activity log for either a message flow or the global cache resource type, complete one of the following steps.
 - To open the activity log for a message flow, right-click the message flow and click **Open Activity Log**.
 - To open the activity log for the global cache, expand the **Resource Managers** folder, right-click the global cache resource type, then click **Open Activity Log**.

Activity information is displayed for the cache or the message flow. For more information about the activities that are recorded for the global cache, see Global cache Activity log.

4. You can save or copy information in the activity log; for more information, see “Working with Activity logs in IBM Integration Explorer” on page 625.

Collecting resource statistics:

About this task

To view resource statistics for the global cache, complete the following steps.

Procedure

1. Ensure that the global cache is available.

The cache is disabled by default. To use the cache, select a broker-level cache policy (see “Configuring the embedded global cache” on page 162).

2. To start collecting resource statistics, use one of the following methods.

For more information about working with resource statistics, see “Analyzing resource performance” on page 529.

- Ensure that the broker is running, then run the **mqsichangeresourcestats** command, setting the **-c** parameter to active.

For example, to start collecting resource statistics for integration server *myIS* on broker *myBroker*, run the following command.

```
mqsichangeresourcestats myBroker -c active -e myIS
```

For more information about this command, see **mqsichangeresourcestats** command.

- In the **WebSphere MQ Explorer - Navigator** view of IBM Integration Explorer, expand the **Brokers** folder. Right-click one or more integration servers and click **Statistics > Start Resource Statistics**.
3. To view the statistics, use IBM Integration Explorer.

For more information about the data that is collected for the global cache, see Resource statistics data: global cache.

 - a. In IBM Integration Explorer, click **Window > Show View > Resource Statistics**. The Resource Statistics and Resource Statistics Graph views open.
 - b. In the Resource Statistics view, click the global cache tab.

If you are displaying statistics for this integration server for the first time, the views might be empty until the first data is received. Update messages are sent every 20 seconds, and the views refresh automatically.

Related concepts:

“Data caching overview” on page 142

WebSphere eXtreme Scale provides IBM Integration Bus with data caching capability.

“Resource statistics” on page 530

Resource statistics are collected by a broker to record performance and operating details of resources that are used by integration servers.

Related tasks:

“Managing data caching” on page 141

Store data that you want to reuse by using the embedded global cache or an external WebSphere eXtreme Scale grid.

Chapter 3, “Performance, monitoring, and workload management,” on page 449

You can change various aspects of your broker configuration to tune brokers and message flows, and monitor message flows and publish/subscribe applications.

Related reference:



mqsicacheadmin command

Use the **mqsicacheadmin** command to provide information about the global cache and complete basic administration.



Global cache Activity log

The global cache activity log provides a high-level overview of how IBM Integration Bus interacts with the global cache or external grid so that you can understand what your message flows are doing.



Resource statistics data: global cache

Learn about the data that is returned for the GlobalCache resource type when you activate resource statistics collection.



Activity logs

Activity logs contain information about message flows and how they interact with external resources. They can be generated for a message flow or a resource type. Learn about the structure and content of Activity logs, and about the *tags* that enrich the log data and can be used to filter its content.

Administering Java applications

Manage the Java applications that are deployed to a broker.

About this task

You can deploy message flows that contain Java applications to a broker. The Java code is run within a JVM that is created by the integration server. The JVM runs in the same process as other broker components such as message parsing and nodes that are not Java-based.

Administration of Java applications includes the following tasks:

- “Tuning JVM parameters” on page 187
- “Configuring classloaders for Java user-defined nodes” on page 187
- “Configuring classloaders for JavaCompute nodes” on page 187
- “Configuring classloaders for ESQL routines” on page 188

Tuning JVM parameters

About this task

Use the `mqsichangeproperties` command to tune the JVM parameters to ensure that there are sufficient resources for all the Java applications that are deployed to the integration server. For more details, see `mqsichangeproperties` command and JVM parameter values.

Configuring classloaders for Java user-defined nodes

About this task

Java user-defined nodes are manually installed onto a broker as either a PAR file or a JAR file. Because PAR and JAR files are only loaded by the broker on startup, the broker must be restarted. For more details, see [Packaging a Java user-defined node](#).

A PAR file is given its own Java classloader, ensuring that the node classes are isolated from any other node classes.

For more details, see [User-defined node class loading](#).

Configuring classloaders for JavaCompute nodes

About this task

A JavaCompute node is deployed to an integration server as part of a BAR file. A JavaCompute node can specify a `JavaClassLoader` configurable service to be used by the node. A `JavaClassLoader` configurable service defines the behavior of the classloaders that are used by the node. For more details, see [JavaCompute node classloading](#).

If a JavaCompute node specifies a `JavaClassLoader` configurable service, you must define a configurable service with the name specified by the node on the broker. For more details, see [JavaCompute node classloading using a configurable service](#).

Related concepts:



The IBM Integration Bus environment

An integration node (broker) is a set of execution processes that hosts one or more message flows to route, transform, and enrich in flight messages.



JavaCompute node classloading

Details the default Java classloader options and the precedence order of each type.



JavaCompute node classloading using a configurable service

Details alternative configurable Java classloader options and the precedence order of each type.



User-defined node class loading

Details the Java classes packaging options and loading order precedence for user-defined nodes.

Related tasks:

Chapter 1, “Administering brokers and broker resources,” on page 1

Administering brokers and associated integration node resources includes the tasks that you perform frequently to activate and manage those resources. Choose the method you prefer to administer your brokers and associated resources.

Related reference:



JavaCompute node

Use the JavaCompute node to work with messages by using the Java language.

Configuring classloaders for ESQL routines

About this task

You can identify Java methods to invoke from ESQL routines by using the CREATE FUNCTION statement or the CREATE PROCEDURE statement with a LANGUAGE clause of JAVA. You use the EXTERNAL NAME clause of the statement to specify the fully-qualified class and name of the method. To find the Java class that contains a method, the broker uses the search algorithm that is described in Deploying Java classes. You can optionally specify a JavaClassLoader configurable service when you identify a Java method in this way. The JavaClassLoader configurable service defines the behavior of the classloader that is used to load the class specified in the EXTERNAL NAME clause. If you include a CLASSLOADER clause in a CREATE FUNCTION statement or a CREATE PROCEDURE statement, you must ensure that the corresponding configurable service is defined on the broker. For more details, see JavaClassLoader configurable service.

ESQL Java routines and JavaCompute nodes that specify the same JavaClassLoader configurable service share one instance of the classloader. Therefore, the nodes and the routines use the same in-memory version of Java classes, and have access to the same static variables. The classloading mechanism for ESQL routines is the same as for JavaCompute nodes; for more details, see JavaCompute node classloading using a configurable service.

Related concepts:



ESQL modules

A module is a sequence of declarations that define variables and their initialization, and a sequence of subroutine (function and procedure) declarations that define a specific behavior for a message flow node.



JavaCompute node classloading

Details the default Java classloader options and the precedence order of each type.



JavaCompute node classloading using a configurable service

Details alternative configurable Java classloader options and the precedence order of each type.

Related reference:



CREATE FUNCTION statement

The CREATE FUNCTION statement defines a callable function or procedure.



CREATE PROCEDURE statement

The CREATE PROCEDURE statement defines a callable function or procedure.



JavaClassLoader configurable service

Select the objects and properties that you want to change for the JavaClassLoader configurable service.

Managing JMS resources

Manage the JMS resources associated with your message flows.

About this task

You can deploy message flows that use JMS resources to a broker. The IBM Integration Bus built-in JMS nodes support the exchange of JMS messages and behave like JMS clients.

Administration of JMS resources includes the following tasks:

- “Understanding behavior using Activity log”
- “Using configurable services”
- “Monitoring JMS resource statistics”

Understanding behavior using Activity log

About this task

Use the Activity log for an overview of activity in your message flows and JMS resources. The Activity log can give you a high-level overview to understand recent activities affecting your system and associated external resources, for example, a lost JMS connection. You do not need to enable Activity log, so it can provide an early indication of potential problems.

For more details, see “Using Activity logs” on page 623.

Using configurable services

About this task

Use JMS provider configurable services to interface with proprietary JMS API calls from some providers that differ from the standard JMS specification.

For more details, see *Configuring for JMS*.

Monitoring JMS resource statistics

About this task

Use the resource statistics to check that your systems are using JMS resources efficiently. The JMS resource statistics allow you to monitor the number of JMS connections, sessions, messages and connection failures. You can monitor your system resources to keep the usage within boundaries you consider acceptable.

For more details, see “Resource statistics” on page 530.

Related concepts:

“Resource statistics” on page 530

Resource statistics are collected by a broker to record performance and operating details of resources that are used by integration servers.

Related tasks:

Chapter 1, “Administering brokers and broker resources,” on page 1

Administering brokers and associated integration node resources includes the tasks that you perform frequently to activate and manage those resources. Choose the method you prefer to administer your brokers and associated resources.

“Using Activity logs” on page 623

Use Activity logs to get an overview of recent activities in your message flows and associated external resources.

Related reference:

 JMSInput node

Use the JMSInput node to receive messages from JMS destinations. JMS destinations are accessed through a connection to a JMS provider.

 Using JMS in IBM Integration Bus

Use IBM Integration Bus to connect to applications that send and receive messages that conform to the Java Message Service (JMS) standard.

Accessing Administration log information

Active brokers record information about their operations in the Administration log. Access the events that are written by the broker by using the IBM Integration Explorer and the IBM Integration Toolkit. For some actions, you can also use the IBM Integration API (also known as the CMP).

About this task

- “Viewing Administration log information”
- “Saving Administration log information” on page 192
- “Clearing Administration log information” on page 193
- “Changing Administration Log view preferences” on page 194

Related concepts:

 The IBM Integration Bus environment

An integration node (broker) is a set of execution processes that hosts one or more message flows to route, transform, and enrich in flight messages.

Related reference:

 Administration Log view

The Administration Log view shows administration requests and their results, changes made to objects, and the result of deployment actions on brokers. The Administration Log view can also describe configuration changes that have been automatically applied as a result of moving the broker from one version to another; for example, after applying maintenance.

Related information:

 IBM Integration API (CMP)

Viewing Administration log information

You can view Administration log information by using either the IBM Integration Explorer, or the CMP.

About this task

Follow the instructions in this topic to use the IBM Integration Explorer. If you prefer to use the CMP, see “Developing applications that use the IBM Integration API” on page 58 and IBM Integration API.

Administration log information is written to the Administration Log view in the IBM Integration Explorer. The Administration Log view displays actions performed

on the integration node (broker) by all users. Alternatively, you can view the results of deployment actions from a single user using the Deployment Log view in the IBM Integration Toolkit, see Deployment Log view.

The Administration Log view contains information about events that occur within your integration nodes. These events can be information, errors, or warnings and relate to your own actions. To view events for a particular integration node, look for the name of the integration node in the Source column.

Each event contains the following information:

- Message: The event number.
- Source: Where the event has come from.
- TimeStamp: The date and time that the event occurred. Time stamps are taken from the computer that is hosting the integration node.
- Details: What has caused the event and what action is needed to rectify it.

You can only view Administration log information for a specific integration node using the IBM Integration Explorer if the integration node is both running and connected. Local integration nodes are automatically connected, but you must manually connect to remote integration nodes.

Procedure

1. To view Administration log information in the IBM Integration Explorer, ensure that the Administration Log view is visible. If the Administration Log view is not visible, from the menu, click **Window > Show View > Administration Log**.
2. Expand the Integration Nodes folder, and select the integration node with which you want to work. The Administration log information for the selected integration node is displayed in the Administration Log view. The Administration log information is displayed in time and date order.
3. Double-click the message to display the full details for a message. The message is opened in a new window.

Related concepts:



IBM Integration Explorer

The IBM Integration Explorer is a graphical user interface based on the Eclipse platform for administering your integration nodes (brokers).



The IBM Integration Bus environment

An integration node (broker) is a set of execution processes that hosts one or more message flows to route, transform, and enrich in flight messages.



The IBM Integration API

The IBM Integration API is a programming interface that your applications can use to control brokers and their resources through a remote interface.

Related tasks:

“Clearing Administration log information” on page 193

Clear Administration log information to reduce the size of the log by using either the IBM Integration Explorer or the CMP.

“Saving Administration log information” on page 192

Save the Administration log information that is written to the Administration Log view in the IBM Integration Explorer.

Related reference:

Administration Log view

The Administration Log view shows administration requests and their results, changes made to objects, and the result of deployment actions on brokers. The Administration Log view can also describe configuration changes that have been automatically applied as a result of moving the broker from one version to another; for example, after applying maintenance.

Saving Administration log information

Save the Administration log information that is written to the Administration Log view in the IBM Integration Explorer.

About this task

Administration log information is deleted automatically from the integration node (broker) when the integration node restarts. You can save the log contents to file if you want to retain them. You cannot save the Administration log information from the Deployment Log view in the IBM Integration Toolkit.

Procedure

1. In the IBM Integration Explorer, expand the Integration Nodes folder in the Navigator view.
2. Select the integration node with which you want to work, to display the Administration log information in the Administration Log view.
3. Right-click in the Administration Log view, and click **Save Log As**.
4. Enter an appropriate directory in which to save the log information.
5. Enter a name for the log file, and click **Save Log**.

Results

Each message recorded in the Administration log is written to the text file with the same information that is detailed in the Administration log itself.

To view the saved log, open the log file in an appropriate text editor.

Related concepts:

IBM Integration Explorer

The IBM Integration Explorer is a graphical user interface based on the Eclipse platform for administering your integration nodes (brokers).

The IBM Integration Bus environment

An integration node (broker) is a set of execution processes that hosts one or more message flows to route, transform, and enrich in flight messages.

Related tasks:

“Viewing Administration log information” on page 190

You can view Administration log information by using either the IBM Integration Explorer, or the CMP.

“Clearing Administration log information” on page 193

Clear Administration log information to reduce the size of the log by using either the IBM Integration Explorer or the CMP.

Related reference:



Administration Log view

The Administration Log view shows administration requests and their results, changes made to objects, and the result of deployment actions on brokers. The Administration Log view can also describe configuration changes that have been automatically applied as a result of moving the broker from one version to another; for example, after applying maintenance.

Clearing Administration log information

Clear Administration log information to reduce the size of the log by using either the IBM Integration Explorer or the CMP.

About this task

Follow the instructions in this topic to use the IBM Integration Explorer to clear the log. If you prefer to use the CMP, see “Developing applications that use the IBM Integration API” on page 58 and IBM Integration API. You can empty the contents of the Deployment Log view in the IBM Integration Toolkit by right-clicking in the Deployment Log view, and clicking **Clean Log**. Emptying the contents of the Deployment Log view does not remove the entries from the Administration log.

To clear all the Administration log information from the Administration log:

Procedure

1. Expand the Integration Nodes folder in the Navigator view.
2. Select the integration node (broker) with which you want to work, to display the Administration log information in the Administration Log view.
3. Right-click in the Administration Log view, and click **Clear Administration Log**.

If you have set the IBM Integration Explorer preference to *warn before deleting log events*, a prompt asks you to confirm deletion. Click **OK**.

If you have not set the IBM Integration Explorer preference to *warn before deleting log events*, the Administration log is cleared automatically.

All log entries will be deleted for this user.

Related concepts:



IBM Integration Explorer

The IBM Integration Explorer is a graphical user interface based on the Eclipse platform for administering your integration nodes (brokers).



The IBM Integration Bus environment

An integration node (broker) is a set of execution processes that hosts one or more message flows to route, transform, and enrich in flight messages.

Related tasks:

“Viewing Administration log information” on page 190

You can view Administration log information by using either the IBM Integration Explorer, or the CMP.

“Saving Administration log information” on page 192

Save the Administration log information that is written to the Administration Log view in the IBM Integration Explorer.

Related reference:

Administration Log view

The Administration Log view shows administration requests and their results, changes made to objects, and the result of deployment actions on brokers. The Administration Log view can also describe configuration changes that have been automatically applied as a result of moving the broker from one version to another; for example, after applying maintenance.

Related information:

IBM Integration API (CMP)

Changing Administration Log view preferences

You can change preferences for the Administration Log view by using the **IBM Integration** preferences page in the IBM Integration Explorer.

About this task

You can choose not to display a warning before deleting log events. The default is to display a warning. You can also choose how long to wait for responses from the integration node (broker) after a deployment.

To change preferences:

Procedure

1. Click **Window > Preferences**.
2. Expand the **IBM Integration** category in the left pane.
3. Make your selections.
4. Click **OK**.

Related tasks:

“Accessing Administration log information” on page 190

Active brokers record information about their operations in the Administration log. Access the events that are written by the broker by using the IBM Integration Explorer and the IBM Integration Toolkit. For some actions, you can also use the IBM Integration API (also known as the CMP).

Related reference:

Administration Log view

The Administration Log view shows administration requests and their results, changes made to objects, and the result of deployment actions on brokers. The Administration Log view can also describe configuration changes that have been automatically applied as a result of moving the broker from one version to another; for example, after applying maintenance.

Changing the location of the work path

The work path directory is the location where a component stores internal data, such as installation logs, component details, and trace output. The shared-classes directory is also located in the work path directory and is used for deployed Java code. If the work path directory does not have enough capacity, redirect the directory to another file system that has enough capacity.

About this task

The work path is fixed at installation time so that IBM Integration Bus can always find the information that it needs, and always knows where to store new information.

If you need to change the location (for example, if you do not have enough capacity on the automatically-designated file system), do not change the path to the directory; instead, redirect the old work path directory to a new location.

Changing the location of the work path on Windows systems

About this task

When you change the location of the work path, you mount the new partition at the location of the old work path directory.

To change the location of the work path on Windows:

Procedure

1. Shut down all IBM Integration Bus services and processes.
2. Create a new partition on the system. The new partition can be on the same drive as the old work path, or on a different drive.
3. In the IBM Integration Console, locate the work path directory for your installation on the local system by running the following command:
`echo %MQSI_WORKPATH%`
4. Copy the contents of the work path directory to the new partition.
5. Delete the contents of the old work path directory.
6. Open the Computer Management dialog: click **Start** > **Settings** > **Control Panel** > **Administrative Tools** > **Computer Management**; the Computer Management dialog opens.
7. In the left pane of the Computer Management dialog, click **Disk Management**. The new partition that you added, and any existing partitions, are listed in the right pane.
8. Right-click the new partition, then click **Change Drive Letter and Paths**. The Change Drive Letter and Paths dialog opens.
9. Click **Add**. The Add Drive Letter or Path dialog opens.
10. Ensure that **Mount in the following empty NTFS folder** is selected, then browse to the old work path location.
11. Click **OK**, then click **OK** again.

Results

Any files that IBM Integration Bus creates in the work path location are stored on the new partition.

Changing the location of the work path on Linux and UNIX systems

About this task

When you change the location of the work path, you can either mount the new partition at the location of the old work path directory, or you can replace the old work path directory with a soft link that points to the new work path directory.

To change the location of the work path on UNIX and Linux:

Procedure

1. Shut down all IBM Integration Bus services and processes.
2. Create a new directory on a suitable file system.
3. Locate the work path directory for your installation on the local system by running the following command:

```
echo $MQSI_WORKPATH
```
4. Copy the contents of the work path directory to the new partition.
5. Delete the contents of the old work path directory.
6. Perform one of the following tasks so that the IBM Integration Bus installation uses the new work path location:
 - Use the **mount** command to mount the new work path directory at the location of the old work path directory.
 - Delete the old work path directory and replace it with a soft link. Give the soft link the same name as the old work path directory and point the link to the new work path directory.

Results

Any files that IBM Integration Bus creates in the work path location are stored in the new location.

Related tasks:

“Can you see all of your files and folders?” on page 654

How to show all files in Windows Explorer:

Related reference:



mqsi createbroker command

Use the **mqsi createbroker** command to create a broker and its associated resources.



mqsi changebroker command

Use the **mqsi changebroker** command to change one or more of the configuration parameters of the broker.

Backing up resources

Back up your broker components, and the working files associated with brokers, the IBM Integration Explorer, and the IBM Integration Toolkit, so that you can restore these resources if required.

About this task

Back up your broker components regularly to ensure that you can return to a known operational state if necessary. In addition to configuration and operation state, the broker maintains additional resources in its work path, and you can request that these resources are also backed up.

Back up the workspaces and connection files that you have created in the IBM Integration Explorer.

Back up the workspaces you have created in the IBM Integration Toolkit; these resources contain your application development resources; for example message

flows and message model schema files. If you use a development repository to store application resources, such as Rational® ClearCase®, see the documentation associated with that repository to check how you can back up this data.

The following topics tell you how to back up and restore brokers and the IBM Integration Toolkit workspace:

- “Backing up the broker”
- “Restoring the broker” on page 198
- “Backing up the IBM Integration Explorer and IBM Integration Toolkit workspace” on page 200

On distributed systems, you can use the backed up components to restore the broker only on an identical operating environment. The operating system must be at the same level, and the broker and queue manager names must be identical.

Related concepts:



Development repository

Use a development repository to benefit from features such as version control and access control of files, which make it easier for teams to work on shared resources.

Related tasks:

“Recovering after failure” on page 887

Follow a set of procedures to recover after a serious problem.

Related reference:



`mqsicreatebroker` command

Use the `mqsicreatebroker` command to create a broker and its associated resources.

Backing up the broker

Back up the broker configuration and all associated resources.

Before you begin

Before you start: Create the broker.

About this task

You can back up the broker and its resources to preserve the current state of the broker configuration. You can use the backup file that is created to restore a broker in an identical operating environment: the operating system must be at the same level, and the broker and queue manager names must be identical.

You can run this command for a broker that is active. However, you must not take a backup while the broker is processing configuration changes and deployments; the backup file created might contain incomplete information. If the file contains partial records, you cannot use it to restore the broker at a later time.

To ensure that the backup is complete and correct, take a backup either when the broker is not processing a configuration change (such as a deployment or change property) or when the broker is stopped.

Procedure

1. If you want to back up an active broker, check that no configuration change requests are in progress. For example, if you are changing broker properties, or have initiated a deployment, wait for these actions to complete before you back up the broker. Active message flows are unaffected by the backup process.
If you prefer, you can stop the broker before you take a backup by using the **mqsisstop** command.
2. Back up the broker. Specify the broker name and the location in which the backup file is created. You can also optionally specify the name of the backup file, and the name of a file to which a detailed trace is written.
 - **Linux** **UNIX** **Windows** Run the **mqsibackupbroker** command, specifying the broker name and the directory to which the backup file is written.
For example, to back up a broker on Windows, enter the following command:

```
mqsibackupbroker IB9NODE -d c:\MQSI\BACKUP
```
 - **z/OS** Customize and submit the JCL member BIPBUBK.
3. When the command has completed successfully, you can continue to use the broker. If you stopped the broker, restart it by using the **mqsisstart** command.

Results

The current broker configuration is saved in the backup file. Keep the file safe so that you can restore the broker at a later date if required.

Related tasks:

“Restoring the broker”

Restore a broker configuration that you backed up previously.



Customizing the broker JCL

This subtask is part of the larger task of creating a broker on z/OS.

Related reference:



mqsibackupbroker command

Use the **mqsibackupbroker** command to back up the current configuration of a broker.



mqsirestorebroker command

Use the **mqsirestorebroker** command to restore the broker configuration from a backup file.



mqsisstart command

Use the **mqsisstart** command to start the specified broker if all initial verification tests complete successfully.



mqsisstop command

Use the **mqsisstop** command to stop the specified component.

Restoring the broker

Restore a broker configuration that you backed up previously.

Before you begin

Before you start: Back up the broker.

About this task

You can restore a broker on a computer that has an identical configuration by using the backup file that you created. The operating system must be at the same level, and the broker and queue manager names must be identical.

Procedure

1. If you have deleted the broker and it no longer exists, or if you are restoring it on a different computer, create it by using the **mqsicreatebroker** command. Use the same name and parameters that you used for the broker that you backed up, including the name of the queue manager.
2. If the broker is running, stop it by using the **mqsistop** command. If you intend to restore common configuration information from other brokers that you have configured on this computer, you must also stop all the brokers that share this common information. For example, you can restore profile information from common files.
3. Restore the broker. Specify the broker name and the name and location of the backup file. If you want to restore common configuration information, or if you want a trace of the actions taken, specify the appropriate parameters for your platform.
 - **Linux** **UNIX** **Windows** Run the **mqsirestorebroker** command.
For example, to restore a broker on Windows, enter the following command:
mqsirestorebroker WBRK_BROKER -d c:\MQSI\BACKUP -a mybroker.zip
 - **z/OS** Customize and submit the JCL member BIPRSBK.
4. When the command has completed successfully, start the broker by using the **mqsistart** command.

What to do next

Next: The broker configuration has been restored; you can continue your work with this broker.

Related tasks:

“Backing up the broker” on page 197

Back up the broker configuration and all associated resources.



Customizing the broker JCL

This subtask is part of the larger task of creating a broker on z/OS.

Related reference:



mqsibackupbroker command

Use the **mqsibackupbroker** command to back up the current configuration of a broker.



mqsirestorebroker command

Use the **mqsirestorebroker** command to restore the broker configuration from a backup file.



mqsistart command

Use the **mqsistart** command to start the specified broker if all initial verification tests complete successfully.



mqsistop command

Use the **mqsistop** command to stop the specified component.

Backing up the IBM Integration Explorer and IBM Integration Toolkit workspace

The IBM Integration Explorer and IBM Integration Toolkit workspaces contain your personal settings and data, such as message flow and message set resources. You can have multiple workspaces in different locations, and you can also have references to projects that are in other locations, therefore consider all these locations when you back up your resources.

About this task

The default workspace directory for the IBM Integration Explorer depends on the platform on which it is running:

- **Windows** On Windows, the default workspace directory is created at `C:\Users\user_ID\IBM\MQ Explorer\<server project>\`.
- **Linux** On Linux, the default workspace directory is created at `/home/user_ID/IBM/MQ Explorer/\<server project>/`.

where *user_ID* is the user name with which you are logged on. Back up files in these locations, and in all other locations in which you have saved workspace files.

The default workspace directory for the IBM Integration Toolkit depends on the platform on which it is running:

- **Windows** On Windows: `C:\Users\user_ID\IBM\IntegrationToolkit90\workspace`.
- **Linux** On Linux: `/home/user_ID/IBM/IntegrationToolkit90/workspace`.

where *user_ID* is the user name with which you are logged on. Back up files in these locations, and in all other locations in which you have saved workspace files.

The IBM Integration Toolkit workspace directory contains a directory called `.metadata`, which contains your personal settings and preferences for the IBM Integration Toolkit. If the `.metadata` directory gets corrupted, you lose these settings, and the IBM Integration Toolkit reverts to the default layout and preferences. If you have not backed up the `.metadata` directory, you must manually set all preferences again, and import all projects, such as message flow projects, that were displayed in the Application Development view. To back up the `.metadata` directory, take a copy of the directory.

The IBM Integration Toolkit workspace also contains a directory for each project (for example, a message flow project) that you have created in the IBM Integration Toolkit. These directories contain your data, which you must back up.

Use one of the following methods to back up the data in your workspace; the instructions are the same for both the IBM Integration Explorer and the IBM Integration Toolkit.

Procedure

- Export your working projects. You can export the projects directly as a compressed file. For further information, see *Exporting in the Eclipse Workbench User Guide*.
- Copy the project directories from the workspace directory to another location.
- IBM Integration Explorer only: Export all `.broker` connection files to save the details of connections to all your brokers.

- IBM Integration Toolkit only: Take copies of all your BAR files to back up their contents. Include all the associated source files when you build your broker archive (BAR) files; you can then save all content by saving only the BAR file. To add resources to a BAR file that is ready for deployment, select **Include source files**, which adds the message flow and message set source files, and the compiled files.

What to do next

If you want to restore the resources, copy the directories back into your workspace directory and import the projects. For instructions, see Importing in the Eclipse Workbench User Guide.

Related concepts:



Resources

The projects, folders, and files that you work with in the IBM Integration Toolkit workspace are called *resources*. By default, these resources are stored with their metadata in the workspace directory in your local file system. The workspace directory is created the first time that you start the IBM Integration Toolkit.

Related tasks:

“Backing up resources” on page 196

Back up your broker components, and the working files associated with brokers, the IBM Integration Explorer, and the IBM Integration Toolkit, so that you can restore these resources if required.

Related information:



Workbench User Guide - Customizing the Workbench

Chapter 2. Security

Security is an important consideration for both developers of IBM Integration Bus applications, and for system administrators configuring IBM Integration Bus authorities.

When you are designing an IBM Integration Bus application, it is important to consider the security measures that are needed to protect the information in the system.

The “Security overview” introduces the concepts that you need to understand before you set up security for IBM Integration Bus.

The following topics explain the different aspects of security that you need to consider when you are designing your applications:

- “Administration security” on page 213
- “Message flow security” on page 242
- “Integration Bus server security” on page 396

Related concepts:



What's new in Version 9.0?

Learn about the main new functions in IBM Integration Bus Version 9.0.



WS-Security

Web Services Security (WS-Security) describes enhancements to SOAP messaging to provide quality of protection through message integrity, message confidentiality, and single message authentication. WS-Security mechanisms can be used to accommodate a wide variety of security models and encryption technologies.

Related reference:



Security requirements for administrative tasks

You can configure access permissions to govern which users and groups can manipulate objects in the broker network. Security requirements for administrative tasks depend on the platform that you use.

Security overview

When you are designing an IBM Integration Bus application it is important to consider the security measures that are needed to protect the information in the system.

An important aspect of securing an enterprise system is the ability to protect the information that is passed from third parties. This capability includes restricting access to WebSphere MQ and JMS queues. For information about the steps involved, refer to the documentation supplied by your transport provider. If you are using HTTPS, you need to set specific properties in the HTTP nodes. For information about this option, see HTTPInput node, HTTPRequest node, and HTTPReply node.

In addition to securing the transport, you can secure individual messages based on their identity. For more information about securing messages in a message flow, see “Message flow security” on page 242.

Some security configuration is required to enable IBM Integration Bus to work correctly and to protect the information in the system. For example, you can secure the messaging transport with SSL connections, restrict access to queues, apply WS-Security to Web services, and secure access to message flows.

In addition, system administrators need IBM Integration Bus authorities that allow them to perform customization and configuration tasks, run utilities, perform problem determination, and collect diagnostic materials.

System administrators can control users' access to broker resources through the web user interface and REST API. Users can be assigned to roles, which have security permissions assigned to them. For more information about roles and web security, see "Role-based security" on page 220.

The following topics introduce the concepts that you need to understand before you set up security for IBM Integration Bus:

- Planning for security
- "Administration security overview" on page 214
- "Message flow security overview" on page 243
- "Role-based security" on page 220
- WS-Security
- "Authorization for configuration tasks" on page 205
- "Security exits" on page 205
- "Public key cryptography" on page 206
- "Digital certificates" on page 208
- "Digital signatures" on page 212

Additionally, if WebSphere MQ Version 7.1, or later, has been selected for the queue manager and channel auth security is required to be enabled, see "Activating broker administration security for WebSphere MQ Version 7.1, or later" on page 241.

Related concepts:

"Message flow security overview" on page 243

IBM Integration Bus provides a security manager, which enables you to control access to individual messages in a message flow, using the identity of the message.



WS-Security

Web Services Security (WS-Security) describes enhancements to SOAP messaging to provide quality of protection through message integrity, message confidentiality, and single message authentication. WS-Security mechanisms can be used to accommodate a wide variety of security models and encryption technologies.



What's new in Version 9.0?

Learn about the main new functions in IBM Integration Bus Version 9.0.

Related tasks:

"Integration Bus server security" on page 396

You must consider several security aspects when you are setting up brokers running on Windows, Linux, z/OS, or UNIX platforms.

"Setting up administration security" on page 221

Control the actions that users can request against a broker and its resources.

Related reference:



Security requirements for administrative tasks

You can configure access permissions to govern which users and groups can manipulate objects in the broker network. Security requirements for administrative tasks depend on the platform that you use.

Authorization for configuration tasks

Authorization is the process of granting or denying access to a system resource.

For IBM Integration Bus, authorization is concerned with controlling who has permission to access IBM Integration Bus resources, and ensuring that users who attempt to work with those resources have the necessary authorization to do so.

Examples of tasks that require authorization are:

- Configuring a broker using, for example, the **mqsicreatebroker** command.
- Accessing queues, for example, putting a message to the input queue of a message flow.
- Taking actions within the IBM Integration Toolkit, for example, deploying a message flow to an integration server.

Related tasks:

"Integration Bus server security" on page 396

You must consider several security aspects when you are setting up brokers running on Windows, Linux, z/OS, or UNIX platforms.

Related reference:



Security requirements for administrative tasks

You can configure access permissions to govern which users and groups can manipulate objects in the broker network. Security requirements for administrative tasks depend on the platform that you use.



mqsicreatebroker command

Use the **mqsicreatebroker** command to create a broker and its associated resources.

Security exits

Use security exit programs to verify that the partner at the other end of a connection is genuine.

When you connect from the IBM Integration Toolkit or IBM Integration Explorer to a broker on another computer, a security exit is not started by default to monitor the connection. If you want to protect access to the broker from client programs, you can use the WebSphere MQ security exit facility.

You can enable a security exit at each end of the connection between your client session and the broker:

- Set up a security exit on the channel at the broker end. This security exit has no special requirements; you can provide a standard security exit.
- Set up a security exit in the IBM Integration Toolkit or IBM Integration Explorer. Identify the security exit properties when you connect to the broker. The security exit is a standard WebSphere MQ security exit, written in Java.

For an overview of security exits and details of their implementation, see "Channel security exit programs" in the *Intercommunication* section of the WebSphere MQ Version 7 product documentation online.

Related concepts:

“Security overview” on page 203

When you are designing an IBM Integration Bus application it is important to consider the security measures that are needed to protect the information in the system.

Related tasks:

“Using security exits” on page 399

Define a security exit on the WebSphere MQ channel when you create an integration node (broker) connection.

“Integration Bus server security” on page 396

You must consider several security aspects when you are setting up brokers running on Windows, Linux, z/OS, or UNIX platforms.

Related reference:

Security requirements for administrative tasks

You can configure access permissions to govern which users and groups can manipulate objects in the broker network. Security requirements for administrative tasks depend on the platform that you use.

Related information:

WebSphere MQ Version 7 product documentation

Public key cryptography

All encryption systems rely on the concept of a key. A key is the basis for a transformation, usually mathematical, of an ordinary message into an unreadable message. For centuries, most encryption systems have relied on private key encryption. Public key encryption is the only challenge to private key encryption that has appeared in the last 30 years.

Private key encryption

Private key encryption systems use a single key that is shared between the sender and the receiver. Both must have the key; the sender encrypts the message by using the key, and the receiver decrypts the message with the same key. Both the sender and receiver must keep the key private to keep their communication private. This kind of encryption has characteristics that make it unsuitable for widespread general use:

- Private key encryption requires a key for every pair of individuals who need to communicate privately. The necessary number of keys rises dramatically as the number of participants increases.
- Keys must be shared between pairs of communicators, therefore the keys must be distributed to the participants. The need to transmit secret keys makes them vulnerable to theft.
- Participants can communicate only by prior arrangement. You cannot send a usable encrypted message to someone spontaneously. You and the other participant must make arrangements to communicate by sharing keys.

Private key encryption is also called symmetric encryption because the same key is used to encrypt and decrypt the message.

Public key encryption

Public key encryption uses a pair of mathematically-related keys. A message that is encrypted with the first key must be decrypted with the second key, and a message that is encrypted with the second key must be decrypted with the first key.

Each participant in a public key system has a pair of keys. One key is nominated as the private key and is kept secret. The other key is distributed to anyone who wants it; this key is the public key.

Anyone can encrypt a message by using your public key, but only you can read it. When you receive the message, you decrypt it by using your private key.

Similarly, you can encrypt a message for anyone else by using their public key, and they decrypt it by using their private key. You can then send the message safely over an unsecured connection.

This kind of encryption has characteristics that make it very suitable for general use:

- Public key encryption requires only two keys per participant.
- The need for secrecy is more easily met: only the private key needs to be kept secret, and because it does not need to be shared, it is less vulnerable to theft in transmission than the shared key in a symmetric key system.
- Public keys can be published, which eliminates the need for prior sharing of a secret key before communication. Anyone who knows your public key can use it to send you a message that only you can read.

Public key encryption is also called asymmetric encryption, because the same key cannot be used to encrypt and decrypt the message. Instead, one key of a pair is used to undo the work of the other.

With symmetric key encryption, beware of stolen or intercepted keys. In public key encryption, where anyone can create a key pair and publish the public key, the challenge is in verifying the identity of the owner of the public key. Nothing prevents a user from creating a key pair and publishing the public key under a false name. The listed owner of the public key cannot read messages that are encrypted with that key because the owner does not have the corresponding private key. If the creator of the false public key can intercept these messages, that person can decrypt and read messages that are intended for someone else. To counteract the potential for forged keys, public key systems provide mechanisms for validating public keys and other information with digital certificates and digital signatures.

Public Key Infrastructure (PKI)

PKI is an infrastructure that uses public key technology to allow applications to interact securely. PKI uses public key encryption to provide privacy. In practice, only a small amount of data is encrypted in this way. Typically, a *session key* is used with a symmetric algorithm to transmit the bulk of the data efficiently.

In business transactions, trust is even more important than privacy. PKI uses the private key to allow an application to sign a document. For the recipient to authenticate the sender, it needs a reliable way to obtain the public key for the sender. This public key is provided in the form of a digital certificate, which is

mediated by a trusted third party certificate authority (CA).

Related concepts:

“Digital certificates”

Certificates provide a way of authenticating users. Instead of requiring each participant in an application to authenticate every user, third-party authentication relies on the use of digital certificates.

“Digital signatures” on page 212

A digital signature is a number that is attached to a document. For example, in an authentication system that uses public-key encryption, digital signatures are used to sign certificates.



Policy sets

Policy sets and bindings define and configure your WS-Security and WS-RM requirements, supported by IBM Integration Bus, for the SOAPInput, SOAPReply, SOAPRequest, SOAPAsyncRequest, and SOAPAsyncResponse nodes.



WS-Security

Web Services Security (WS-Security) describes enhancements to SOAP messaging to provide quality of protection through message integrity, message confidentiality, and single message authentication. WS-Security mechanisms can be used to accommodate a wide variety of security models and encryption technologies.



WS-Security mechanisms

The WS-Security specification provides three mechanisms for securing Web services at the message level: authentication, integrity, and confidentiality.

Digital certificates

Certificates provide a way of authenticating users. Instead of requiring each participant in an application to authenticate every user, third-party authentication relies on the use of digital certificates.

A digital certificate is equivalent to an electronic ID card. The certificate serves two purposes:

- Establishes the identity of the owner of the certificate
- Distributes the owner's public key

Certificates are issued by trusted third parties, called certificate authorities (CAs). These authorities can be commercial ventures or local entities, depending on the requirements of your application. The CA is trusted to adequately authenticate users before issuing certificates. A CA issues certificates with digital signatures. When a user presents a certificate, the recipient of the certificate validates it by using the digital signature. If the digital signature validates the certificate, the certificate is recognized as intact and authentic. Participants in an application need to validate certificates only; they do not need to authenticate users. The fact that a user can present a valid certificate proves that the CA has authenticated the user. The designation, "trusted third parties", indicates that the system relies on the trustworthiness of the CAs.

The certificates and private keys are stored in files called *keystores* and *truststores*.

- A keystore holds the private keys and public key certificates for an application.
- A truststore contains the CA certificates required to authenticate certificates that are presented by another application.

Contents of a digital certificate

A certificate contains several pieces of information, including information about the owner of the certificate and the issuing CA. Specifically, a certificate includes:

- The distinguished name (DN) of the owner. A DN is a unique identifier, a fully qualified name including not only the common name (CN) of the owner but also the owner's organization and other distinguishing information.
- The public key of the owner.
- The date on which the certificate is issued.
- The date on which the certificate expires.
- The distinguished name of the issuing CA.
- The digital signature of the issuing CA. The message-digest function creates a signature based upon all the previously listed fields.

The idea of a certificate is that a CA takes the public key of the owner, signs the public key with its own private key, and returns the information to the owner as a certificate. When the owner distributes the certificate to another party, it signs the certificate with its private key. The receiver can extract the certificate that contains the CA signature with the public key of the owner. By using the CA public key and the CA signature on the extracted certificate, the receiver can validate the CA signature. If valid, the public key that is used to extract the certificate is considered good. The owner signature is validated, and if the validation succeeds, the owner is successfully authenticated to the receiver.

The additional information in a certificate helps an application to determine whether to honor the certificate. With the expiration date, the application can determine if the certificate is still valid. With the name of the issuing CA, the application can check that the CA is considered trustworthy by the site.

An application that needs to authenticate itself must provide its personal certificate, the one containing its public key, and the certificate of the CA that signed its certificate, called a signer certificate. In cases where chains of trust are established, several signer certificates can be involved.

Requesting certificates

To get a certificate, send a certificate request to the CA. The certificate request includes:

- The distinguished name of the owner or the user for whom the certificate is requested
- The public key of the owner
- The digital signature of the owner

The message digest function creates a signature based on all the previously listed fields.

The CA verifies the signature with the public key in the request to ensure that the request is intact and authentic. The CA then authenticates the owner. Exactly what the authentication consists of depends on a prior agreement between the CA and the requesting organization. If the owner in the request is authenticated successfully, the CA issues a certificate for that owner.

Using certificates: Chain of trust and self-signed certificate

To verify the digital signature on a certificate, you must have the public key of the issuing CA. Public keys are distributed in certificates, therefore you must have a certificate for the issuing CA that is signed by the issuer. One CA can certify other CAs, so a chain of CAs can issue certificates for other CAs, all of whose public keys you need. Eventually, you reach a root CA that issues itself a self-signed certificate. To validate a user certificate, you need certificates for all of the intervening participants back to the root CA. You then have the public keys that you need to validate each certificate, including the user certificate.

A self-signed certificate contains the public key of the issuer and is signed with the private key. The digital signature is validated like any other, and if the certificate is valid, the public key it contains is used to check the validity of other certificates issued by the CA. However, anyone can generate a self-signed certificate. In fact, you can probably generate self-signed certificates for testing purposes before installing production certificates. The fact that a self-signed certificate contains a valid public key does not mean that the issuer is a trusted certificate authority. To ensure that self-signed certificates are generated by trusted CAs, such certificates must be distributed by secure means; for example, hand-delivered on floppy disks, downloaded from secure sites, and so on.

Applications that use certificates store these certificates in a keystore file. This file typically contains the necessary personal certificates, its signing certificates, and its private key. The private key is used by the application to create digital signatures. Servers always have personal certificates in their keystore files. A client requires a personal certificate only if the client must authenticate to the server when mutual authentication is enabled.

To allow a client to authenticate a server, a server keystore file contains the private key and the certificate of the server and the certificates of its CA. A client truststore file must contain the signer certificates of the CAs of each server, which the client must authenticate. The following diagram illustrates how a client authenticates a server.

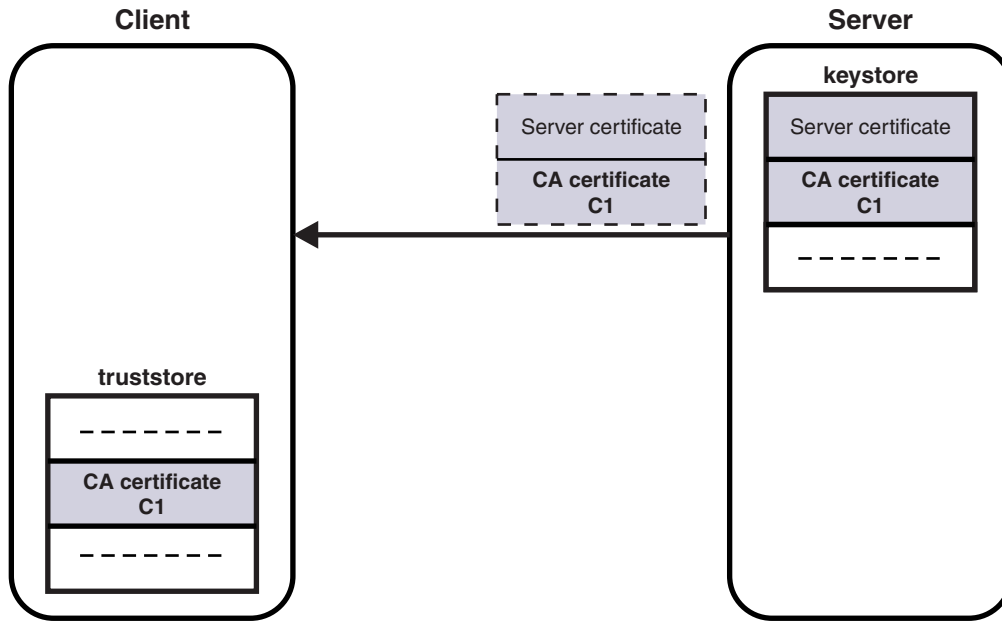


Figure 14. This diagram shows how a client authenticates a server, and is described in the preceding text.

If mutual authentication is needed, the client keystore file must contain the client private key and certificate. The server truststore file requires a copy of the certificate of the client CA. The following diagram illustrates mutual authentication.

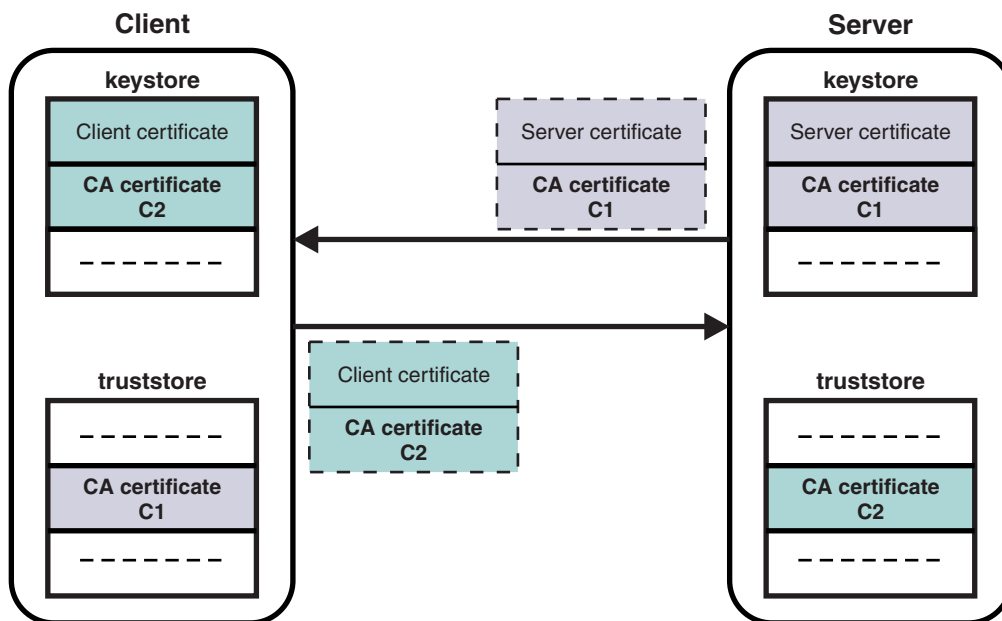


Figure 15. This diagram illustrates mutual authentication, and is described in the preceding text.

Related concepts:

“Public key cryptography” on page 206

All encryption systems rely on the concept of a key. A key is the basis for a transformation, usually mathematical, of an ordinary message into an unreadable message. For centuries, most encryption systems have relied on private key encryption. Public key encryption is the only challenge to private key encryption that has appeared in the last 30 years.

“Digital signatures”

A digital signature is a number that is attached to a document. For example, in an authentication system that uses public-key encryption, digital signatures are used to sign certificates.



Policy sets

Policy sets and bindings define and configure your WS-Security and WS-RM requirements, supported by IBM Integration Bus, for the SOAPInput, SOAPReply, SOAPRequest, SOAPAsyncRequest, and SOAPAsyncResponse nodes.



WS-Security

Web Services Security (WS-Security) describes enhancements to SOAP messaging to provide quality of protection through message integrity, message confidentiality, and single message authentication. WS-Security mechanisms can be used to accommodate a wide variety of security models and encryption technologies.



WS-Security mechanisms

The WS-Security specification provides three mechanisms for securing Web services at the message level: authentication, integrity, and confidentiality.

Digital signatures

A digital signature is a number that is attached to a document. For example, in an authentication system that uses public-key encryption, digital signatures are used to sign certificates.

This signature establishes the following information:

- The integrity of the message: Is the message intact? That is, has the message been modified between the time it was digitally signed and now?
- The identity of the signer of the message: Is the message authentic? That is, was the message signed by the user who claims to have signed it?

A digital signature is created in two steps. The first step distills the document into a large number. This number is the digest code or fingerprint. The digest code is then encrypted, which results in the digital signature. The digital signature is appended to the document from which the digest code is generated.

Several options are available for generating the digest code. This process is not encryption, but a sophisticated checksum. The message cannot regenerate from the resulting digest code. The crucial aspect of distilling the document to a number is that if the message changes, even in a trivial way, a different digest code results. When the recipient gets a message and verifies the digest code by recomputing it, any changes in the document result in a mismatch between the stated and the computed digest codes.

To stop someone from intercepting a message, changing it, recomputing the digest code, and retransmitting the modified message and code, you need a way to verify the digest code as well. To verify the digest code, reverse the use of the public and private keys. For private communication, it makes no sense to encrypt messages with your private key; these keys can be decrypted by anyone with your public

key. However, this technique can be useful for proving that a message came from you. No one can create it because no-one else has your private key. If some meaningful message results from decrypting a document by using someone's public key, the decryption process verifies that the holder of the corresponding private key did encrypt the message.

The second step in creating a digital signature takes advantage of this reverse application of public and private keys. After a digest code is computed for a document, the digest code is encrypted with the sender's private key. The result is the digital signature, which is attached to the end of the message.

When the message is received, the recipient follows these steps to verify the signature:

1. Recomputes the digest code for the message.
2. Decrypts the signature by using the sender's public key. This decryption yields the original digest code for the message.
3. Compares the original and recomputed digest codes. If these codes match, the message is both intact and authentic. If not, something has changed and the message is not to be trusted.

Related concepts:

“Public key cryptography” on page 206

All encryption systems rely on the concept of a key. A key is the basis for a transformation, usually mathematical, of an ordinary message into an unreadable message. For centuries, most encryption systems have relied on private key encryption. Public key encryption is the only challenge to private key encryption that has appeared in the last 30 years.

“Digital certificates” on page 208

Certificates provide a way of authenticating users. Instead of requiring each participant in an application to authenticate every user, third-party authentication relies on the use of digital certificates.



Policy sets

Policy sets and bindings define and configure your WS-Security and WS-RM requirements, supported by IBM Integration Bus, for the SOAPInput, SOAPReply, SOAPRequest, SOAPAsyncRequest, and SOAPAsyncResponse nodes.



WS-Security

Web Services Security (WS-Security) describes enhancements to SOAP messaging to provide quality of protection through message integrity, message confidentiality, and single message authentication. WS-Security mechanisms can be used to accommodate a wide variety of security models and encryption technologies.



WS-Security mechanisms

The WS-Security specification provides three mechanisms for securing Web services at the message level: authentication, integrity, and confidentiality.

Administration security

Administration security controls the rights of users to complete administrative tasks for a broker and its resources.

The following topics introduce the concepts that you need to understand before you can set up administration security, and explain the steps involved in securing integration nodes and their resources:

- “Administration security overview”
- “Setting up administration security” on page 221

Related concepts:

Chapter 2, “Security,” on page 203

Security is an important consideration for both developers of IBM Integration Bus applications, and for system administrators configuring IBM Integration Bus authorities.

“Security overview” on page 203

When you are designing an IBM Integration Bus application it is important to consider the security measures that are needed to protect the information in the system.



WS-Security

Web Services Security (WS-Security) describes enhancements to SOAP messaging to provide quality of protection through message integrity, message confidentiality, and single message authentication. WS-Security mechanisms can be used to accommodate a wide variety of security models and encryption technologies.

“Role-based security” on page 220

You can control access to broker resources by associating web users with roles.

“Message flow security overview” on page 243

IBM Integration Bus provides a security manager, which enables you to control access to individual messages in a message flow, using the identity of the message.

Administration security overview

Broker administration security controls the rights of users to complete administrative tasks for a broker and its resources.

Broker administration security is an optional feature of the broker. When you create a broker, the default setting is that broker administration security is not enabled. You can specify an additional parameter to enable security when you create the broker. You can also change the status of the broker administration security after you have created the broker, and can therefore enable or disable it when appropriate.

When you have enabled broker administration security, set up security control by registering WebSphere MQ permissions for specific user IDs. Permissions are recorded on the following authorization queues that are defined on the broker queue manager:

- `SYSTEM.BROKER.AUTH`. This queue represents the broker and its properties. Only one queue exists of this name for each broker. This queue is defined as a local queue.
- One `SYSTEM.BROKER.AUTH.EG` for each integration server that you define on the broker, where *EG* is the name of the integration server. These queues are defined as alias queues.

When you create a broker, the queue `SYSTEM.BROKER.AUTH` is created. Read, write, and execute authorities are automatically granted to the user group `mqbrkrs` on this queue. This queue is created even if you do not enable security at this time.

If you enable security, the broker checks the authorizations that you have set up on this queue when it receives a request that views or changes its properties or resources. If the user ID associated with the request is not authorized, the broker refuses the request.

When you create an integration server on a broker for which you have enabled security, the integration server authorization queue `SYSTEM.BROKER.AUTH.EG` is created, where `EG` is the name of the integration server. Read, write, and execute authorities are automatically granted to the user group `mqbrkr`s on this queue.

If you are migrating from WebSphere Message Broker Version 6.1, and you use Access Control Lists (ACLs) that you define to the Configuration Manager, you cannot migrate your ACLs directly to later versions of IBM Integration Bus. Review the guidance provided in *Migrating Configuration Manager ACLs* to understand how to set up security by using the ACLs as a basis for security in your later IBM Integration Bus environment.

See the following topics for more information about permissions and queues:

- “IBM Integration Bus permissions and equivalent WebSphere MQ permissions” on page 217
- “Authorization queues for broker administration security” on page 218
- “Role-based security” on page 220

Authorization on z/OS

On z/OS, WebSphere MQ uses the System Authorization Facility (SAF) to route requests for authority checks to an external security manager (ESM) such as the z/OS Security Server Resource Access Control Facility (RACF®). WebSphere MQ does no authority checks of its own. All information about broker administration security on z/OS assumes that you are using RACF as your ESM. If you are using a different ESM, you might need to interpret the information provided for RACF in a way that is relevant to your ESM.

If you are activating security on the WebSphere MQ queue manager on z/OS for the first time, you must set up the profiles or other resources that are required by your ESM to access queues. You must also check that the queue manager is configured to access the security profiles in the correct class; `MQQUEUE` for uppercase queue names and `MXQUEUE` for mixed case queue names.

For further information about queue manager security and security profiles, see the *z/OS System Administration Guide* and *z/OS System Setup Guide* sections of the WebSphere MQ Version 7 product documentation online.

Authority checking

If you have activated broker administration security, all actions performed by users of the following interfaces are subject to authority checking:

- IBM Integration Toolkit sessions
- IBM Integration Explorer sessions
- IBM Integration Bus
- Java programs that use the REST API to perform operations on the broker
- Java programs that use the CMP to perform operations on the broker
- All the following commands:
 - `mqsichangeresourcestats`
 - `mqsicreateexecutiongroup`
 - `mqsideleteexecutiongroup`
 - `mqsideploy`
 - `mqsilist`
 - `mqsimode`
 - `mqsireloadsecurity`
 - `mqsireportresourcestats`

- `mqsistartmsgflow`
- `mqsistopmsgflow`
- `mqswebuseradmin`

For additional authorization required for these commands, see *Commands and authorizations for broker administration security*.

You can run all commands that are not stated here only on the computer on which the broker is running. Either your user ID, or the ID under which the broker is running, must be a member of the security group `mqbrkr` when you run the unlisted commands. Each command topic describes the authority that is required.

Users of the IBM Integration Explorer and IBM Integration Toolkit who do not have read, write, and execute authority for the broker or integration servers, have only restricted access to those resources. An icon is displayed against each resource to indicate that user authority is restricted. The actions that the user can request against a resource are determined by the restricted authority that is in place for that user.

Authority persistence

If a user ID is granted authority to access a broker, the access is retained at least until the current connection or session is ended by one of the following events:

- The user closes the IBM Integration Explorer or IBM Integration Toolkit session
- The CMP application disconnects from the BrokerProxy object

Even if you update or remove the authority for this user ID, the authorization does not change while the connection is active.

However, authorization is always checked for operations against integration servers and message flows; if you change or remove the authorization for a user ID to an integration server, subsequent requests in the current connection might fail.

Additional administration security

In your environment, a check on the user ID making a request might not provide a sufficient level of security. If you require a more secure solution, one or both of the following options are available:

- You can enable SSL on a WebSphere MQ client connection between the source of the request (for example, the IBM Integration Explorer) and the target queue manager on which the broker is running.
- You can configure your WebSphere MQ network so that certain types of users can be directed through a specific server connection (SVRCONN) channel, provided they comply with the CHLAUTH rules. For details of channel security, see the *System Administration Guide* section in the WebSphere MQ Version 7 product documentation online.

Related concepts:

“Security overview” on page 203

When you are designing an IBM Integration Bus application it is important to consider the security measures that are needed to protect the information in the system.

“Role-based security” on page 220

You can control access to broker resources by associating web users with roles.

Related tasks:

Migrating Configuration Manager ACLs

If you are migrating from WebSphere Message Broker Version 6.1, you can use the Access Control Lists (ACLs) that you set up in the Configuration Manager as the basis for your security model in IBM Integration Bus Version 9.0.

“Enabling administration security” on page 222

Enable administration security on a broker to control which users can complete specific tasks against that broker and its resources.

“Authorizing users for administration” on page 228

Grant authority to one or more groups or users to authorize them to complete specific tasks against a broker and its resources.

“Enabling SSL on the WebSphere MQ Java Client” on page 400

The WebSphere MQ Java Client supports SSL-encrypted connections over the server-connection (SVRCONN) channel between an application and the queue manager. Configure SSL support for connections between applications that use the CMP (including the IBM Integration Toolkit and the IBM Integration Explorer) and a broker.

“Setting up administration security” on page 221

Control the actions that users can request against a broker and its resources.

Related reference:

Commands

All IBM Integration Toolkit and runtime commands that are provided on distributed systems are listed, grouped by function, with references to command details.

Related information:

 [WebSphere MQ Version 7 product documentation](#)

IBM Integration Bus permissions and equivalent WebSphere MQ permissions

If you have enabled broker administration security, you can give different permissions to user IDs to allow them to complete various actions against a broker or its resources.

When a user requests an action against a broker or an integration server, the broker accesses the appropriate authorization queues to check that the user ID has the correct authority for that action against the target resource.

Permission to perform a broker administration task is mapped to a WebSphere MQ authority associated with the relevant authorization queue, and is created and maintained by the broker administrator. The mapping from broker permission to WebSphere MQ permission is shown in the following table.

Broker permission	WebSphere MQ permission
Read	Inquire
Write	Put
Execute	Set

For information about the authorizations that are required for specific tasks, see [Tasks and authorizations for administration security](#).

WebSphere MQ specific and generic profiles

WebSphere MQ supports both specific and generic profiles to manage WebSphere MQ permissions. When you enable broker administration security, you can create specific profiles to define WebSphere MQ permissions on SYSTEM.BROKER.AUTH and on one or more SYSTEM.BROKER.AUTH.EG queues (where EG is the name of a specific integration server).

You might want to grant a user, or group of users, authority to a number of integration servers, or perhaps all integration servers. You can use a WebSphere MQ generic profile to grant authority in this way. A generic profile defines authority to an existing set of integration servers, and all additional groups, that match the profile. A generic profile is one that uses special characters (wildcard characters) in the profile name, such as asterisks (*).

For example, if you want to create a generic profile to authorize access to all integration servers defined on the broker, you can specify SYSTEM.BROKER.AUTH.**. If you want a profile for a set of integration servers with names that all start with the same character string, you can specify SYSTEM.BROKER.AUTH.TEST**.

For more information about WebSphere MQ generic profile wildcard characters, see “Using wildcard characters”, and for information about WebSphere MQ generic profile priorities, see “Profile priorities”. You can find both topics in the *System Administration Guide* section of the WebSphere MQ Version 7 product documentation online.

Related concepts:

“Security overview” on page 203

When you are designing an IBM Integration Bus application it is important to consider the security measures that are needed to protect the information in the system.

“Authorization queues for broker administration security”

If you have enabled broker administration security, the broker examines specific queues to determine if a user has the authority to complete a particular task against a broker or its resources.

Related tasks:

“Enabling administration security” on page 222

Enable administration security on a broker to control which users can complete specific tasks against that broker and its resources.

“Authorizing users for administration” on page 228

Grant authority to one or more groups or users to authorize them to complete specific tasks against a broker and its resources.

Related reference:



Tasks and authorizations for administration security

If you have enabled broker administration security, users require specific authority so that they can complete administration tasks.

Related information:



WebSphere MQ Version 7 product documentation

Authorization queues for broker administration security

If you have enabled broker administration security, the broker examines specific queues to determine if a user has the authority to complete a particular task against a broker or its resources.

When you create a broker, the queue `SYSTEM.BROKER.AUTH` is created. Read, write, and execute authorities are automatically granted to the user group `mqrbrks` on this queue. This queue is created even if you do not enable security at this time.

The `SYSTEM.BROKER.AUTH` queue is created as a local queue, and is used to define which users are authorized to perform actions on the broker and the broker properties.

When you create an integration server on a broker for which you have enabled security, the integration server authorization queue `SYSTEM.BROKER.AUTH.EG` is created, where *EG* is the name of the integration server. Read, write, and execute authorities are automatically granted to the user group `mqrbrks` on this queue. The dedicated integration server queues are created as aliases to the queue `SYSTEM.BROKER.AUTH`.

If you create a broker without administration security, you can change it later. If you have defined one or more integration servers on that broker when you change its security setting, the required integration server authorization queues are defined.

A queue can be created only by a user ID that is a member of the WebSphere MQ security group `mqm`. Therefore the user ID who creates a broker, changes a broker, and the ID under which the broker is running when an integration server is created, must be a member of that security group. If the user ID does not have this authority, a message is returned to the command (for the `mqsichangebroker` command only), or written to the system log, with the error and the name of the queue. You must create the queue yourself, or ask your WebSphere MQ administrator to create it for you.

WebSphere MQ restricts the length of a queue name to 48 characters. Queue name characters must be in the `En_US` ASCII character set, and contain only uppercase and lowercase letters, digits, and the following special characters; period (`.`), forward slash (`/`), underscore (`_`), and percent (`%`). If the name of your integration server includes a character that is not valid, that character is replaced in the WebSphere MQ queue name by an underscore character. For example, if you create an integration server with the name `test@environment`, the authorization queue is created with the name `SYSTEM.BROKER.AUTH.test_environment`.

If you are running a secure environment, limit the names of your integration servers to 29 characters. This limit ensures that the authorization queue names generated, which include the prefix `SYSTEM.BROKER.AUTH`, do not exceed the WebSphere MQ limit of 48 characters.

If your integration server names do not all conform to the length and character requirements, integration servers with similar names might result in a shared authorization queue. If this situation occurs, a warning message is returned to the user that issued the command, or is written to the system log, when the second integration server is created to state that the queue is shared.

When you delete an integration server, its associated authorization queue is retained. The queue is deleted if you specify the appropriate parameter when you delete the broker. The queue can be reused if you re-create the integration server, but you must check the authorities that you have defined on the queue to ensure that they are still valid.

If you rename an integration server, you must first create an authorization queue with the appropriate name. You must also re-create the WebSphere MQ permissions associated with the original authorization queue on this queue before you rename the integration server; the broker does not perform this task on your behalf. The broker rejects the rename request if the authorization queue does not exist, to ensure that security is not affected by the renaming. If you do not re-create these permissions, no user IDs are authorized to perform a task against the renamed integration server.

When you delete a broker, you can specify that all its authorization queues are also deleted; they are not deleted by default. If you specify that the queue manager is deleted at this time, all queues are deleted.

Related concepts:

“Security overview” on page 203

When you are designing an IBM Integration Bus application it is important to consider the security measures that are needed to protect the information in the system.

“IBM Integration Bus permissions and equivalent WebSphere MQ permissions” on page 217

If you have enabled broker administration security, you can give different permissions to user IDs to allow them to complete various actions against a broker or its resources.

Related tasks:

“Enabling administration security” on page 222

Enable administration security on a broker to control which users can complete specific tasks against that broker and its resources.

“Authorizing users for administration” on page 228

Grant authority to one or more groups or users to authorize them to complete specific tasks against a broker and its resources.

Related information:

 [WebSphere MQ Version 7 product documentation](#)

Role-based security

You can control access to broker resources by associating web users with roles.

A *role* is a system user account that has a set of security permissions assigned to it, and each web user account is associated with a particular role. The permissions are checked to determine a web user's authorization to perform tasks in the web user interface or the REST application programming interface (API).

As a broker administrator, you can control the access that web users have to broker resources, by assigning each user to a predefined role. You can authorize users with a particular role to complete specific actions, by enabling or disabling aspects of the web or REST interface, or by configuring the web user interface to display only the options for which users are authorized. For example, you might allow users with one role to view broker resources, while allowing users with another role to modify them.

You can grant the same authorizations to multiple users by assigning them to the same role, but each user can be assigned to only one role.

For more information about assigning web users to roles, see “Managing web user accounts” on page 226, “Controlling access to data and resources in the web user interface” on page 239, and the `mqswebuseradmin` command.

Related concepts:



The IBM Integration API

The IBM Integration API is a programming interface that your applications can use to control brokers and their resources through a remote interface.

Related tasks:

“Accessing the web user interface” on page 133

You can access broker resources by logging on to the web user interface.

“Recording, viewing, and replaying data” on page 595

Record data that is processed by a message flow, or emitted from WebSphere Application Server. View and replay the data.

“Enabling security for record and replay” on page 598

You can restrict the users who can view and replay data for a broker by enabling security.

“Administering brokers using the web user interface” on page 127

You can use the IBM Integration Bus web user interface to administer broker resources.

Chapter 1, “Administering brokers and broker resources,” on page 1

Administering brokers and associated integration node resources includes the tasks that you perform frequently to activate and manage those resources. Choose the method you prefer to administer your brokers and associated resources.

Setting up administration security

Control the actions that users can request against a broker and its resources.

About this task

You can activate administrative control for an individual broker when you create it. If appropriate, you can enable the security after creation by changing broker properties. When you have activated broker administration security, grant users or groups authority to complete their expected tasks.

You can also grant permissions to users of the web user interface by associating the web user account with a predefined role. For more information, see “Role-based security” on page 220 and “Managing web user accounts” on page 226.

- “Enabling administration security” on page 222
- “Authenticating users for administration” on page 225
- “Authorizing users for administration” on page 228
- “Disabling administration security” on page 240

On z/OS, WebSphere MQ uses the System Authorization Facility (SAF) to route requests for authority checks to an external security manager (ESM) such as the z/OS Security Server Resource Access Control Facility (RACF). WebSphere MQ does no authority checks of its own. All information about broker administration security on z/OS assumes that you are using RACF as your ESM. If you are using a different ESM, you might need to interpret the information provided for RACF in a way that is relevant to your ESM.

Related concepts:

“Security overview” on page 203

When you are designing an IBM Integration Bus application it is important to consider the security measures that are needed to protect the information in the system.

“Administration security overview” on page 214

Broker administration security controls the rights of users to complete administrative tasks for a broker and its resources.

“Role-based security” on page 220

You can control access to broker resources by associating web users with roles.

Related tasks:

“Managing web user accounts” on page 226

You can control a web user's access to broker resources by associating the web user ID with a role, which has security permissions assigned to it.

Related reference:



Tasks and authorizations for administration security

If you have enabled broker administration security, users require specific authority so that they can complete administration tasks.



Commands and authorizations for broker administration security

If you have enabled broker administration security, users require specific authority to be able to run the administration commands.

Enabling administration security

Enable administration security on a broker to control which users can complete specific tasks against that broker and its resources.

About this task

You can optionally enable administration security for a broker when you create it. If you decide to enable administrative security after you have created the broker, you can change the appropriate broker property.

When you create or change a broker, your user ID must be a member of the WebSphere MQ control group mqm.

Procedure

1. To enable broker administration security when you create the broker, select the security option in the “Create broker” wizard in IBM Integration Explorer, or specify the parameter **-s active** on the **mqsicreatebroker** command. For example, to create a broker called IB9NODE with security enabled on AIX®, enter the following command:

```
mqsicreatebroker IB9NODE -q IB9QMGR -s active
```

The broker creates the authorization queue SYSTEM.BROKER.AUTH. This queue is used to define which users are authorized to perform an action on the broker.

The broker also assigns default permissions of inquire, put, and set authority to this queue. These permissions grant read, write, and execute authority on the broker to all members of the mqbrkr group. Therefore, you must ensure that at least one member of your broker administration team is a member of this group. You must also manage the membership of this group with care, and ensure that this level of authorization is granted only to users who require it.

On z/OS, these permissions are implemented as levels of authority in the external security manager (ESM) that you are using with WebSphere MQ. If you are using RACF as your ESM, the levels are hierarchical: for example, ALTER access implies READ and WRITE access. You must therefore check the

documentation for your ESM to understand the authorization levels that it supports. On distributed platforms, no equivalent hierarchy exists, and the three permissions are independent.

2. To enable broker administration security on an existing broker:
 - a. Stop the broker in the IBM Integration Explorer, or run the **mqsistop** command.
 - b. Select the security option for this broker in the IBM Integration Explorer, or run the **mqsichangebroker** command, specifying the parameter **-s active**. For example, to enable security for the broker IB9NODE, enter the following command:

```
mqsichangebroker IB9NODE -s active
```

The broker creates a queue for each defined integration server, with a name that conforms to the format SYSTEM.BROKER.AUTH.EG, where EG is the name of the integration server. It assigns default permissions of inquire, put, and set authority to the queue, which grants read, write, and execute access to the integration server and its properties, for the mqbrkrs group. These queues, and the broker authorization queue SYSTEM.BROKER.AUTH, are now ready for use.

The names of queues that are generated for your integration servers might not match exactly the name of the integration servers, because WebSphere MQ enforces some restrictions on the authorization queue names. For details of these restrictions, and the possible effects, see “Authorization queues for broker administration security” on page 218.

- c. Start the broker in the IBM Integration Explorer, or run the **mqsistart** command.
3. Check that the user ID under which your broker is running is a member of the WebSphere MQ security group mqm. Without this authority, the broker cannot create or delete the authorization queues for integration servers at run time. Because mqm authority grants full access control to all WebSphere MQ resources, you might not want your broker running with this level of authority. If you do not want the broker to run with mqm authority, you must work with your WebSphere MQ administrator to ensure that the required queues are created (and deleted) at the appropriate time.

If you want to give your broker mqm authority:

- On Linux and UNIX systems, add to mqm the user ID that started the broker.
 - On Windows, add to mqm the user ID that you specified as the service user ID. When you add this user ID, the same level of authority is granted to all user IDs defined in the same primary group. You must therefore control carefully your group memberships to ensure that access is not granted to user IDs that do not require it.
 - On z/OS, grant equivalent permissions to the started task user ID.
4. Check also that the user ID associated with the broker, defined in the previous step, has WebSphere MQ altuser authority. This authority is required by the broker to request WebSphere MQ to check authorities.

Display registry entries for a broker by using **mqsireportbroker** *brokerName*.

What to do next

Next: Grant authority to users to reflect what tasks you want them to be able to complete, by populating the queues with the appropriate details. This task is

described in “Authorizing users for administration” on page 228.

Related concepts:

“Security overview” on page 203

When you are designing an IBM Integration Bus application it is important to consider the security measures that are needed to protect the information in the system.

“IBM Integration Bus permissions and equivalent WebSphere MQ permissions” on page 217

If you have enabled broker administration security, you can give different permissions to user IDs to allow them to complete various actions against a broker or its resources.

“Role-based security” on page 220

You can control access to broker resources by associating web users with roles.

Related tasks:

“Authorizing users for administration” on page 228

Grant authority to one or more groups or users to authorize them to complete specific tasks against a broker and its resources.

“Disabling administration security” on page 240

Disable administration security to remove control over a broker and its resources.

“Starting and stopping a broker” on page 19

Run the appropriate command to start or stop a broker.

“Managing web user accounts” on page 226

You can control a web user's access to broker resources by associating the web user ID with a role, which has security permissions assigned to it.

Related reference:

 **mqsicreatebroker** command

Use the **mqsicreatebroker** command to create a broker and its associated resources.

 **mqsichangebroker** command

Use the **mqsichangebroker** command to change one or more of the configuration parameters of the broker.

 **mqsireportbroker** command

Use the **mqsireportbroker** command to display broker registry entries.

 Tasks and authorizations for administration security

If you have enabled broker administration security, users require specific authority so that they can complete administration tasks.

 Commands and authorizations for broker administration security

If you have enabled broker administration security, users require specific authority to be able to run the administration commands.

 **mqswebuseradmin** command

Use the **mqswebuseradmin** command to administer user accounts for the web user interface.

Related information:

 [WebSphere MQ Version 7 product documentation](#)

Authenticating users for administration

Authentication is the process of establishing the identity of a user or system and verifying that the identity is valid. You can control access to the IBM Integration Bus administration interfaces by using the authentication capabilities that are provided with the product.

Before you begin

Before you start:

Read the following topics:

- “Administration security overview” on page 214
- “Setting up administration security” on page 221
- “Role-based security” on page 220

About this task

IBM Integration Bus provides authentication support for the following broker administration interfaces:

- IBM Integration Bus web user interface
- IBM Integration Bus RESTful application programming interface (API)

If administration security is enabled, users of the web user interface and the RESTful API must log in with a user ID and password, which are authenticated by checking them against the credentials held in the broker. Users' access to data and broker resources is controlled by the permissions that have been associated with their role (system user ID).

If administration security is not enabled, web users can interact with the web user interface without logging on; they interact with the web UI as the 'default' user and can access all data and broker resources. For users of the RESTful API, all REST requests are unrestricted if administration security is not enabled.

For the following administration interfaces, authentication is provided only by the system login; no additional authentication is carried out by the broker:

- IBM Integration Explorer
- IBM Integration Toolkit
- IBM Integration API
- Command line

When you access the broker through these interfaces, the broker accepts the system user ID and uses it to check against authorization queues.

For more information about authenticating users for broker administration, see “Managing web user accounts” on page 226 and “Accessing the web user interface” on page 133.

Related concepts:

“Security overview” on page 203

When you are designing an IBM Integration Bus application it is important to consider the security measures that are needed to protect the information in the system.

“Administration security” on page 213

Administration security controls the rights of users to complete administrative

tasks for a broker and its resources.

“Administration security overview” on page 214

Broker administration security controls the rights of users to complete administrative tasks for a broker and its resources.

“Role-based security” on page 220

You can control access to broker resources by associating web users with roles.



The IBM Integration API

The IBM Integration API is a programming interface that your applications can use to control brokers and their resources through a remote interface.

Related tasks:

“Enabling administration security” on page 222

Enable administration security on a broker to control which users can complete specific tasks against that broker and its resources.

“Disabling administration security” on page 240

Disable administration security to remove control over a broker and its resources.

“Managing web user accounts”

You can control a web user's access to broker resources by associating the web user ID with a role, which has security permissions assigned to it.

“Accessing the web user interface” on page 133

You can access broker resources by logging on to the web user interface.

“Administering brokers using the web user interface” on page 127

You can use the IBM Integration Bus web user interface to administer broker resources.

“Configuring the web user interface server” on page 128

To enable access to broker resources through the web user interface, configure the IBM Integration Bus web user interface server.

Chapter 1, “Administering brokers and broker resources,” on page 1

Administering brokers and associated integration node resources includes the tasks that you perform frequently to activate and manage those resources. Choose the method you prefer to administer your brokers and associated resources.

Related reference:



`mqswebuseradmin` command

Use the `mqswebuseradmin` command to administer user accounts for the web user interface.

Managing web user accounts:

You can control a web user's access to broker resources by associating the web user ID with a role, which has security permissions assigned to it.

Before you begin

Before you start:

Read the following topics:

- “Administering brokers using the web user interface” on page 127
- “Accessing the web user interface” on page 133
- “Role-based security” on page 220

About this task

Broker administrators can use the **mqswebuseradmin** command to create a new web user, to set or change a web user's password, to remove a web user, or to assign a web user to a role.

As a broker administrator, you can set up multiple system user IDs on the system that the broker is running on, with different permissions set on the authorization queues (SYSTEM.BROKER.AUTH, SYSTEM.BROKER.AUTH.*integrationServerName*, and SYSTEM.BROKER.DC.AUTH). These permissions then apply to web users through their assigned role. Web users also require the following permissions to use the web user interface:

- GET and PUT authority on the queue
SYSTEM.BROKER.WEBADMIN.SUBSCRIPTION
- SUBSCRIBE and PUBLISH authority on the topic SYSTEM.BROKER.MB.TOPIC

If administration security is enabled, web users can access the web UI only when they have logged on using their web user account. Their access to data and broker resources is controlled by the permissions that have been associated with their role (system user ID). If administration security is not enabled, web users can interact with the web UI without logging on; they interact with the web UI as the 'default' user and can access all data and broker resources.

The steps shown in this task are based on the assumption that you want to have broker administration security enabled.

Complete these steps to grant access to web users based on their assigned role:

Procedure

1. If you are enabling broker administration security:
 - a. Create a system user account (on the operating system) for each role that you have identified. For example, you might decide that your web users can be categorized into two main roles: web administrators and web users. Create a system user account for each of the roles, such as *ibmuser* and *ibmadmin*. These users will typically require different authorizations to perform tasks in the broker administration interface (such as permission to view or modify resources), according to their role.
 - b. Grant permissions on the authorization queues for the system user accounts that you have created for your roles (*ibmuser* and *ibmadmin*). For information about how to do this, see “Authorizing users for administration” on page 228.
 - c. Enable broker administration security for your broker by setting the **-s** parameter to **active**.
 - To enable administration security when you create the broker, run the **mqscreatebroker** command, as shown in the following example:
mqscreatebroker brokerName -q brokerQueueManagerName -s active
 - (If you run this command on Windows, you must also set the **-i** parameter. For details, see **mqscreatebroker** command.)
 - To enable administrative security for a broker that you have already created, stop the broker, then run the **mqschangebroker**, as shown in the following example:
mqschangebroker brokerName -s active

For more information, see “Enabling administration security” on page 222.

2. Use the **mqswebuseradmin** command to create your web user accounts. If you have broker administration security enabled, you can also use the **mqswebuseradmin** command to modify your web user accounts and assign them to the appropriate roles. For more information, see **mqswebuseradmin** command.

Related concepts:

“Role-based security” on page 220

You can control access to broker resources by associating web users with roles.



The IBM Integration API

The IBM Integration API is a programming interface that your applications can use to control brokers and their resources through a remote interface.

“Administration security” on page 213

Administration security controls the rights of users to complete administrative tasks for a broker and its resources.

Related tasks:

“Accessing the web user interface” on page 133

You can access broker resources by logging on to the web user interface.

“Administering brokers using the web user interface” on page 127

You can use the IBM Integration Bus web user interface to administer broker resources.

“Recording, viewing, and replaying data” on page 595

Record data that is processed by a message flow, or emitted from WebSphere Application Server. View and replay the data.

“Enabling security for record and replay” on page 598

You can restrict the users who can view and replay data for a broker by enabling security.

“Configuring the web user interface server” on page 128

To enable access to broker resources through the web user interface, configure the IBM Integration Bus web user interface server.

Chapter 1, “Administering brokers and broker resources,” on page 1

Administering brokers and associated integration node resources includes the tasks that you perform frequently to activate and manage those resources. Choose the method you prefer to administer your brokers and associated resources.

Related reference:



mqswebuseradmin command

Use the **mqswebuseradmin** command to administer user accounts for the web user interface.

Authorizing users for administration

Grant authority to one or more groups or users to authorize them to complete specific tasks against a broker and its resources.

About this task

You must activate broker administration security before you can grant or revoke permissions for specific user IDs. When you first activate security, you must set up initial control for each user to authorize them to perform certain operations. You can then grant additional authority, revoke permissions, as required.

You can grant authorities to individual principals (user IDs), to groups of users, or both, on all platforms:

- If you grant a group or a user ID authority at the broker level (on queue SYSTEM.BROKER.AUTH), it does not inherit authority for integration servers. You must explicitly grant authority to all, or to individual, integration servers.
- On Linux and UNIX, you can authorize both principals and groups. However, when authorizing a principal, Message Broker additionally authorizes the primary group of that principal. If there are many users who belong to that primary group, they become authorized at the same time. Message Broker users should consider using groups instead of primary groups for authorization, because of differences in how variants of UNIX use primary groups.
- If a user ID is a member of the WebSphere MQ security group mqm, it automatically has authority to all WebSphere MQ objects.
- On Windows, if a user ID is a member of the security group Administrators, it automatically has authority to all WebSphere MQ objects.

When you change authorizations on a queue, the broker accesses the updated values the next time that a request is processed. You do not have to stop and restart the broker.

If you update user ID or group membership by using the operating system facilities on the platform on which the broker queue manager is running, you must ensure that the queue manager is aware of these changes. Select the option **Refresh Authorization Service** in the WebSphere MQ Explorer to notify the queue manager of the updated status.

The authority that is required depends on the requirements of the user:

- “Required authority for administrators”
- “Required authority for users who connect to the broker” on page 230
- “Required authority for recording and replaying data” on page 231
- “Required authority for users of the web user interface” on page 230
- “Required authority for developers” on page 231

The way in which you set up the required authorities differs by platform:

- “Granting and revoking authority on Linux, UNIX, and Windows systems” on page 232
- “Granting and revoking authority on z/OS systems” on page 234

These platform-specific topics also give examples of command usage for viewing what authorizations are in place by using the WebSphere MQ **dspmqa** command, and for dumping this information by using the **dmpmqaut** command.

For information about the additional permissions that are required for recording and replaying data, see “Enabling security for record and replay” on page 598.

Required authority for administrators:

About this task

When you activate administration security, the WebSphere MQ permissions for inquire, put, and set are granted for the group mqbrkrs for the queue SYSTEM.BROKER.AUTH. These permissions grant read, write, and execute authority on the broker and its properties to all user IDs that are members of mqbrkrs.

If you want additional user IDs to have administrator authorization, either add those IDs to the group mqbrkrs, or add WebSphere MQ permissions for inquire, put, and set for those user IDs to this queue.

The following table summarizes the WebSphere MQ permissions that are required:

Object	Name	Permissions
Queue manager	The queue manager associated with the broker; for example, IB9QMGR	Connect Inquire
Queue	SYSTEM.BROKER.DEPLOY.QUEUE	Put
Queue	SYSTEM.BROKER.DEPLOY.REPLY	Get Put
Queue	SYSTEM.BROKER.AUTH	Inquire Put Set
Queue	SYSTEM.BROKER.AUTH.EG	Inquire Put Set

For more information about permissions on authorization queues, see “IBM Integration Bus permissions and equivalent WebSphere MQ permissions” on page 217 and Tasks and authorizations for administration security.

Required authority for users who connect to the broker:

About this task

If a user or application wants to connect to a broker, you must grant them the appropriate permissions. All applications written to the CMP, and users of the IBM Integration Explorer and the IBM Integration Toolkit, require permissions based on their expected actions. The following table shows the WebSphere MQ permissions that are required:

Object	Name	Permissions
Queue manager	The queue manager associated with the broker; for example, IB9QMGR	Connect Inquire
Queue	SYSTEM.BROKER.DEPLOY.QUEUE	Put
Queue	SYSTEM.BROKER.DEPLOY.REPLY	Get Put
Queue	SYSTEM.BROKER.AUTH	Inquire ¹
Queue	SYSTEM.BROKER.AUTH.EG	Inquire ¹

Notes:

1. Users and applications can connect to the broker without this level of authority, but are unable to request actions against the broker, including viewing properties.

Additional permissions are also required for users who will connect to the broker through the web user interface, as described in “Required authority for users of the web user interface.”

Required authority for users of the web user interface:

About this task

Before your users can use the web user interface, you must set security permissions for the roles (system user accounts) that are associated with the web user accounts. The permissions are checked to determine a web user's authorization to perform tasks in the web user interface or the REST application programming interface (API). Ensure that users have the following authorizations:

Object	Name	Permissions
Queue	SYSTEM.BROKER.WEBADMIN.SUBSCRIPTION	GET PUT
Topic	SYSTEM.BROKER.MB.TOPIC	SUBSCRIBE PUBLISH

For more information about roles and web user accounts, see “Role-based security” on page 220 and “Managing web user accounts” on page 226.

Required authority for recording and replaying data: About this task

Before you can record and replay data, you must set security permissions for data capture in addition to setting administrative security. The queue SYSTEM.BROKER.DC.AUTH controls the record and replay actions that a user can complete on the broker. Ensure that users have the appropriate authorizations to complete the following actions on this queue:

Action	Authority required
To view data, bit streams, and exception lists	READ (+INQ)
To replay data	EXECUTE (+SET)

Required authority for developers: About this task

If your users are working with existing integration servers, and development resources such as BAR files, add the WebSphere MQ permissions inquire, put, and set for those users to one or more SYSTEM.BROKER.AUTH.EG queues (where EG is the name of the integration server).

When you run a broker with administration security enabled, you might need to restrict the names and the length of the names that you give to your integration servers, because WebSphere MQ enforces some restrictions on the authorization queue names. For details of these restrictions, and the possible effects, see “Authorization queues for broker administration security” on page 218.

Related concepts:

“Security overview” on page 203

When you are designing an IBM Integration Bus application it is important to consider the security measures that are needed to protect the information in the system.

“Administration security overview” on page 214

Broker administration security controls the rights of users to complete administrative tasks for a broker and its resources.

“Authorization queues for broker administration security” on page 218

If you have enabled broker administration security, the broker examines specific queues to determine if a user has the authority to complete a particular task

against a broker or its resources.

“IBM Integration Bus permissions and equivalent WebSphere MQ permissions” on page 217

If you have enabled broker administration security, you can give different permissions to user IDs to allow them to complete various actions against a broker or its resources.

Related tasks:

“Enabling administration security” on page 222

Enable administration security on a broker to control which users can complete specific tasks against that broker and its resources.

“Granting and revoking authority on Linux, UNIX, and Windows systems”

Grant or revoke authority to one or more groups or users to complete specific tasks against a broker running on Linux, UNIX, or Windows.

“Granting and revoking authority on z/OS systems” on page 234

Grant or revoke authority to one or more groups or users to complete specific tasks against a broker running on z/OS.

“Disabling administration security” on page 240

Disable administration security to remove control over a broker and its resources.

“Managing web user accounts” on page 226

You can control a web user's access to broker resources by associating the web user ID with a role, which has security permissions assigned to it.

“Enabling security for record and replay” on page 598

You can restrict the users who can view and replay data for a broker by enabling security.

Related reference:



Tasks and authorizations for administration security

If you have enabled broker administration security, users require specific authority so that they can complete administration tasks.

Related information:



WebSphere MQ Version 7 product documentation

Granting and revoking authority on Linux, UNIX, and Windows systems:

Grant or revoke authority to one or more groups or users to complete specific tasks against a broker running on Linux, UNIX, or Windows.

Before you begin

Before you start:

Activate broker administration security for the broker before you grant and revoke authority for requests sent to that broker.

About this task

Use WebSphere MQ commands to set up and manage your required security levels. If you prefer, you can make authorization changes to the security queues by using the WebSphere MQ Explorer.

For security reasons, it is important that authorities are set correctly. The **setmqaut** command grants and revokes authorities cumulatively. Therefore, to avoid retaining unwanted pre-existing authorities, it is helpful to set authorities explicitly

on each **setmqaut** command, rather than granting and revoking individual authorities. Granting and revoking is achieved by specifying "-all" (to remove all authorities) followed by the required authorities.

The following command grants execute authority and retains any pre-existing authorities:

```
setmqaut -m test -t queue -n SYSTEM.BROKER.AUTH -g group1 +set
```

The following command grants execute authority only and does not retain pre-existing authorities:

```
setmqaut -m test -t queue -n SYSTEM.BROKER.AUTH -g group1 -all +set
```

Multiple authorities can also be set in this manner. For example, the following command grants execute and write authorities only (and not retain pre-existing authorities):

```
setmqaut -m test -t queue -n SYSTEM.BROKER.AUTH -g group1 -all +set +put
```

It is also helpful to use the **dspmqaut** command after each **setmqaut** command, to check that authorities have been correctly set.

For further information about the commands shown in the following examples, and for details of the parameters, see the WebSphere MQ Version 7 product documentation online.

Examples

All the examples shown here are for a broker that is associated with the queue manager test.

Grant only execute authority to the broker to the user IDs that are defined in the group group1:

```
setmqaut -m test -t queue -n SYSTEM.BROKER.AUTH -g group1 -all +set  
dspmqaut -m test -t queue -n SYSTEM.BROKER.AUTH -g group1
```

Grant only execute and write authority to the broker to the user IDs that are defined in the group group2:

```
setmqaut -m test -t queue -n SYSTEM.BROKER.AUTH -g group2 -all +set +put  
dspmqaut -m test -t queue -n SYSTEM.BROKER.AUTH -g group2
```

Revoke execute authority from the user IDs that are defined in the group group2:

```
setmqaut -m test -t queue -n SYSTEM.BROKER.AUTH -g group2 -set  
dspmqaut -m test -t queue -n SYSTEM.BROKER.AUTH -g group2
```

Using a generic WebSphere MQ profile on a UNIX system, grant only write authority for all integration servers for the user IDs that are defined in the group group3:

```
setmqaut -m test -t queue -n "SYSTEM.BROKER.AUTH.**" -g group3 -all +put  
dspmqaut -m test -t queue -n "SYSTEM.BROKER.AUTH.**" -g group3
```

Note: You enclose generic profile names in quotes on UNIX and Linux systems. For more information see the WebSphere MQ Version 7 product documentation online and search for the “Using OAM generic profiles on UNIX systems and Windows” topic.

Using a generic WebSphere MQ revoke write authority on a UNIX system for all integration servers for the user IDs that are defined in the group group3:

```
setmqaut -m test -t queue -n "SYSTEM.BROKER.AUTH.**" -g group3 -all -put  
dspmqaut -m test -t queue -n "SYSTEM.BROKER.AUTH.**" -g group3
```

Grant only read authority for a specific integration server called default for group group4:

```
setmqaut -m test -t queue -n SYSTEM.BROKER.AUTH.default -g group4 -all +inq  
dspmqaut -m test -t queue -n SYSTEM.BROKER.AUTH.default -g group4
```

Revoke execute and write authority for a specific integration server called default for group group5:

```
setmqaut -m test -t queue -n SYSTEM.BROKER.AUTH.default -g group5 -set -put  
dspmqaut -m test -t queue -n SYSTEM.BROKER.AUTH.default -g group5
```

Using a generic WebSphere MQ on a non-UNIX system, dump all WebSphere MQ authorities for all integration servers:

```
dmpmqaut -m test -t queue -n SYSTEM.BROKER.AUTH.**
```

Related concepts:

“Security overview” on page 203

When you are designing an IBM Integration Bus application it is important to consider the security measures that are needed to protect the information in the system.

“Administration security overview” on page 214

Broker administration security controls the rights of users to complete administrative tasks for a broker and its resources.

Related tasks:

“Enabling administration security” on page 222

Enable administration security on a broker to control which users can complete specific tasks against that broker and its resources.

“Disabling administration security” on page 240

Disable administration security to remove control over a broker and its resources.

“Authorizing users for administration” on page 228

Grant authority to one or more groups or users to authorize them to complete specific tasks against a broker and its resources.

“Granting and revoking authority on z/OS systems”

Grant or revoke authority to one or more groups or users to complete specific tasks against a broker running on z/OS.

Related information:

 [WebSphere MQ Version 7 product documentation](#)

Granting and revoking authority on z/OS systems:

Grant or revoke authority to one or more groups or users to complete specific tasks against a broker running on z/OS.

Before you begin

Before you start:

Activate broker administration security for the broker before you grant and revoke authority for requests sent to that broker.

About this task

Configure the external security manager (ESM) that you are using with WebSphere MQ to grant the required permissions on z/OS systems. For example, if you are using RACF, set up profiles to hold the information required for WebSphere MQ security checking. The examples in this topic assume that you are using RACF.

Complete the following steps:

Procedure

1. Activate security on the queue manager that is associated with the broker.
2. Activate queue security on the same queue manager:
 - If you use uppercase profiles, you must define profiles in MQQUEUE (Member Class) or GMQUEUE (Group Class).
 - If you use mixed case profiles, you must define profiles in MXQUEUE (Member Class) or GMXQUEUE (Group Class).

For example:

- Define the broker profile for queue manager MQ01:
RDEFINE MQQUEUE MQ01.SYSTEM.BROKER.AUTH UACC(NONE)
 - Define the profile for all integration servers on queue manager MQ01:
RDEFINE MXQUEUE MQ01.SYSTEM.BROKER.AUTH.** UACC(NONE)
 - Define a profile for the specific integration server called default for queue manager MQ01:
RDEFINE MXQUEUE MQ01.SYSTEM.BROKER.AUTH.default UACC(NONE)
3. Activate the WebSphere MQ class so that security checks can be made by the broker. For example, activate the class MQQUEUE:
SETROPTS CLASSACT(MQQUEUE)
 4. Define WebSphere MQ permissions. The mapping between broker permissions, associated WebSphere MQ permissions, and associated RACF access levels is shown in the following table.

Broker permission	WebSphere MQ permission	RACF access level
Read	Inquire	READ
Write	Put	UPDATE
Execute	Set	ALTER

Examples

All the examples shown here are for a broker that is associated with the queue manager MQ01.

Add execute permission for group GROUP1 to the broker:

```
PERMIT MQ01.SYSTEM.BROKER.AUTH CLASS(MQQUEUE) ID(GROUP1) ACCESS(ALTER)
```

Remove broker execute permission to the broker for group GROUP2:

```
PERMIT MQ01.SYSTEM.BROKER.AUTH CLASS(MQQUEUE) ID(GROUP2) ACCESS(ALTER) DEL
```

Add write permission to all integration servers for group GROUP3:

```
PERMIT MQ01.SYSTEM.BROKER.AUTH.** CLASS(MXQUEUE) ID(GROUP3) ACCESS(UPDATE)
```

Remove write permission to all integration servers for group GROUP4:

```
PERMIT MQ01.SYSTEM.BROKER.AUTH.** CLASS(MXQUEUE) ID(GROUP4) DEL
```

Add read permission to a specific integration server called default for group GROUP5:

```
PERMIT MQ01.SYSTEM.BROKER.AUTH.default CLASS(MXQUEUE) ID(GROUP5) ACCESS(READ)
```

The following JCL file shows one way in which you can check the RACF permissions that you have set for the broker MQTEST:

```
//RACFDUMP JOB ,MQTEST,USER=MQTEST,TIME=1,MSGCLASS=H
//STEP1 EXEC PGM=IKJEFT01,REGION=64M,DYNAMNBR=99
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
/* LIST ALL EXISTING PROFILES IN THE MQADMIN CLASS */
SEARCH CLASS(MQADMIN)
/* LIST UPPERCASE PROFILES IN THE MQQUEUE MEMBER CLASS */
SEARCH CLASS(MQQUEUE)
/* LIST MIXED CASE PROFILES IN THE MQQUEUE MEMBER CLASS */
SEARCH CLASS(MXQUEUE)
/* LIST THE QMGR PROFILE */
RLIST MQADMIN MI09.NO.SUBSYS.SECURITY ALL
/*
```

This JCL job might produce results like the following output:

```
READY
/* LIST ALL EXISTING PROFILES IN THE MQADMIN CLASS */
READY
SEARCH CLASS(MQADMIN)
EP00.NO.SUBSYS.SECURITY
EP01.NO.SUBSYS.SECURITY
EP02.NO.SUBSYS.SECURITY
EP03.NO.SUBSYS.SECURITY
EP04.NO.SUBSYS.SECURITY
MA00.NO.SUBSYS.SECURITY
MA01.NO.SUBSYS.SECURITY
MA02.NO.SUBSYS.SECURITY
MA03.NO.SUBSYS.SECURITY
MA04.NO.SUBSYS.SECURITY
MA05.NO.SUBSYS.SECURITY
MA06.NO.SUBSYS.SECURITY
MA07.NO.SUBSYS.SECURITY
MA08.NO.SUBSYS.SECURITY
MA09.NO.SUBSYS.SECURITY
MA10.NO.SUBSYS.SECURITY
MA11.ALTERNATE.USER.KMCMUL
MA11.ALTERNATE.USER.KMCMUL3
MA11.ALTERNATE.USER.MA15USR
MA11.CHANNEL.MA11.TO.REG1
MA11.CONTEXT
MA11.NO.CMD.CHECKS
MA11.NO.CMD.RESC.CHECKS
MA11.NO.CMDS.CHECKS
MA11.NO.CMDS.RESC.CHECKS
MA11.NO.SUBSYS.SECURITY
MA11.RESLEVEL
MA12.NO.SUBSYS.SECURITY
MA13.NO.SUBSYS.SECURITY
MA14.NO.SUBSYS.SECURITY
MA15.NO.SUBSYS.SECURITY
MA16.NO.SUBSYS.SECURITY
```

```

MA17.NO.SUBSYS.SECURITY
MA18.NO.SUBSYS.SECURITY
MA19.NO.SUBSYS.SECURITY
MA20.NO.SUBSYS.SECURITY
MI00.NO.SUBSYS.SECURITY
MI01.NO.SUBSYS.SECURITY
MI02.NO.SUBSYS.SECURITY
MI03.NO.SUBSYS.SECURITY
MI04.NO.SUBSYS.SECURITY
MI05.NO.SUBSYS.SECURITY
MI06.NO.SUBSYS.SECURITY
MI07.NO.SUBSYS.SECURITY
MI08.NO.SUBSYS.SECURITY
MI09.ALTERNATE.USER.MI09STC
MI09.ALTERNATE.USER.NHARRIS
MI09.NO.CMD.CHECKS
MI09.NO.CONNECT.CHECKS
MI09.NO.CONTEXT.CHECKS
MI09.NO.SUBSYS.SECURITY
MI10.NO.SUBSYS.SECURITY
MI11.NO.SUBSYS.SECURITY
MI12.NO.SUBSYS.SECURITY
MI13.NO.SUBSYS.SECURITY
MI14.NO.SUBSYS.SECURITY
MI15.NO.SUBSYS.SECURITY
MI16.NO.SUBSYS.SECURITY
MI17.NO.SUBSYS.SECURITY
MI18.NO.SUBSYS.SECURITY
MI19.NO.SUBSYS.SECURITY
MI20.NO.SUBSYS.SECURITY
MI09.CHANNEL.** (G)
MI09.QUEUE.** (G)
READY
/* LIST UPPERCASE PROFILES IN THE MQQUEUE MEMBER CLASS */
READY
  SEARCH CLASS(MQQUEUE)
MA11.INPUT2.QUEUE
MA11.KMBRK
MA11.MA11.DEAD.QUEUE
MA11.MA15
MA11.REG1
MA11.SUBSCRIBER.RESULTS.QUEUE
MA11.SUBSCRIBER3.RESULTS.QUEUE
MA11.SUBSCRIBER4.RESULTS.QUEUE
MA11.SUBSCRIBER5.RESULTS.QUEUE
MA11.SUBSCRIBER6.RESULTS.QUEUE
MA11.SUBSCRIBER9.RESULTS.QUEUE
MA11.SYSTEM.CHANNEL.EVENT
MA11.SYSTEM.CHANNEL.SYNCQ
MA11.SYSTEM.CLUSTER.COMMAND.QUEUE
MA11.SYSTEM.COMMAND.INPUT
MA11.SYSTEM.COMMAND.REPLY.MODEL
MI09.SYSTEM.BROKER.AUTH.SECURITY_EXE
MA11.SYSTEM.BROKER.** (G)
MA11.SYSTEM.** (G)
MA11.** (G)
MI09.** (G)
READY
/* LIST MIXED CASE PROFILES IN THE MQQUEUE MEMBER CLASS */
READY
  SEARCH CLASS(MXQUEUE)
NO ENTRIES MEET SEARCH CRITERIA
READY

```



```

/* LIST THE QMGR PROFILE */
READY
RLIST MQADMIN MI09.NO.SUBSYS.SECURITY ALL
CLASS      NAME
-----
MQADMIN    MI09.NO.SUBSYS.SECURITY

GROUP CLASS NAME
-----
GMQADMIN

RESOURCE GROUPS
-----
NONE

LEVEL  OWNER      UNIVERSAL ACCESS  YOUR ACCESS  WARNING
-----
 00    MQTEST          NONE             NONE         NO

INSTALLATION DATA
-----
NONE

APPLICATION DATA
-----
NONE

SECLEVEL
-----
NO SECLEVEL

CATEGORIES
-----
NO CATEGORIES

SECLABEL
-----
NO SECLABEL

AUDITING
-----
FAILURES(READ)

NOTIFY
-----
NO USER TO BE NOTIFIED

CREATION DATE  LAST REFERENCE DATE  LAST CHANGE DATE
(DAY) (YEAR)    (DAY) (YEAR)         (DAY) (YEAR)
-----
 237   09        237   09             237   09

ALTER COUNT  CONTROL COUNT  UPDATE COUNT  READ COUNT
-----
 000000      000000        000000        000000

USER      ACCESS  ACCESS COUNT
-----
NO USERS IN ACCESS LIST

ID      ACCESS  ACCESS COUNT  CLASS          ENTITY NAME

```

NO ENTRIES IN CONDITIONAL ACCESS LIST
READY
END

Related concepts:

“Security overview” on page 203

When you are designing an IBM Integration Bus application it is important to consider the security measures that are needed to protect the information in the system.

“Administration security overview” on page 214

Broker administration security controls the rights of users to complete administrative tasks for a broker and its resources.

Related tasks:

“Enabling administration security” on page 222

Enable administration security on a broker to control which users can complete specific tasks against that broker and its resources.

“Disabling administration security” on page 240

Disable administration security to remove control over a broker and its resources.

“Authorizing users for administration” on page 228

Grant authority to one or more groups or users to authorize them to complete specific tasks against a broker and its resources.

“Granting and revoking authority on Linux, UNIX, and Windows systems” on page 232

Grant or revoke authority to one or more groups or users to complete specific tasks against a broker running on Linux, UNIX, or Windows.

Controlling access to data and resources in the web user interface:

Broker administrators can control web users' access to data and broker resources by assigning permissions to users based on their role.

Before you begin

Before you start:

- Read the following topics:
 - “Administering brokers using the web user interface” on page 127
 - “Role-based security” on page 220
- Ensure that your web administration server has been configured; for more information, see “Configuring the web user interface server” on page 128
- Sign in to the web administration server, as described in “Accessing the web user interface” on page 133

About this task

Broker administrators can restrict web users' access to data and broker resources only if administration security is enabled. If administration security is disabled, web users can interact with the web UI without logging on, which means that they access the web UI as the 'default' user and have access to all data and broker resources.

As a broker administrator, you can restrict users' access by setting permissions on authorization queues. For example, you can ensure that data technicians see only

their profile and the Data viewer in the web UI, by granting them read (+inq) authority on the SYSTEM.BROKER.DC.AUTH queue, and no permissions on the SYSTEM.BROKER.AUTH queue.

Web users with no permissions on the SYSTEM.BROKER.AUTH queue, but with read (+inq) authority on the SYSTEM.BROKER.DC.AUTH queue, are able to view and download recorded messages. Web users with no permissions on the SYSTEM.BROKER.AUTH queue, but with read (+inq) and execute (+set) authority on the SYSTEM.BROKER.DC.AUTH queue, are able to view, download, and replay recorded messages.

With administration security enabled, REST users can view only the URIs for which they are authorized. For example, a user with no permissions on the SYSTEM.BROKER.AUTH queue, but with read (+inq) authority on the SYSTEM.BROKER.DC.AUTH queue, can request information about messages recorded under a DataCaptureStore, whereas a user with read and execute (+inq and +set) authority on the SYSTEM.BROKER.DC.AUTH queue can view and replay messages. If administration security is disabled, all REST requests are unrestricted.

Broker administrators can also allow web users to start and stop integration servers, applications, and message flows from the web user interface, by granting permissions to the roles that are associated with the users.

For more information about role-based access, see “Role-based security” on page 220 and “Managing web user accounts” on page 226.

Related concepts:



The IBM Integration API

The IBM Integration API is a programming interface that your applications can use to control brokers and their resources through a remote interface.

Related tasks:

“Administering brokers using the web user interface” on page 127

You can use the IBM Integration Bus web user interface to administer broker resources.

“Configuring the web user interface server” on page 128

To enable access to broker resources through the web user interface, configure the IBM Integration Bus web user interface server.

“Accessing the web user interface” on page 133

You can access broker resources by logging on to the web user interface.

“Recording, viewing, and replaying data” on page 595

Record data that is processed by a message flow, or emitted from WebSphere Application Server. View and replay the data.

Disabling administration security

Disable administration security to remove control over a broker and its resources.

About this task

You can disable administrative security for a broker by updating the relevant broker property.

Procedure

1. Stop the broker in the IBM Integration Explorer, or run the `mqsistop` command.

2. Clear the security option for this broker in the IBM Integration Explorer, or run the **mqsichangebroker** command, specifying the parameter **-s inactive**. For example, to disable security for the broker IB9NODE, enter the following command:

```
mqsichangebroker IB9NODE -s inactive
```
3. Start the broker in the IBM Integration Explorer, or run the **mqsistart** command.

Results

When this command completes, all users are able to complete tasks against the broker; for example, they can run the **mqsichangeproperties** command to change broker properties. They can also complete all tasks against all integration servers that have been defined, and are defined in the future, on this broker.

When you disable broker administration security, the authorization queues that are associated with this broker are retained. If you enable security again, these queues are reused, therefore you must ensure that the authorizations defined by these queues are correct.

Related concepts:

“Security overview” on page 203

When you are designing an IBM Integration Bus application it is important to consider the security measures that are needed to protect the information in the system.

“Administration security overview” on page 214

Broker administration security controls the rights of users to complete administrative tasks for a broker and its resources.

Related tasks:

“Enabling administration security” on page 222

Enable administration security on a broker to control which users can complete specific tasks against that broker and its resources.

“Authorizing users for administration” on page 228

Grant authority to one or more groups or users to authorize them to complete specific tasks against a broker and its resources.

Related reference:



mqsichangebroker command

Use the **mqsichangebroker** command to change one or more of the configuration parameters of the broker.

Activating broker administration security for WebSphere MQ Version 7.1, or later

Describes how to enable channel authentication security and the effect of this on a broker queue manager.

About this task

When you create a broker, if the WebSphere MQ queue manager does not exist, the queue manager is automatically created. If WebSphere MQ Version 7.1, or later, has been selected for the queue manager, the channel authentication security will be automatically disabled.

If more precise control over the access granted to connecting systems is required at a channel level, channel authentication security can be enabled. For more details, see WebSphere MQ Version 7.1 Channel authentication records

To enable channel authentication security in order to start using channel authentication records you must run this MQSC command:

```
ALTER QMGR CHLAUTH(ENABLED)
```

Once enabled, there will be consequences for any channel based communication with a broker. However, this should not affect any privileged or non-privileged user from accessing local brokers. See following table for a definition of privileged users who have full administrative authorities:

Table 2. . Privileged users by platform.

Platform	Privileged users
Windows systems	<ul style="list-style-type: none"> • SYSTEM • Members of the mqm group • Members of the Administrators group
UNIX and Linux systems	<ul style="list-style-type: none"> • Members of the mqm group

Privileged or non-privileged users wanting to remotely administer a broker by means of CMP/API/MBX/Toolkit must run the following commands in order to grant their user access:

Procedure

1. Enable remote administration on Queue Manager
2. `setmqaut -m QMNAME -n SYSTEM.MQEXPLORER.REPLY.MODEL -t queue -p username +dsp +inq +put +get`
3. `SET CHLAUTH('SYSTEM.BKR.CONFIG') TYPE (ADDRESSMAP) ADDRESS('address-of-machine-who-is-allowed') MCAUSER('NonPrivilegedUser') ACTION(ADD)`

Where 'NonPrivilegedUser' is a user defined on the remote machine.

What to do next

The rule would need to be set up for any ip-address wanting to administer the broker remotely.

Related concepts:

“Security overview” on page 203

When you are designing an IBM Integration Bus application it is important to consider the security measures that are needed to protect the information in the system.

Related reference:



mqsi createbroker command

Use the **mqsi createbroker** command to create a broker and its associated resources.

Message flow security

Control access to individual messages in a message flow, using the identity of the messages.

The following topics introduce the concepts that you need to understand before you can configure message flow security, and they explain the steps involved in setting up security for your message flows:

- “Message flow security overview”
- “Setting up message flow security” on page 292

Related concepts:

Chapter 2, “Security,” on page 203

Security is an important consideration for both developers of IBM Integration Bus applications, and for system administrators configuring IBM Integration Bus authorities.

“Security overview” on page 203

When you are designing an IBM Integration Bus application it is important to consider the security measures that are needed to protect the information in the system.



WS-Security

Web Services Security (WS-Security) describes enhancements to SOAP messaging to provide quality of protection through message integrity, message confidentiality, and single message authentication. WS-Security mechanisms can be used to accommodate a wide variety of security models and encryption technologies.

Related tasks:

“Setting up administration security” on page 221

Control the actions that users can request against a broker and its resources.

Message flow security overview

IBM Integration Bus provides a security manager, which enables you to control access to individual messages in a message flow, using the identity of the message.

You can configure an integration node for processing end-to-end an identity that is carried in a message through a message flow by using a security profile. By creating a security profile, you can configure security for a message flow to control access based on the identity that is associated with the message and provides a security mechanism that is independent of both the transport type and message format.

Only a subset of the connectors available in IBM Integration Bus use security profiles to control and vary the identity that is used when the connector interacts with an external system. For other connectors, a fixed identity can be specified, which is used to authorize access to the external system. For those connectors, the integration node has its own repository of identities, which can be updated with the `mqsisetdbparms` and `mqsireportdbparms` command. For more information about which external systems use one of these identities to connect, see Advanced configuration.

If you do not enable message flow security, the default security facilities that are in IBM Integration Bus are based on the security facilities that are provided by the transport mechanism. In this case, the integration node processes all messages that are delivered to it, using the integration node service identity as a proxy identity for all message instances. Any identity that is present in the incoming message is ignored.

The security manager enables the integration node to:

- Extract the identity from an inbound message

- Authenticate the identity (by using an external security provider)
- Map the identity to an alternative identity (by using an external security provider)
- Check that either the alternative identity or the original identity is authorized to access the message flow (by using an external security provider)
- Propagate either the alternative identity or the original identity with an outbound message.

The security functions that are associated with a message flow are controlled by using “Security profiles” on page 247, which are created by the broker administrator and accessed by the security manager at run time. The following external security providers (also known as Policy Decision Points or PDPs) are supported:

- WS-Trust V1.3 compliant security token servers (including TFIM V6.2) for authentication, mapping, and authorization
- Tivoli® Federated Identity Manager (TFIM) V6.1 for authentication, mapping, and authorization
- Lightweight Directory Access Protocol (LDAP) for authentication and authorization
- Windows Domain Controllers for authentication
- Kerberos KDCs for authentication

You can invoke message flow security by configuring either a security enabled input node or a SecurityPEP node. The SecurityPEP node enables you to invoke the message flow security manager at any point in the message flow between an input node and an output (or request) node.

For an overview of the sequence of events that occur when the message flow security manager is invoked using either a security enabled input node or a SecurityPEP node, see the following topics:

- “Invoking message flow security using a security enabled input node” on page 266
- “Invoking message flow security using a SecurityPEP node” on page 272

The input nodes that support message flow security are:

- MQInput
- HTTPInput
- SCAInput
- SCAAsyncResponse
- SOAPInput

However, the support for treating security exceptions as normal exceptions is provided by only the MQInput, HTTPInput, SCAInput, and SCAAsyncResponse nodes; it is not available in the SOAPInput node.

The output nodes that support identity propagation are:

- MQOutput
- HTTPRequest
- SCAResponse
- SCAAsyncRequest
- SOAPRequest

- SOAPAsyncRequest

If the message flow is a Web Service that is implemented by using the SOAP nodes, the identity can be taken from the WS-Security header tokens that are defined through appropriate Policy sets and bindings.

To improve performance, the authentication, authorization, and mapping information from the configured providers is cached for reuse. You can use the **mqsireloadsecurity** command to reload the security cache, and you can use the **mqsichangeproperties** command to set the expiry and sweep intervals for the security cache.

For a SOAPRequest and SOAPAsyncRequest node, an appropriate policy set and bindings can be defined to specify how the token is placed in the WS-Security header (rather than the underlying transport headers). For more information, see Policy sets.

The following topics in this section provide more detailed information about message flow security:

- “Identity” on page 251
- “Security profiles” on page 247
- “Authentication and validation” on page 257
- “Authorization” on page 261
- “Identity mapping” on page 263
- “Identity and security token propagation” on page 286
- “Security identities for integration nodes connecting to external systems” on page 289
- “Invoking message flow security using a security enabled input node” on page 266
- “Invoking message flow security using a SecurityPEP node” on page 272
- “Authentication, mapping, and authorization with TFIM V6.2 and TAM” on page 280
- “Authentication, mapping, and authorization with TFIM V6.1 and TAM” on page 277
- “Security exception processing” on page 290

Related concepts:

“Invoking message flow security using a security enabled input node” on page 266
You can invoke the message flow security manager by configuring a security enabled input node.

“Invoking message flow security using a SecurityPEP node” on page 272
You can invoke the message flow security manager at any point in a message flow, between an input node and an output or request node, by using a SecurityPEP node.

Related tasks:

“Setting up message flow security” on page 292

Set up security on a message flow to control access based on the identity of a message passing through the message flow.

“Configuring the extraction of an identity or security token” on page 308

You can configure the SecurityPEP node or security enabled input nodes to extract the identity or security token from a message and store it in the properties tree identity fields, enabling it to be processed throughout the message flow and

propagated at output or request nodes.

“Creating a security profile” on page 294

You can create a security profile for use with Lightweight Directory Access Protocol (LDAP) or a WS-Trust V1.3 compliant security token server (STS), such as Tivoli Federated Identity Manager (TFIM) V6.2. Support is also provided for TFIM V6.1, for compatibility with previous versions of IBM Integration Bus. You can create the security profile by using either the **mqsicreateconfigurable-service** command or an editor in the IBM Integration Explorer.

“Configuring identity authentication and security token validation” on page 312

You can configure a message flow to perform identity authentication or security token validation using Integrated Windows Authentication (IWA), Lightweight Directory Access Protocol (LDAP), a WS-Trust V1.3 compliant security token sever (STS) such as Tivoli Federated Identity Manager (TFIM) Version 6.2, or HTTP requests. Support for TFIM V6.1 is also provided, for compatibility with previous versions of IBM Integration Bus.

“Configuring identity mapping” on page 333

Configure a security token server (STS), such as Tivoli Federated Identity Manager (TFIM) V6.2, to map the incoming security token and, if required, to authenticate and authorize it.

“Configuring authorization” on page 342

Configure the broker to use an external security provider to authorize an identity in a message flow. You can use either Lightweight Directory Access Protocol (LDAP) or a WS-Trust V1.3 compliant security token server (STS) such as Tivoli Federated Identity Manager (TFIM) V6.2. Support for TFIM V6.1 is also provided for compatibility with previous versions.

“Configuring for identity propagation” on page 389

To enable a message flow to perform identity propagation, the input nodes must extract the identity and the output node must propagate it.

Related reference:



mqsicreateconfigurable-service command

Use the **mqsicreateconfigurable-service** command to create an object name for a broker external resource.



mqsdeleteconfigurable-service command

Use the **mqsdeleteconfigurable-service** command to delete a configurable service, such as a JMS provider, JDBC provider, or FTP server, that you have created by using the **mqsicreateconfigurable-service** command.



mqschangeproperties command

Use the **mqschangeproperties** command to modify broker properties and properties of broker resources.



mqsireportproperties command

Use the **mqsireportproperties** command to display properties that relate to a broker, an integration server, or a configurable service.



mqsireloadsecurity command

Use the **mqsireloadsecurity** command to force the immediate expiry of some or all the entries in the security cache.



Parameter values for the securitycache component

Select the objects and properties associated with the securitycache component that you want to change.

MQInput node

Use the MQInput node to receive messages from clients that connect to the broker by using the WebSphere MQ Enterprise Transport, and that use the MQI and AMI application programming interfaces.

HTTPInput node

Use the HTTPInput node to receive an HTTP message from an HTTP client for processing by a message flow.

SOAPInput node

Use the SOAPInput node to process client SOAP messages, so that the broker operates as a SOAP Web Services provider.

SOAPAsyncRequest node

Use the SOAPAsyncRequest node with the SOAPAsyncResponse node to construct a pair of message flows that call a Web service asynchronously.

SCAInput node

Use the SCAInput node with the SCAREply node to process messages from WebSphere Process Server.

SecurityPEP node

Use the SecurityPEP node in a message flow to invoke the message flow security manager at any point in the message flow.

HTTPRequest node

Use the HTTPRequest node to interact with a web service.

MQOutput node

Use the MQOutput node to send messages to clients that connect to the broker using the WebSphere MQ Enterprise Transport and that use the MQI and AMI application programming interfaces.

Security profiles

A security profile defines the security operations that are to be performed in a message flow at SecurityPEP nodes and security enabled input and output nodes.

Security profiles are configured by the broker administrator before deploying a message flow, and are accessed by the security manager at run time.

A security profile allows a broker administrator to specify whether identity and security token propagation, authentication, authorization, and mapping are performed on the identity or security tokens associated with messages in the message flow, and if so, which external security provider (also known as a Policy Decision Point or PDP) is used. IBM Tivoli Federated Identity Manager (TFIM) V6.1, and WS-Trust v1.3 compliant security token servers (including TFIM V6.2), are supported for authentication, authorization, and mapping. Lightweight Directory Access Protocol (LDAP) is supported for authentication and authorization.

Security profiles apply to the SecurityPEP node and to security enabled input, output, and request nodes, and are configured by the administrator at deployment time in the Broker Archive editor. These nodes have a **Security Profile** property (in the Broker Archive editor), which can be left blank, set to *No Security*, or set to a specific security profile name. Set *No Security* to explicitly turn off security for the node. If you leave the **Security Profile** property blank, the node inherits the

Security Profile property that is set at the message flow level. If you leave the **Security Profile** property blank at both levels, security is turned off for the node. When this property is set to the name of a specific security profile, that profile determines what message flow security is configured. If the named security profile does not exist in the run time, the message flow fails to deploy. If the specified external security provider does not support the type of token configured on the node for the security operation, an error is reported and the message flow fails to deploy.

The security profile also specifies whether propagation is required. A pre-configured profile that specifies propagation is provided for use by output and request nodes. This profile is the *Default Propagation* security profile. This profile can also be used on an input node to extract tokens and put them into the message tree ready for propagation or processing in a SecurityPEP node.

Security profiles contain values for the following properties:

authentication

Defines the type of authentication that is performed on the source identity. This property applies only to SecurityPEP nodes and input nodes. For more information, see “Authentication and validation” on page 257.

authenticationConfig

Defines the information that the broker needs to connect to the provider, and the information needed to look up the identity tokens. It is a provider-specific configuration string. This property applies only to SecurityPEP nodes and input nodes.

mapping

Defines the type of mapping that is performed on the source identity. This property applies only to SecurityPEP nodes and input nodes. For more information, see “Identity mapping” on page 263.

mappingConfig

Defines how the broker connects to the provider, and contains additional information required to look up the mapping routine. It is a provider-specific configuration string. This property applies only to SecurityPEP nodes and input nodes.

authorization

Defines the types of authorization checks that are performed on the mapped or source identity. This property applies only to SecurityPEP nodes and input nodes. For more information, see “Authorization” on page 261.

authorizationConfig

Defines how the broker connects to the provider, and contains additional information that can be used to check access (for example, a group that can be checked for membership). It is a provider-specific configuration string. This property applies only to SecurityPEP nodes and input nodes.

passwordValue

Defines how passwords are treated when they enter a message flow. If *PLAIN* is selected, the password appears in the Properties folder in plain text. If *OBFUSCATE* is selected, the password appears in the Properties folder in base64 encoding. If *MASK* is selected, the password appears in the Properties folder as four asterisks (****). This property applies only to SecurityPEP nodes and input nodes.

Propagation

Enables or disables identity propagation on output and request nodes. On the security enabled input nodes, you can choose to select only identity propagation, without specifying any other security operations, to make the extracted incoming identity or security token available for use in the other nodes in the message flow, such as output or request nodes. For more information, see “Identity and security token propagation” on page 286.

For information on configuring a security profile for LDAP, TFIM, or a WS-Trust v1.3 compliant security token server (STS), see “Creating a security profile” on page 294.

Related concepts:

“Identity” on page 251

In IBM Integration Bus, an identity is a security token that uniquely identifies an individual, or that provides a set of assertions that can be validated.

“Authentication and validation” on page 257

Authentication is the process of establishing the identity of a user or system and verifying that the identity is valid. Applying authentication to a SAML security token involves validating the assertions that it carries and confirming that it is being processed within its validity period.

“Identity mapping” on page 263

Identity mapping is the transformation of a security token from one format to another format, or the federation of an identity from one realm to an equivalent identity in another realm.

“Authorization” on page 261

Authorization is the process of verifying that an identity token has permission to access a message flow.

“Identity and security token propagation” on page 286

Identity and security token propagation enables the identity and security tokens (associated with each message) to be propagated throughout a message flow, and on to target applications through output or request nodes.

Related tasks:

“Configuring the extraction of an identity or security token” on page 308

You can configure the SecurityPEP node or security enabled input nodes to extract the identity or security token from a message and store it in the properties tree identity fields, enabling it to be processed throughout the message flow and propagated at output or request nodes.

“Configuring identity authentication and security token validation” on page 312

You can configure a message flow to perform identity authentication or security token validation using Integrated Windows Authentication (IWA), Lightweight Directory Access Protocol (LDAP), a WS-Trust V1.3 compliant security token sever (STS) such as Tivoli Federated Identity Manager (TFIM) Version 6.2, or HTTP requests. Support for TFIM V6.1 is also provided, for compatibility with previous versions of IBM Integration Bus.

“Configuring identity mapping” on page 333

Configure a security token server (STS), such as Tivoli Federated Identity Manager (TFIM) V6.2, to map the incoming security token and, if required, to authenticate and authorize it.

“Creating a security profile” on page 294

You can create a security profile for use with Lightweight Directory Access Protocol (LDAP) or a WS-Trust V1.3 compliant security token server (STS), such as Tivoli Federated Identity Manager (TFIM) V6.2. Support is also provided for TFIM V6.1, for compatibility with previous versions of IBM Integration Bus. You can create the security profile by using either the **mqscreateconfigurable** command or an editor in the IBM Integration Explorer.

“Configuring authorization” on page 342

Configure the broker to use an external security provider to authorize an identity in a message flow. You can use either Lightweight Directory Access Protocol (LDAP) or a WS-Trust V1.3 compliant security token server (STS) such as Tivoli Federated Identity Manager (TFIM) V6.2. Support for TFIM V6.1 is also provided for compatibility with previous versions.

“Configuring for identity propagation” on page 389

To enable a message flow to perform identity propagation, the input nodes must extract the identity and the output node must propagate it.

“Setting up message flow security” on page 292

Set up security on a message flow to control access based on the identity of a message passing through the message flow.

Related reference:

 **mqscreateconfigurable** command

Use the **mqscreateconfigurable** command to create an object name for a broker external resource.

 **mqsdeleteconfigurable** command

Use the **mqsdeleteconfigurable** command to delete a configurable service, such as a JMS provider, JDBC provider, or FTP server, that you have created by using the **mqscreateconfigurable** command.

 **mqschangeproperties** command

Use the **mqschangeproperties** command to modify broker properties and properties of broker resources.

 **mqsreportproperties** command

Use the **mqsreportproperties** command to display properties that relate to a broker, an integration server, or a configurable service.

 MQInput node

Use the MQInput node to receive messages from clients that connect to the broker by using the WebSphere MQ Enterprise Transport, and that use the MQI and AMI application programming interfaces.

 HTTPInput node


Use the HTTPInput node to receive an HTTP message from an HTTP client for processing by a message flow.


 SOAPInput node


Use the SOAPInput node to process client SOAP messages, so that the broker operates as a SOAP Web Services provider.

 SOAPAsyncRequest node

Use the SOAPAsyncRequest node with the SOAPAsyncResponse node to construct a pair of message flows that call a Web service asynchronously.

 HTTPRequest node
Use the HTTPRequest node to interact with a web service.

 MQOutput node
Use the MQOutput node to send messages to clients that connect to the broker using the WebSphere MQ Enterprise Transport and that use the MQI and AMI application programming interfaces.

 SecurityPEP node
Use the SecurityPEP node in a message flow to invoke the message flow security manager at any point in the message flow.

Identity

In IBM Integration Bus, an identity is a security token that uniquely identifies an individual, or that provides a set of assertions that can be validated.

When a SecurityPEP node or a supported input node is configured with a security profile, the extracted identity is held in the broker as eight properties in the Properties folder of the message tree structure. These properties define two identities in the broker: source and mapped. For both the source and mapped identities, values are held for **Type**, **Token**, **Password**, and **IssuedBy** properties:

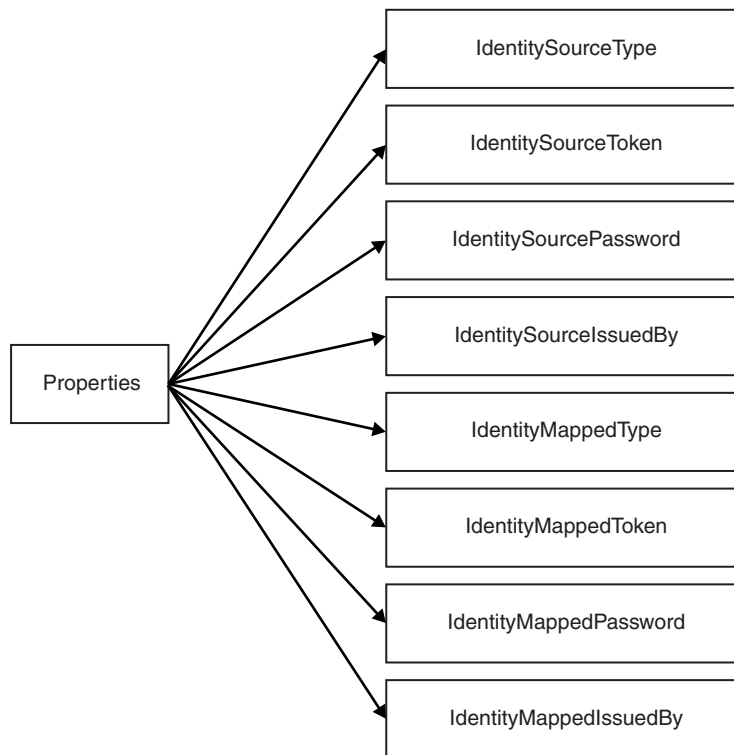


Figure 16. Diagram showing the eight identity properties.

The **Identity token type** property on the security-enabled input nodes can be set to a value of *Transport Default*, which causes the token type to be created from the default identity header or fields of the transport. For WebSphere MQ, *Transport Default* provides an identity type of *Username*. For HTTP, *Transport Default* provides an identity type of *Username and Password*. The Identity token type property on the

SecurityPEP node can be set to *Current Token*, which enables it to use the identity in the Properties folder fields instead of extracting a new identity from the message.

The following table shows the support that is provided (by the message flow security manager and external security providers) for the extraction of different types of security token. For information about the token types that are supported for identity propagation, see “Identity and security token propagation” on page 286.

Table 3. Support for security token types - token extraction

Token type (format)	Broker security manager support for token extraction	External security provider support
Username	<p>Username tokens are supported for extraction by the following nodes:</p> <ul style="list-style-type: none"> • HTTPInput • MQInput • SCAInput • SCAAsyncResponse • SecurityPEP • SOAPInput <p>The token is obtained from one of the following transport headers:</p> <ul style="list-style-type: none"> • MQ <ul style="list-style-type: none"> – From MQMD user ID • HTTP <ul style="list-style-type: none"> – From HTTP BasicAuth header containing only a username part • SOAP <ul style="list-style-type: none"> – From a WS-Security UsernameToken header. The policy set and binding (associated with the SOAP node) must define a username profile, and the wsse:UsernameToken must contain only a wsse:Username element. – From the Kerberos subject in a WS-Security header. The policy set and binding (associated with the SOAP node) must define a Kerberos profile. – From the HTTP BasicAuth header containing only a username part if no policy set is defined on the SOAP node. <p>Alternatively, the token can be taken from any part of the message tree when the token location is specified (on the node) using an XPath expression or ESQL field path.</p> <p>The literal string value used by the broker (and which you can use to specify the token type in an ESQL or Java program) is <i>username</i>.</p>	LDAP: Authorization

Table 3. Support for security token types - token extraction (continued)

Token type (format)	Broker security manager support for token extraction	External security provider support
<p>Username and password or Username and RACF PassTicket</p>	<p>Username and password tokens are supported for extraction by the following nodes:</p> <ul style="list-style-type: none"> • HTTPInput • MQInput • SCAInput • SecurityPEP • SCAAsyncResponse • SOAPInput <p>The token is obtained from one of the following transport headers:</p> <ul style="list-style-type: none"> • HTTP <ul style="list-style-type: none"> – From HTTP BasicAuth header containing both a username and password part • SOAP <ul style="list-style-type: none"> – From a WS-Security UsernameToken header. The policy set and binding (associated with the SOAP node) must define a username profile and the wsse:UsernameToken must contain both wsse:Username and wsse:Password elements. – From the HTTP BasicAuth header containing only a username part if no policy set is defined on the SOAP node <p>Alternatively, the token can be obtained from any part of the message tree when the token location is specified (on the node) using an XPath expression or ESQL path.</p> <p>The password token can carry either a clear text password or a RACF PassTicket. If you are using a WS-Trust V1.3 STS (such as TFIM V6.2), you can use it to map (issue) or validate RACF PassTickets, by specifying the token type as <i>Username and password</i>. This support is available with security enabled input nodes and SecurityPEP nodes.</p> <p>The literal string value used by the broker (and which you can use to specify the token type in an ESQL or Java program) is <i>usernameAndPassword</i>.</p>	<p>LDAP:</p> <ul style="list-style-type: none"> • Authentication • Authorization <p>TFIM V6.1:</p> <ul style="list-style-type: none"> • Authentication • Mapping • Authorization <p>WS-Trust V1.3 STS (including TFIM V6.2):</p> <ul style="list-style-type: none"> • Authentication • Mapping • Authorization <p>IWA:</p> <ul style="list-style-type: none"> • Authentication

Table 3. Support for security token types - token extraction (continued)

Token type (format)	Broker security manager support for token extraction	External security provider support
SAML assertion	<p>SAML tokens are supported for extraction by the following nodes:</p> <ul style="list-style-type: none"> • SecurityPEP • MQInput • HTTPInput • SCAInput • SCAAsyncResponse • SOAPInput <p>The token is obtained from one of the following places:</p> <ul style="list-style-type: none"> • SOAP <ul style="list-style-type: none"> – From a WS-Security header. The policy set and binding (associated with the SOAP node) must define a SAML profile. • Any part of the message tree when the token location is specified using an XPath expression or ESQL path. <p>The literal string value used by the broker (and which you can use to specify the token type in an ESQL or Java program) is <i>SAML</i>.</p>	<p>WS-Trust V1.3 STS (including TFIM V6.2):</p> <ul style="list-style-type: none"> • Authentication • Mapping • Authorization
Kerberos GSS v5 AP_REQ	<p>Kerberos tickets are supported for processing by the message flow security manager from the SecurityPEP node. A WS-Security Kerberos token profile is supported by the following SOAP nodes, but in this case, the Kerberos Key Distribution Center is communicated with directly, and the properties folder is populated with a Username token representing the Kerberos subject:</p> <ul style="list-style-type: none"> • SOAPInput <p>The token is obtained from any part of the message tree at a SecurityPEP node, if the token location is specified using an XPath expression or ESQL path.</p> <p>The literal string value used by the broker (and which you can use to specify the token type in an ESQL or Java program) is <i>kerberosTicket</i>.</p>	<p>WS-Trust V1.3 STS (including TFIM V6.2):</p> <ul style="list-style-type: none"> • Authentication • Mapping • Authorization
LTPA v2 token	<p>LTPA tokens are supported for extraction by the following nodes:</p> <ul style="list-style-type: none"> • SecurityPEP • SOAPInput <p>The token is obtained from any part of the message tree at a SecurityPEP node, if the token location is specified using an XPath expression or ESQL path.</p> <p>The literal string value used by the broker (and which you can use to specify the token type in an ESQL or Java program) is <i>LTPA</i>.</p>	<p>WS-Trust V1.3 STS (including TFIM V6.2):</p> <ul style="list-style-type: none"> • Authentication • Mapping • Authorization

Table 3. Support for security token types - token extraction (continued)

Token type (format)	Broker security manager support for token extraction	External security provider support
X.509 Certificate	<p>X.509 tokens are supported for extraction by the following nodes:</p> <ul style="list-style-type: none"> • SecurityPEP • MQInput • HTTPInput • SCAInput • SCAAsyncResponse • SOAPInput <p>The token is obtained from one of the following places:</p> <ul style="list-style-type: none"> • SOAP <ul style="list-style-type: none"> – From a WS-Security header. The policy set and binding (associated with the SOAP node) must define an X.509 profile. • Any part of the message tree when the token location is specified using an XPath expression or ESQL path. <p>The literal string value used by the broker (and which you can use to specify the token type in an ESQL or Java program) is <i>X.509</i>.</p>	<p>TFIM V6.1:</p> <ul style="list-style-type: none"> • Authentication • Mapping • Authorization. <p>WS-Trust V1.3 STS (including TFIM V6.2):</p> <ul style="list-style-type: none"> • Authentication • Mapping • Authorization.
Universal WSSE token	<p>Universal WSSE tokens are supported for extraction by the SecurityPEP node only.</p> <p>The token is obtained from any part of the message tree at a SecurityPEP node, if the token location is specified using an XPath expression or ESQL path.</p> <p>The literal string value used by the broker (and which you can use to specify the token type in an ESQL or Java program) is <i>UniversalWsse</i>.</p>	<p>WS-Trust V1.3 STS (including TFIM V6.2):</p> <ul style="list-style-type: none"> • Authentication • Mapping • Authorization.

The source identity is set by the SecurityPEP or input node only if a security profile is associated with the node. The information to complete these fields is typically found in the headers of a message but can also be located in the body (provided that the node has been configured with an ESQL Path or XPath reference for the various properties). If multiple identities are available (for example, if you are using message aggregation), the first identity is used. The token extraction is transport specific and can be performed only using transports that support the flow of identities. These transports are: Websphere MQ, HTTP(S), and SOAP. See MQInput node and HTTPInput node for more information.

You can modify the values in the properties (for example, from ESQL), but do not write to the *IdentitySource** values. For example, you can create a custom identity mapping routine in ESQL or Java by using the *IdentitySource** values to create custom *IdentityMapped** values.

SAML and Universal WSSE tokens are stored in the Properties tree IdentitySourceToken or IdentityMappedToken field as a character bit stream. To access this data as a message tree, parse it into a suitable parser, such as XMLNSC:

```
-- Parse the mapped SAML2.0 token in the properties folder and set it in the message body
CREATE LASTCHILD OF OutputRoot DOMAIN('XMLNSC') PARSE(InputRoot.Properties.IdentityMappedToken,
InputProperties.Encoding, InputProperties.CodedCharSetId);
```

To set either SAML or Universal WSSE tokens into the properties fields, you must obtain the bit stream of a tree; for example, by using the ESQLEASBITSTREAM function.

Related concepts:

“Authentication and validation” on page 257

Authentication is the process of establishing the identity of a user or system and verifying that the identity is valid. Applying authentication to a SAML security token involves validating the assertions that it carries and confirming that it is being processed within its validity period.

“Identity mapping” on page 263

Identity mapping is the transformation of a security token from one format to another format, or the federation of an identity from one realm to an equivalent identity in another realm.

“Authorization” on page 261

Authorization is the process of verifying that an identity token has permission to access a message flow.

“Identity and security token propagation” on page 286

Identity and security token propagation enables the identity and security tokens (associated with each message) to be propagated throughout a message flow, and on to target applications through output or request nodes.

“Security profiles” on page 247

A security profile defines the security operations that are to be performed in a message flow at SecurityPEP nodes and security enabled input and output nodes.

“Integrated Windows Authentication” on page 260

Integrated Windows Authentication (IWA) refers to a set of authentication protocols, NTLM, Kerberos, and SPNEGO, that are used to provide transport-level security. You can configure IBM Integration Bus to provide an IWA-secured service on a broker running on any operating system, and to consume an IWA-secured service on a broker running on Windows, when you are using the HTTP and SOAP nodes.

Related tasks:

“Configuring the extraction of an identity or security token” on page 308

You can configure the SecurityPEP node or security enabled input nodes to extract the identity or security token from a message and store it in the properties tree identity fields, enabling it to be processed throughout the message flow and propagated at output or request nodes.

“Configuring identity authentication and security token validation” on page 312

You can configure a message flow to perform identity authentication or security token validation using Integrated Windows Authentication (IWA), Lightweight Directory Access Protocol (LDAP), a WS-Trust V1.3 compliant security token server (STS) such as Tivoli Federated Identity Manager (TFIM) Version 6.2, or HTTP requests. Support for TFIM V6.1 is also provided, for compatibility with previous versions of IBM Integration Bus.

“Configuring identity mapping” on page 333

Configure a security token server (STS), such as Tivoli Federated Identity Manager (TFIM) V6.2, to map the incoming security token and, if required, to authenticate and authorize it.

“Providing credentials in HTTP requests” on page 328

Use a security profile to configure HTTPRequest and SOAPRequest nodes to authenticate with a remote server.

“Creating a security profile” on page 294

You can create a security profile for use with Lightweight Directory Access Protocol (LDAP) or a WS-Trust V1.3 compliant security token server (STS), such as Tivoli Federated Identity Manager (TFIM) V6.2. Support is also provided for TFIM V6.1, for compatibility with previous versions of IBM Integration Bus. You can create the security profile by using either the **mqs:createconfigurable-service** command or an editor in the IBM Integration Explorer.

“Configuring authorization” on page 342

Configure the broker to use an external security provider to authorize an identity in a message flow. You can use either Lightweight Directory Access Protocol (LDAP) or a WS-Trust V1.3 compliant security token server (STS) such as Tivoli Federated Identity Manager (TFIM) V6.2. Support for TFIM V6.1 is also provided for compatibility with previous versions.

“Configuring for identity propagation” on page 389

To enable a message flow to perform identity propagation, the input nodes must extract the identity and the output node must propagate it.

Related reference:



MQInput node

Use the MQInput node to receive messages from clients that connect to the broker by using the WebSphere MQ Enterprise Transport, and that use the MQI and AMI application programming interfaces.



HTTPInput node

Use the HTTPInput node to receive an HTTP message from an HTTP client for processing by a message flow.



SecurityPEP node

Use the SecurityPEP node in a message flow to invoke the message flow security manager at any point in the message flow.

Authentication and validation

Authentication is the process of establishing the identity of a user or system and verifying that the identity is valid. Applying authentication to a SAML security token involves validating the assertions that it carries and confirming that it is being processed within its validity period.

In IBM Integration Bus message flow security, authentication involves the security manager passing the identity type and token to an external security provider. For more information about security tokens, see “Identity” on page 251.

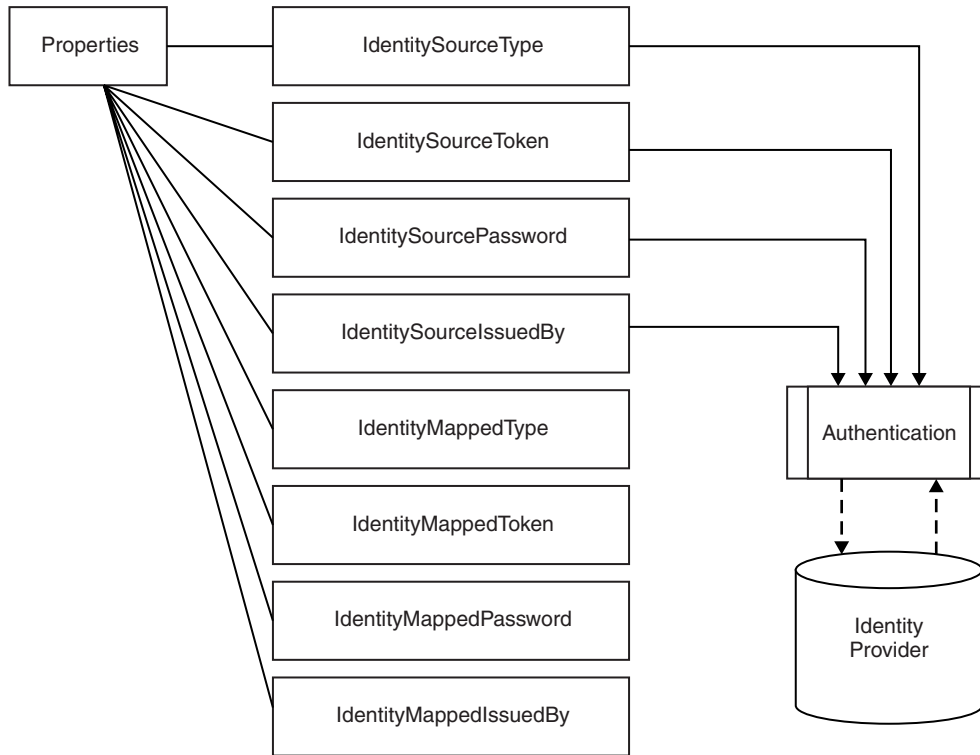


Figure 17. Diagram showing the flow of identity authentication.

The following external security providers (also known as Policy Decision Points) are supported for authentication:

- Lightweight Directory Access Protocol (LDAP)
- Tivoli Federated Identity Manager (TFIM) V6.1
- WS-Trust V1.3 security token servers (STS), including TFIM V6.2
- Windows domain controller and Kerberos Key Distribution Center

The external security provider checks the identities and returns a value to confirm whether the identity is authentic. If the identity is not authentic, a security exception is raised.

Some identity providers support only a single type of authentication token. If a token of another type is passed into the message flow, an exception is raised. For example, LDAP supports only a *Username and password* token.

You can use an LDAP provider for the authentication of an incoming identity token. The LDAP server must be LDAP Version 3 compliant.

Alternatively, you can use a WS-Trust v1.3 STS provider (for example, TFIM Version 6.2) for the authentication of an incoming identity or security token. The security manager invokes the WS-Trust v1.3 provider once, even if it is set for additional security operations (such as mapping or authorization). As a result, when you are using TFIM, you must configure a single module chain to perform all the required authentication, mapping, and authorization operations.

For more information about using TFIM V6.2 for authentication, see “Authentication, mapping, and authorization with TFIM V6.2 and TAM” on page 280.

TFIM V6.1 is also supported, for compatibility with previous versions of IBM Integration Bus. For more information about using TFIM V6.1 for authentication, see “Authentication, mapping, and authorization with TFIM V6.1 and TAM” on page 277.

Windows domain controller and Kerberos Key Distribution Center providers are reached by using Integrated Windows Authentication. For more information, see “Integrated Windows Authentication” on page 260.

Related concepts:

“Identity” on page 251

In IBM Integration Bus, an identity is a security token that uniquely identifies an individual, or that provides a set of assertions that can be validated.

“Security profiles” on page 247

A security profile defines the security operations that are to be performed in a message flow at SecurityPEP nodes and security enabled input and output nodes.

Related tasks:

“Configuring identity authentication and security token validation” on page 312

You can configure a message flow to perform identity authentication or security token validation using Integrated Windows Authentication (IWA), Lightweight Directory Access Protocol (LDAP), a WS-Trust V1.3 compliant security token server (STS) such as Tivoli Federated Identity Manager (TFIM) Version 6.2, or HTTP requests. Support for TFIM V6.1 is also provided, for compatibility with previous versions of IBM Integration Bus.

“Providing credentials in HTTP requests” on page 328

Use a security profile to configure HTTPRequest and SOAPRequest nodes to authenticate with a remote server.

“Creating a security profile” on page 294

You can create a security profile for use with Lightweight Directory Access Protocol (LDAP) or a WS-Trust V1.3 compliant security token server (STS), such as Tivoli Federated Identity Manager (TFIM) V6.2. Support is also provided for TFIM V6.1, for compatibility with previous versions of IBM Integration Bus. You can create the security profile by using either the `mqsicreateconfigurable-service` command or an editor in the IBM Integration Explorer.

“Setting up message flow security” on page 292

Set up security on a message flow to control access based on the identity of a message passing through the message flow.

Related reference:

 MQInput node

Use the MQInput node to receive messages from clients that connect to the broker by using the WebSphere MQ Enterprise Transport, and that use the MQI and AMI application programming interfaces.

 HTTPInput node

Use the HTTPInput node to receive an HTTP message from an HTTP client for processing by a message flow.

 SOAPInput node

Use the SOAPInput node to process client SOAP messages, so that the broker operates as a SOAP Web Services provider.

 SOAPAsyncRequest node

Use the SOAPAsyncRequest node with the SOAPAsyncResponse node to construct a pair of message flows that call a Web service asynchronously.



HTTPRequest node

Use the HTTPRequest node to interact with a web service.



MQOutput node

Use the MQOutput node to send messages to clients that connect to the broker using the WebSphere MQ Enterprise Transport and that use the MQI and AMI application programming interfaces.



SecurityPEP node

Use the SecurityPEP node in a message flow to invoke the message flow security manager at any point in the message flow.

Integrated Windows Authentication:

Integrated Windows Authentication (IWA) refers to a set of authentication protocols, NTLM, Kerberos, and SPNEGO, that are used to provide transport-level security. You can configure IBM Integration Bus to provide an IWA-secured service on a broker running on any operating system, and to consume an IWA-secured service on a broker running on Windows, when you are using the HTTP and SOAP nodes.

IWA provides authentication to users who have an identity in Windows domains or in the Kerberos Key Distribution Center (KDC). IWA includes the protocols NT Lan Manager (NTLM), Kerberos, and Simple and Protected Negotiation (SPNEGO):

NTLM

A family of Microsoft security protocols that are used to secure access to resources within and across Windows domains. NTLM is also known as Windows Challenge/Response.

Kerberos

An authentication protocol (defined by RFC 4120), developed by The Massachusetts Institute of Technology, which allows resources to be secured by using a trusted third party, the Kerberos KDC.

SPNEGO

An open standard (RFC 4559) for negotiating an underlying security mechanism. SPNEGO is used to negotiate the use of either NTLM or Kerberos.

You can use IWA with the HTTPInput and SOAPInput nodes to provide a service. For more information on configuring IBM Integration Bus to provide an IWA-secured service, see “Authenticating incoming requests with IWA on Windows” on page 313 and “Authenticating incoming requests with IWA on Linux or UNIX” on page 315.

To consume an IWA-secured service, use the HTTPRequest and SOAPRequest nodes. For more information on configuring this scenario, see “Providing credentials for outbound requests by using IWA” on page 330. You can consume an IWA-secured service only if IBM Integration Bus is running on Windows.

Related reference:



HTTPInput node

Use the HTTPInput node to receive an HTTP message from an HTTP client for processing by a message flow.

🔧 HTTPRequest node

Use the HTTPRequest node to interact with a web service.

🔧 SOAPInput node

Use the SOAPInput node to process client SOAP messages, so that the broker operates as a SOAP Web Services provider.

🔧 SOAPRequest node

Use the SOAPRequest node to send a SOAP request to the remote Web service.

Related information:

🔗 [NTLM Authentication Protocol specification](#)

📖 [RFC 4120: The Kerberos Network Authentication Service](#)

📖 [RFC 4559: SPNEGO-based Kerberos and NTLM HTTP Authentication in Microsoft Windows](#)

Authorization

Authorization is the process of verifying that an identity token has permission to access a message flow.

If authentication and mapping are configured, they are used to verify the identity before it is authorized.

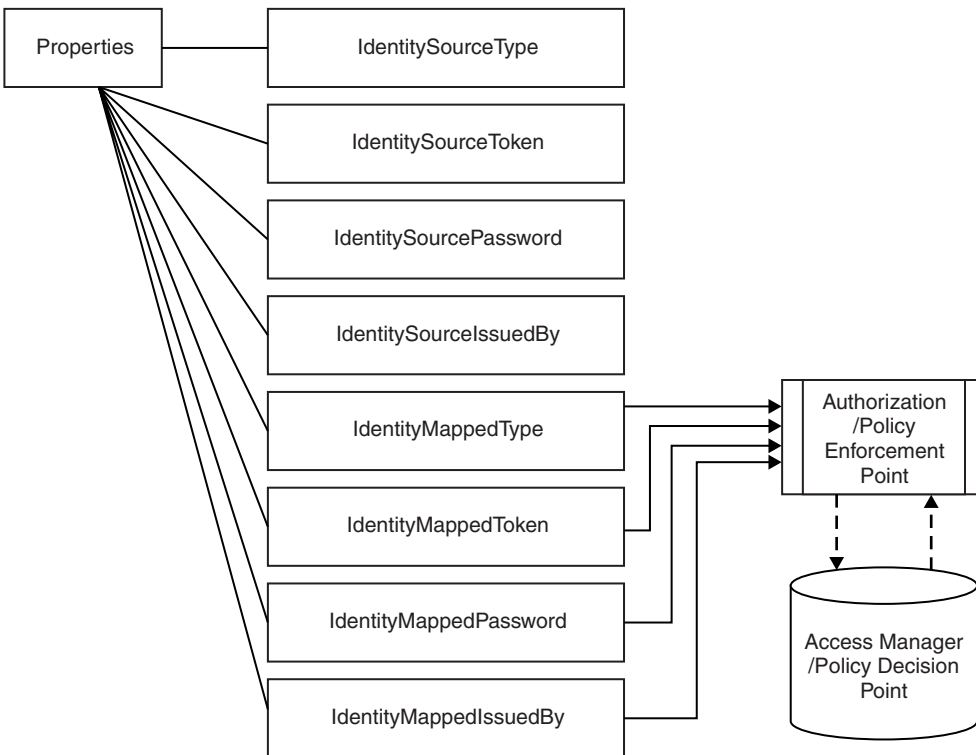


Figure 18. Diagram showing identity authorization.

If a mapped identity exists, authorization is applied to the mapped identity. If a mapped identity does not exist, the source identity is used.

If you specify LDAP as the provider for authorization, the security manager queries the configured LDAP server (which must be LDAP Version 3 compliant), to validate that the identity is a member of the LDAP group that is configured in the security profile.

If you specify *WS-Trust v1.3 STS* as the provider for authorization, the security manager invokes the security token server (STS), such as Tivoli Federated Identity Manager (TFIM) V6.2, to validate that the identity token provided has permission to access the message flow. If you are using TFIM V6.1 rather than TFIM V6.2, you can specify *TFIM* as the provider for authorization.

For more information about using TFIM V6.2 for authorization, see “Authentication, mapping, and authorization with TFIM V6.2 and TAM” on page 280.

For information about using TFIM V6.1 for authorization, see “Authentication, mapping, and authorization with TFIM V6.1 and TAM” on page 277.

Related concepts:

“Identity” on page 251

In IBM Integration Bus, an identity is a security token that uniquely identifies an individual, or that provides a set of assertions that can be validated.

“Authentication and validation” on page 257

Authentication is the process of establishing the identity of a user or system and verifying that the identity is valid. Applying authentication to a SAML security token involves validating the assertions that it carries and confirming that it is being processed within its validity period.

“Identity mapping” on page 263

Identity mapping is the transformation of a security token from one format to another format, or the federation of an identity from one realm to an equivalent identity in another realm.

“Identity and security token propagation” on page 286

Identity and security token propagation enables the identity and security tokens (associated with each message) to be propagated throughout a message flow, and on to target applications through output or request nodes.

“Security profiles” on page 247

A security profile defines the security operations that are to be performed in a message flow at SecurityPEP nodes and security enabled input and output nodes.

“Security exception processing” on page 290

A security exception is raised when a message flow security failure occurs during security processing in an input node or SecurityPEP node.

“Authentication, mapping, and authorization with TFIM V6.2 and TAM” on page 280

You can use IBM Integration Bus, Tivoli Federated Identity Manager (TFIM) V6.2, and Tivoli Access Manager (TAM) to control authentication, mapping, and authorization.

“Authentication, mapping, and authorization with TFIM V6.1 and TAM” on page 277

Use IBM Integration Bus, Tivoli Federated Identity Manager (TFIM) V6.1, and Tivoli Access Manager (TAM) to control authentication, mapping, and authorization.

Related tasks:

“Configuring identity mapping” on page 333

Configure a security token server (STS), such as Tivoli Federated Identity Manager (TFIM) V6.2, to map the incoming security token and, if required, to authenticate and authorize it.

“Creating a security profile” on page 294

You can create a security profile for use with Lightweight Directory Access Protocol (LDAP) or a WS-Trust V1.3 compliant security token server (STS), such as Tivoli Federated Identity Manager (TFIM) V6.2. Support is also provided for TFIM V6.1, for compatibility with previous versions of IBM Integration Bus. You can create the security profile by using either the `mqsicreateconfigurable` service command or an editor in the IBM Integration Explorer.

“Configuring authorization” on page 342

Configure the broker to use an external security provider to authorize an identity in a message flow. You can use either Lightweight Directory Access Protocol (LDAP) or a WS-Trust V1.3 compliant security token server (STS) such as Tivoli Federated Identity Manager (TFIM) V6.2. Support for TFIM V6.1 is also provided for compatibility with previous versions.

“Configuring for identity propagation” on page 389

To enable a message flow to perform identity propagation, the input nodes must extract the identity and the output node must propagate it.

“Setting up message flow security” on page 292

Set up security on a message flow to control access based on the identity of a message passing through the message flow.

Related reference:



SecurityPEP node

Use the SecurityPEP node in a message flow to invoke the message flow security manager at any point in the message flow.

Identity mapping

Identity mapping is the transformation of a security token from one format to another format, or the federation of an identity from one realm to an equivalent identity in another realm.

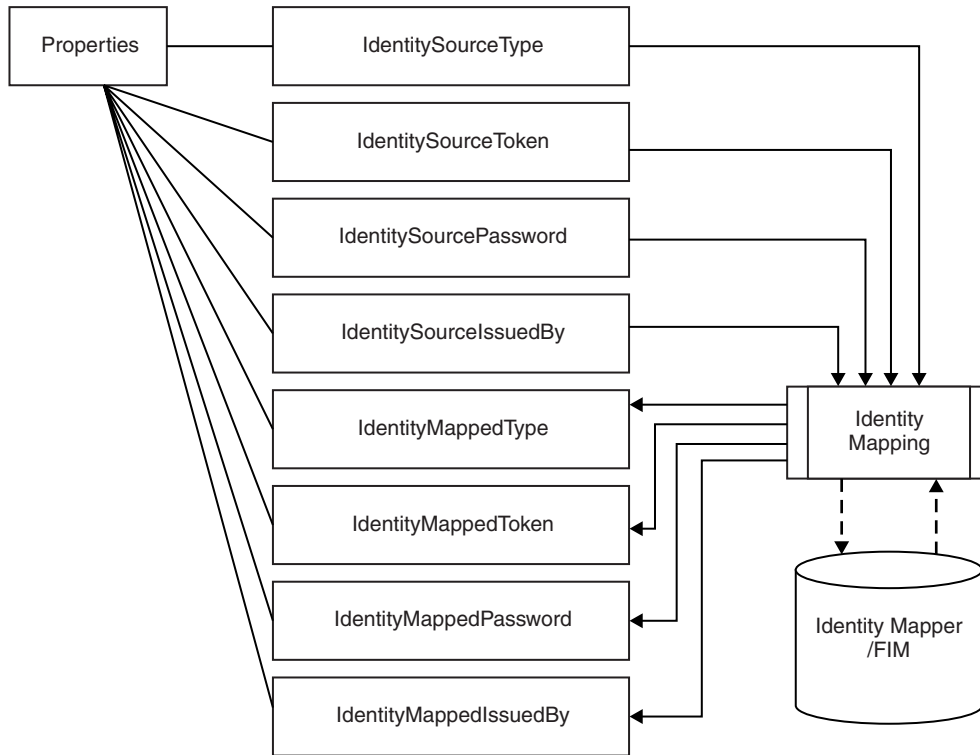


Figure 19. Diagram showing identity mapping.

IBM Integration Bus provides support for identity mapping (also known as identity federation) and token issuance and exchange. Identity mapping is the process of mapping an identity in one realm to another identity in a different realm. For example, you might map *User001* from the eSellers realm to *eSellerUser01* in the eShipping realm. Token issuance and exchange involves the mapping of a token of one type to a token of a different type. For example, an incoming Username and Password token from a client over MQ might be mapped into an equivalent SAML assertion, to be propagated to a Web Services call. Alternatively, you might exchange a SAML 1.1 assertion from a client application for an equivalent SAML 2.0 assertion for an updated backend server.

Mapping using a WS-Trust provider

The IBM Integration Bus security manager supports mapping operations through WS-Trust V1.3 compliant security token servers (STS), such as IBM Tivoli Federated Identity Manager (TFIM) V6.2. Mapping is performed for SecurityPEP nodes and input nodes that have an associated security profile that includes a mapping operation configured with a WS-Trust V1.3 STS.

WS-Trust V1.3 (TFIM V6.2) can also be selected for authentication and authorization in the profile. However, if more than one security operation is associated with the security profile, only one WS-Trust request is issued to the STS. As a result, the STS must be configured to perform all the required operations. For example, if a TFIM V 6.2 server is specified, the TFIM module chain that is invoked must include the appropriate validate, authorize, and issue modules. If you require each operation to be performed through a separate WS-Trust call, you

must use a series of SecurityPEP nodes, each associated with a different security profile that is configured for only one security operation (authentication, authorization, or mapping).

If the security profile specifies only mapping with WS-Trust v1.3 STS, the request is sent with a request type of *Issue*, whereas a mixed set of security operations sends a request type of *Validate*. When mapping is included, the STS must return a token in its response, even if it is the original token; otherwise, an error occurs.

To provide compatibility with previous versions of IBM Integration Bus, support is also provided for TFIM V6.1.

In the broker, identity mapping is performed at the input node or SecurityPEP node, after authentication but before authorization. The source identity is passed to an identity mapper for processing. If both mapping and authorization are configured, the authorization operation uses the mapped output token rather than the source token, which means that the authorization is performed on the federated identity.

Mapping is not performed in output nodes, even if the node has been configured with a security profile.

IBM Integration Bus supports mapping between any type of security token that is supported by the configured security provider. For more information about the support provided, see “Identity” on page 251.

When mapping from an X.509 certificate, TFIM can validate the certificate but cannot verify the identity of the original sender. However, if it is required, this verification integrity check can be performed by the SOAPInput node. For more information, see WS-Security mechanisms.

For more information about using TFIM V6.2 for mapping, see “Authentication, mapping, and authorization with TFIM V6.2 and TAM” on page 280.

For information about using TFIM V6.1, see “Authentication, mapping, and authorization with TFIM V6.1 and TAM” on page 277.

User-defined mapping

When you develop a message flow, you can implement a custom token mapping to be used for identity propagation. For example, you can implement a custom token mapping using a compute node (which can be a Compute, JavaCompute, or PHPCompute node) following the input node. In the compute node, you can read the source identity values from the Properties folder, process them, then write the new identity values to the mapped identity fields. If there is no identity provided in the message, you can still use a compute node to insert some identity credentials into the mapped identity fields. The mapped identity fields are then used in place of the source identity fields by subsequent nodes. Any security operations that are configured on the input node are performed using the source identity, before you can create a new identity in the mapped identity fields (by using the compute node). However, you can include a SecurityPEP node after your compute node has created a new mapped identity.

Related concepts:

“Identity” on page 251

In IBM Integration Bus, an identity is a security token that uniquely identifies an

individual, or that provides a set of assertions that can be validated.

“Authentication and validation” on page 257

Authentication is the process of establishing the identity of a user or system and verifying that the identity is valid. Applying authentication to a SAML security token involves validating the assertions that it carries and confirming that it is being processed within its validity period.

“Authorization” on page 261

Authorization is the process of verifying that an identity token has permission to access a message flow.

“Identity and security token propagation” on page 286

Identity and security token propagation enables the identity and security tokens (associated with each message) to be propagated throughout a message flow, and on to target applications through output or request nodes.

“Security profiles” on page 247

A security profile defines the security operations that are to be performed in a message flow at SecurityPEP nodes and security enabled input and output nodes.

“Authentication, mapping, and authorization with TFIM V6.1 and TAM” on page 277

Use IBM Integration Bus, Tivoli Federated Identity Manager (TFIM) V6.1, and Tivoli Access Manager (TAM) to control authentication, mapping, and authorization.

“Authentication, mapping, and authorization with TFIM V6.2 and TAM” on page 280

You can use IBM Integration Bus, Tivoli Federated Identity Manager (TFIM) V6.2, and Tivoli Access Manager (TAM) to control authentication, mapping, and authorization.

Related tasks:

“Configuring identity mapping” on page 333

Configure a security token server (STS), such as Tivoli Federated Identity Manager (TFIM) V6.2, to map the incoming security token and, if required, to authenticate and authorize it.

“Setting up message flow security” on page 292

Set up security on a message flow to control access based on the identity of a message passing through the message flow.

Related reference:



HTTPRequest node

Use the HTTPRequest node to interact with a web service.



MQOutput node

Use the MQOutput node to send messages to clients that connect to the broker using the WebSphere MQ Enterprise Transport and that use the MQI and AMI application programming interfaces.



SecurityPEP node

Use the SecurityPEP node in a message flow to invoke the message flow security manager at any point in the message flow.

Invoking message flow security using a security enabled input node

You can invoke the message flow security manager by configuring a security enabled input node.

The following diagram shows an example message flow and gives an overview of the sequence of events that occur when an input message is received by a security enabled input node in the message flow:

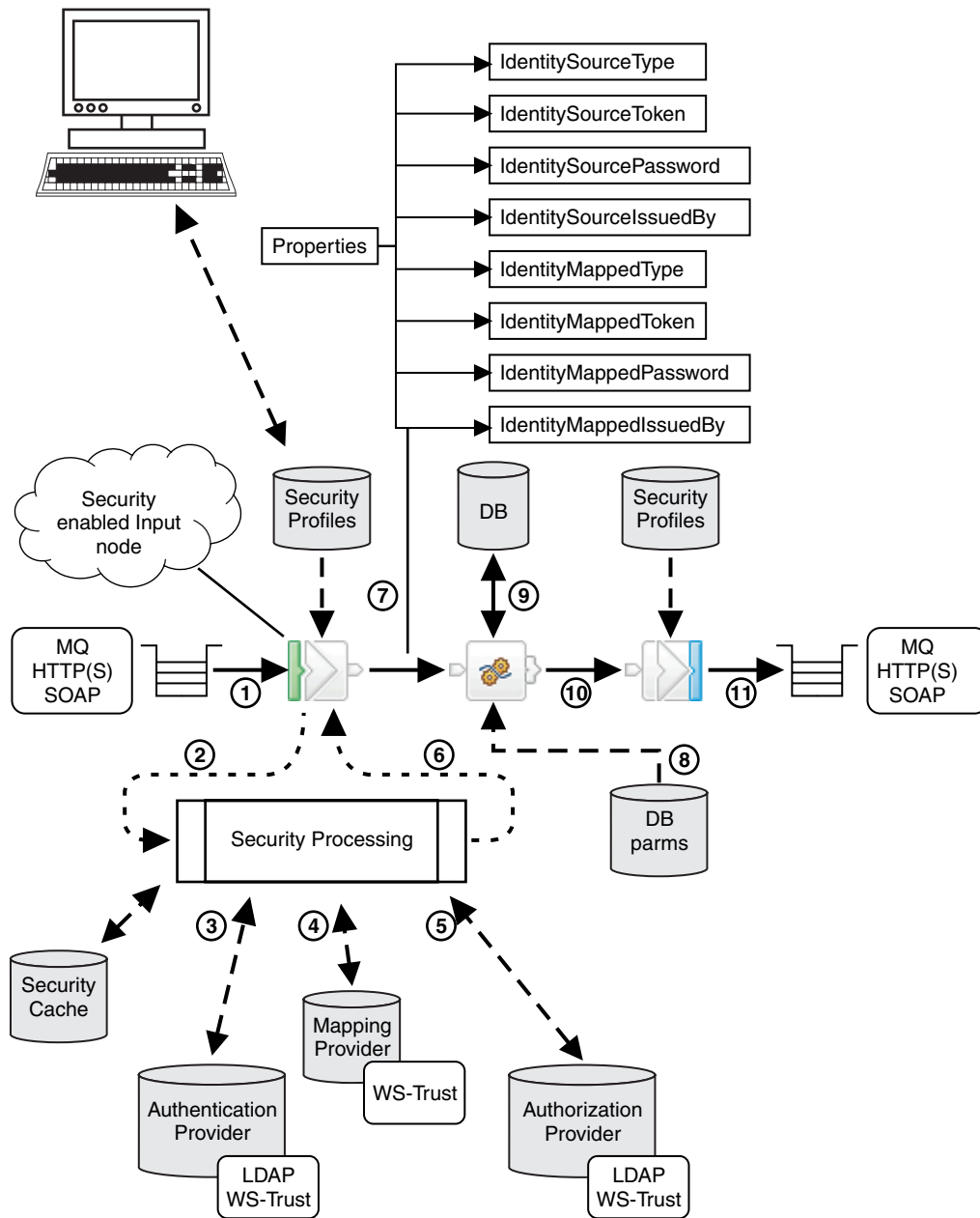


Figure 20. Diagram showing the sequence of events that occur when a message arrives at a security enabled input node in a message flow.

The following steps explain the sequence of events that occur when a message arrives at a security enabled input node in the message flow. The numbers correspond to those in the preceding diagram:

1. When a message arrives at a security enabled input node (MQ, HTTP, SCA, or SOAP), the presence of a security profile associated with the node indicates whether message flow security is configured. SOAP nodes can implement some WS-Security capabilities without using the broker's security manager; for

more information, see *Implementing WS-Security*. The broker's security manager is called to read the profile, which specifies the combination of propagation, authentication, authorization, and mapping to be performed with the identity of the message. It also specifies the external security provider (also known as the Policy Decision Point or PDP) to be used.

You can create security profiles by using either the `mqsicreateconfigurable-service` command or an editor in the IBM Integration Explorer. You then use the Broker Archive editor to configure the security profile on either an individual node or the whole message flow. If you associate the security profile with the message flow, the security profile applies to all security enabled input and output nodes and SecurityPEP nodes in the message flow. However, a security profile that is associated with an individual node takes precedence over a security profile that is associated with the message flow. A predefined security profile, called *Default propagation*, is provided for setting identity propagation. To explicitly set no security on a node, set the security profile to *No security*.

2. If a security profile is associated with the node or message flow, the security enabled input node extracts the identity information from the input message (based on the configuration on the node's **Security** properties page) and sets the Source Identity elements in the Properties folder. If the security tokens cannot be successfully extracted, a security exception is raised.

If you require a SOAPInput node to use the identity in the WS-Security header (rather than an underlying transport identity), you must also define and specify an appropriate policy set and bindings for the relevant token profile. For more information, see *Policy sets*.

For MQ, HTTP, or SCA input nodes, the **Security** properties page is used to configure the extraction of the identity. This defaults to *Transport Default*. For example, an HTTPInput node extracts a username and password from the HTTP BasicAuth header. The **Security** properties page allows the Identity token type and its location to be explicitly configured to control the extraction. This source identity information can be in a message header, the message body, or both.

For information about the tokens that are supported by each node, see "Identity" on page 251.

3. If authentication is specified in the security profile, the security manager calls the configured security provider to authenticate the identity. A failure results in a security exception being returned to the node. The security providers that are supported by IBM Integration for authentication are LDAP, WS-Trust v1.3 compliant security token servers (such as TFIM V6.2) and TFIM V6.1.

A security cache is provided for the authentication result, which enables subsequent messages (with the same credentials) arriving at the message flow to be completed with the cached result, provided that it has not expired.

4. If identity mapping is specified in the security profile, the security manager calls the configured security provider to map the identity to an alternative identity. A failure results in a security exception being returned to the node. Otherwise, the mapped identity information is set in the Mapped Identity elements in the Properties folder.

The security providers that are supported by IBM Integration for identity mapping are WS-Trust V1.3 compliant security token servers (such as TFIM V6.2) and TFIM V6.1.

A security cache is provided for the result of the identity mapping.

Alternatively, you can use a SecurityPEP node at any point in the message flow to map the identity that was authenticated at the security enabled input node. For more information, see “Invoking message flow security using a SecurityPEP node” on page 272.

5. If authorization is specified in the security profile, the security manager calls the configured security provider to authorize that the identity (either mapped or source) has access to this message flow. A failure results in a security exception being returned to the node.

The security providers that are supported by IBM Integration for authorization are LDAP, WS-Trust V1.3 compliant security token servers (such as TFIM V6.2) and TFIM V6.1.

A security cache is provided for the authorization result.

Alternatively, you can use a SecurityPEP node at any point in the message flow to authorize the identity that was authenticated at the security enabled input node. For more information, see “Invoking message flow security using a SecurityPEP node” on page 272.

6. When all security processing is complete, or when a security exception is raised by the message flow security manager, control returns to the input node.

When a security exception is returned to the security enabled input node, it performs the appropriate transport handling and ends the message flow transaction. For example, an HTTPInput node returns an HTTP header with a 401 HTTP response code, without propagating to an output terminal. A SOAPInput node returns a SOAP Fault, reporting the security exception. Alternatively, if the Treat security exceptions as normal property is set on the security enabled input node, a security exception is propagated to the node's Failure terminal. The security enabled input node propagates to its Out terminal only if all the configured operations in the associated security profile complete successfully.

7. The message, including the Properties folder and its source and mapped identity information, is propagated down the message flow.
8. At subsequent nodes in the message flow an identity might be required to access a resource such as a database. The identity used to access such a resource is a proxy identity, either the broker's identity or an identity configured for the specific resource by using the `mqsisetdbparms` command.
9. When you are developing a message flow you can use the identity fields in the Properties folder for application processing (for example, identity-based routing or content building based on identity). Also, as an alternative to invoking mapping through a WS-Trust V1.3 enabled STS (such as TFIM V6.2) or TFIM V6.1, you can set the mapped identity fields in a compute node, such as a Compute, JavaCompute, PHPCompute, or Mapping node.
10. When the message reaches a security enabled output or request node (MQOutput, HTTPRequest, SOAPRequest, or SOAPAsyncRequest), a security profile (with propagation enabled) associated with the node indicates that the current identity token is to be propagated when the message is sent.

If the security profile indicates that propagation is required, the mapped identity is used. If the mapped identity is not set, or if it has a token type that is not supported by the node, the source identity is used. If no identity is set, or if neither the mapped nor source identity has a token type that is supported by the node, a security exception is returned to the node.

SOAP nodes also require the appropriate policy set and bindings for the token profile to be associated with the node.

If you want to include a security token in the message that is issued at an output node, and if the output node is not able to propagate that type of token, you can use a compute node (before the output node) to put the token from the properties tree into the relevant message location.

For information about the tokens that are supported by each node, see “Identity and security token propagation” on page 286.

11. The propagated identity is included in the appropriate message header when it is sent.

Related concepts:

“Message flow security overview” on page 243

IBM Integration Bus provides a security manager, which enables you to control access to individual messages in a message flow, using the identity of the message.

“Invoking message flow security using a SecurityPEP node” on page 272

You can invoke the message flow security manager at any point in a message flow, between an input node and an output or request node, by using a SecurityPEP node.

Related tasks:

“Setting up message flow security” on page 292

Set up security on a message flow to control access based on the identity of a message passing through the message flow.

“Configuring the extraction of an identity or security token” on page 308

You can configure the SecurityPEP node or security enabled input nodes to extract the identity or security token from a message and store it in the properties tree identity fields, enabling it to be processed throughout the message flow and propagated at output or request nodes.

“Creating a security profile” on page 294

You can create a security profile for use with Lightweight Directory Access Protocol (LDAP) or a WS-Trust V1.3 compliant security token server (STS), such as Tivoli Federated Identity Manager (TFIM) V6.2. Support is also provided for TFIM V6.1, for compatibility with previous versions of IBM Integration Bus. You can create the security profile by using either the `mqsicreateconfigurable` service command or an editor in the IBM Integration Explorer.

“Configuring identity authentication and security token validation” on page 312

You can configure a message flow to perform identity authentication or security token validation using Integrated Windows Authentication (IWA), Lightweight Directory Access Protocol (LDAP), a WS-Trust V1.3 compliant security token sever (STS) such as Tivoli Federated Identity Manager (TFIM) Version 6.2, or HTTP requests. Support for TFIM V6.1 is also provided, for compatibility with previous versions of IBM Integration Bus.

“Configuring identity mapping” on page 333

Configure a security token server (STS), such as Tivoli Federated Identity Manager (TFIM) V6.2, to map the incoming security token and, if required, to authenticate and authorize it.

“Configuring authorization” on page 342

Configure the broker to use an external security provider to authorize an identity in a message flow. You can use either Lightweight Directory Access Protocol (LDAP) or a WS-Trust V1.3 compliant security token server (STS) such as Tivoli Federated Identity Manager (TFIM) V6.2. Support for TFIM V6.1 is also provided for compatibility with previous versions.

“Configuring for identity propagation” on page 389

To enable a message flow to perform identity propagation, the input nodes must extract the identity and the output node must propagate it.

Related reference:

mqsicreateconfigurable service command

Use the **mqsicreateconfigurable** service command to create an object name for a broker external resource.

mqsideleteconfigurable service command

Use the **mqsideleteconfigurable** service command to delete a configurable service, such as a JMS provider, JDBC provider, or FTP server, that you have created by using the **mqsicreateconfigurable** service command.

mqsichangeproperties command

Use the **mqsichangeproperties** command to modify broker properties and properties of broker resources.

mqsireportproperties command

Use the **mqsireportproperties** command to display properties that relate to a broker, an integration server, or a configurable service.

mqsireloadsecurity command

Use the **mqsireloadsecurity** command to force the immediate expiry of some or all the entries in the security cache.

Parameter values for the securitycache component

Select the objects and properties associated with the securitycache component that you want to change.

MQInput node

Use the MQInput node to receive messages from clients that connect to the broker by using the WebSphere MQ Enterprise Transport, and that use the MQI and AMI application programming interfaces.

HTTPInput node

Use the HTTPInput node to receive an HTTP message from an HTTP client for processing by a message flow.

SOAPInput node

Use the SOAPInput node to process client SOAP messages, so that the broker operates as a SOAP Web Services provider.

SOAPAsyncRequest node

Use the SOAPAsyncRequest node with the SOAPAsyncResponse node to construct a pair of message flows that call a Web service asynchronously.

SCAInput node

Use the SCAInput node with the SCAREply node to process messages from WebSphere Process Server.

SecurityPEP node

Use the SecurityPEP node in a message flow to invoke the message flow security manager at any point in the message flow.

HTTPRequest node

Use the HTTPRequest node to interact with a web service.

MQOutput node

Use the MQOutput node to send messages to clients that connect to the broker using the WebSphere MQ Enterprise Transport and that use the MQI and AMI application programming interfaces.

Invoking message flow security using a SecurityPEP node

You can invoke the message flow security manager at any point in a message flow, between an input node and an output or request node, by using a SecurityPEP node.

The following diagram shows an example message flow and gives an overview of the sequence of events that occur when an input message is received by an input node that is not security enabled (or that has no associated security profile) and is later processed by a SecurityPEP node in the message flow:

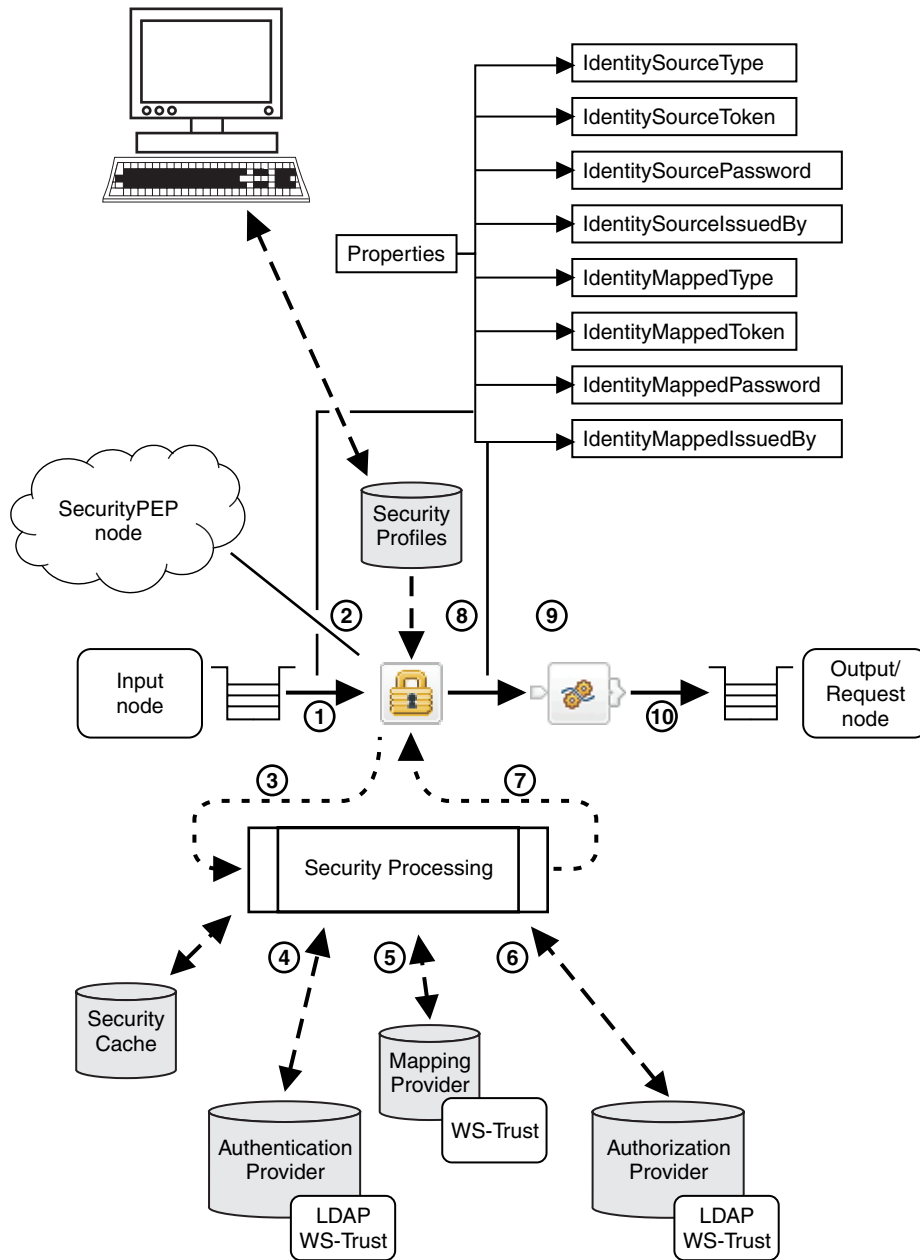


Figure 21. Diagram showing the sequence of events that occur when a message arrives at a SecurityPEP node in a message flow.

The following steps explain the sequence of events that occur when a message arrives at an input node that is not security enabled (or that has no associated security profile). The numbers correspond to those in the preceding diagram:

1. You can use a SecurityPEP node at any point in a message flow between an input and an output or request node. The SecurityPEP node enables security to be applied in a message flow in the following situations:
 - When the message flow input node is not security enabled (for example, FileInput, TCPIPClientInput, SAPInput, and JMSInput nodes).
 - When the message flow input node is security enabled and might be configured to perform authentication operations, but the message flow is

required to perform some routing or filtering before the business function being invoked is known; as a result, authorization needs to be performed later in the message flow logic.

- When the message flow includes multiple output or request nodes, which require a specific identity mapping to be performed before each node, to obtain the appropriate security tokens for propagation.

The message tree that is propagated into the SecurityPEP node includes the properties tree identity fields. These fields are empty, unless a security enabled input node (or a prior SecurityPEP node) has already extracted identity tokens, and possibly performed some security operations.

2. When a message arrives at a SecurityPEP, the presence of a security profile associated with the node indicates whether message flow security is configured. The broker's security manager is called to read the profile, which specifies the combination of propagation, authentication, authorization, and mapping to be performed with the identity of the message. It also specifies the external security provider (also known as the Policy Decision Point or PDP) to be used.

You can create security profiles by using either the **mqsicreateconfigurable** command or an editor in the IBM Integration Explorer. You then use the Broker Archive editor to configure the security profile on either an individual node or the whole message flow. If you associate the security profile with the message flow, the security profile applies to all security enabled input and output and SecurityPEP nodes in the message flow. However, a security profile that is associated with an individual node takes precedence over a security profile that is associated with the message flow. Predefined security profiles are provided for setting identity propagation and for explicitly setting no security on a node.

3. If a security profile is associated with the SecurityPEP node or message flow, the node extracts the identity information from the message tree based on the node configuration and sets the Source Identity elements in the Properties folder. If the node sets a token type of *Current token*, the existing identity tokens in the Mapped Identity properties fields are used (if they exist); if there are no identity tokens in the Mapped Identity properties fields, the tokens in the Source Identity properties fields are used. If the security tokens cannot be successfully extracted, a security exception is raised and propagated to the failure terminal (if wired).
4. If authentication is specified in the security profile, the security manager calls the configured security provider to authenticate the identity. A failure results in a security exception being returned to the node. The security providers that are supported by Message Broker for authentication are LDAP, WS-Trust v1.3 compliant security token servers (such as TFIM V6.2), and TFIM V6.1.

A security cache is provided for the authentication result, which enables subsequent messages (with the same credentials) arriving at the message flow to be completed with the cached result, provided that it has not expired.

5. If identity mapping is specified in the security profile, the security manager calls the configured security provider to map the identity to an alternative identity. A failure results in a security exception being returned to the node. Otherwise, the mapped identity information is set in the Mapped Identity elements in the Properties folder.

The security providers that are supported by Message Broker for identity mapping are WS-Trust V1.3 compliant security token servers (such as TFIM V6.2) and TFIM V6.1.

A security cache is provided for the result of the identity mapping.

6. If authorization is specified in the security profile, the security manager calls the configured security provider to authorize that the identity (either mapped or source) has access to this message flow. A failure results in a security exception being returned to the node.

The security providers that are supported by Message Broker for authorization are LDAP, WS-Trust V1.3 compliant security token servers (such as TFIM V6.2) and TFIM V6.1.

A security cache is provided for the authorization result.

7. When all security processing is complete, or when a security exception is raised by the message flow security manager, control returns to the SecurityPEP node.

When a security exception is returned to the SecurityPEP node, the exception is either propagated to the failure terminal if it is connected, or returned to the preceding node as a recoverable exception. The SecurityPEP node propagates to its Out terminal only if all the configured operations in the associated security profile complete successfully.

8. The message, including the populated Properties folder and its source and mapped identity information, is propagated down the message flow.
9. When you are developing a message flow, you can use the identity fields in the Properties folder for application processing (for example, identity-based routing or content building based on identity). If the identity is to be propagated in an outbound message from an output or request node that does not support propagation of the token, you can use a compute node (including a Compute, JavaCompute, PHPCompute, or Mapping node), to move the identity token into the required transport header or message body location.
10. When the message reaches an output node, a security profile associated with the node can indicate whether an identity is to be taken from the Properties folder and propagated when the message is sent. Only specific transport nodes can propagate tokens that are the default for the transport; any other token type must be handled by a compute node, as described above.

Related concepts:

“Message flow security overview” on page 243

IBM Integration Bus provides a security manager, which enables you to control access to individual messages in a message flow, using the identity of the message.

“Invoking message flow security using a security enabled input node” on page 266

You can invoke the message flow security manager by configuring a security enabled input node.

Related tasks:

“Setting up message flow security” on page 292

Set up security on a message flow to control access based on the identity of a message passing through the message flow.

“Configuring the extraction of an identity or security token” on page 308

You can configure the SecurityPEP node or security enabled input nodes to extract the identity or security token from a message and store it in the properties tree identity fields, enabling it to be processed throughout the message flow and propagated at output or request nodes.

“Creating a security profile” on page 294

You can create a security profile for use with Lightweight Directory Access Protocol (LDAP) or a WS-Trust V1.3 compliant security token server (STS), such as Tivoli Federated Identity Manager (TFIM) V6.2. Support is also provided for TFIM V6.1, for compatibility with previous versions of IBM Integration Bus. You can create the security profile by using either the **mqscreateconfigurable** command or an editor in the IBM Integration Explorer.

“Configuring identity authentication and security token validation” on page 312

You can configure a message flow to perform identity authentication or security token validation using Integrated Windows Authentication (IWA), Lightweight Directory Access Protocol (LDAP), a WS-Trust V1.3 compliant security token sever (STS) such as Tivoli Federated Identity Manager (TFIM) Version 6.2, or HTTP requests. Support for TFIM V6.1 is also provided, for compatibility with previous versions of IBM Integration Bus.

“Configuring identity mapping” on page 333

Configure a security token server (STS), such as Tivoli Federated Identity Manager (TFIM) V6.2, to map the incoming security token and, if required, to authenticate and authorize it.

“Configuring authorization” on page 342

Configure the broker to use an external security provider to authorize an identity in a message flow. You can use either Lightweight Directory Access Protocol (LDAP) or a WS-Trust V1.3 compliant security token server (STS) such as Tivoli Federated Identity Manager (TFIM) V6.2. Support for TFIM V6.1 is also provided for compatibility with previous versions.

“Configuring for identity propagation” on page 389

To enable a message flow to perform identity propagation, the input nodes must extract the identity and the output node must propagate it.

Related reference:

 **mqscreateconfigurable** command

Use the **mqscreateconfigurable** command to create an object name for a broker external resource.

 **mqsdeleteconfigurable** command

Use the **mqsdeleteconfigurable** command to delete a configurable service, such as a JMS provider, JDBC provider, or FTP server, that you have created by using the **mqscreateconfigurable** command.

 **mqschangeproperties** command


Use the **mqschangeproperties** command to modify broker properties and properties of broker resources.

 **mqsreportproperties** command

Use the **mqsreportproperties** command to display properties that relate to a broker, an integration server, or a configurable service.

 **mqsreloadsecurity** command

Use the **mqsreloadsecurity** command to force the immediate expiry of some or all the entries in the security cache.

 Parameter values for the securitycache component

Select the objects and properties associated with the securitycache component that you want to change.

MQInput node

Use the MQInput node to receive messages from clients that connect to the broker by using the WebSphere MQ Enterprise Transport, and that use the MQI and AMI application programming interfaces.

HTTPInput node

Use the HTTPInput node to receive an HTTP message from an HTTP client for processing by a message flow.

SOAPInput node

Use the SOAPInput node to process client SOAP messages, so that the broker operates as a SOAP Web Services provider.

SOAPAsyncRequest node

Use the SOAPAsyncRequest node with the SOAPAsyncResponse node to construct a pair of message flows that call a Web service asynchronously.

SCAInput node

Use the SCAInput node with the SCAREply node to process messages from WebSphere Process Server.

SecurityPEP node

Use the SecurityPEP node in a message flow to invoke the message flow security manager at any point in the message flow.

HTTPRequest node

Use the HTTPRequest node to interact with a web service.

MQOutput node

Use the MQOutput node to send messages to clients that connect to the broker using the WebSphere MQ Enterprise Transport and that use the MQI and AMI application programming interfaces.

Authentication, mapping, and authorization with TFIM V6.1 and TAM

Use IBM Integration Bus, Tivoli Federated Identity Manager (TFIM) V6.1, and Tivoli Access Manager (TAM) to control authentication, mapping, and authorization.

Note: Support for TFIM V6.1 is included for compatibility with previous versions of IBM Integration Bus. If possible, upgrade to TFIM V6.2 and refer to the information in “Authentication, mapping, and authorization with TFIM V6.2 and TAM” on page 280.

IBM Integration Bus makes a single TFIM WS-Trust call for an input node that is configured with a TFIM security profile, which means that a single module chain must be configured to perform all the required authentication, mapping, and authorization operations.

The following diagram shows the configuration of IBM Integration Bus, TFIM, and TAM to enable authentication, mapping, and authorization of an identity in a message flow:

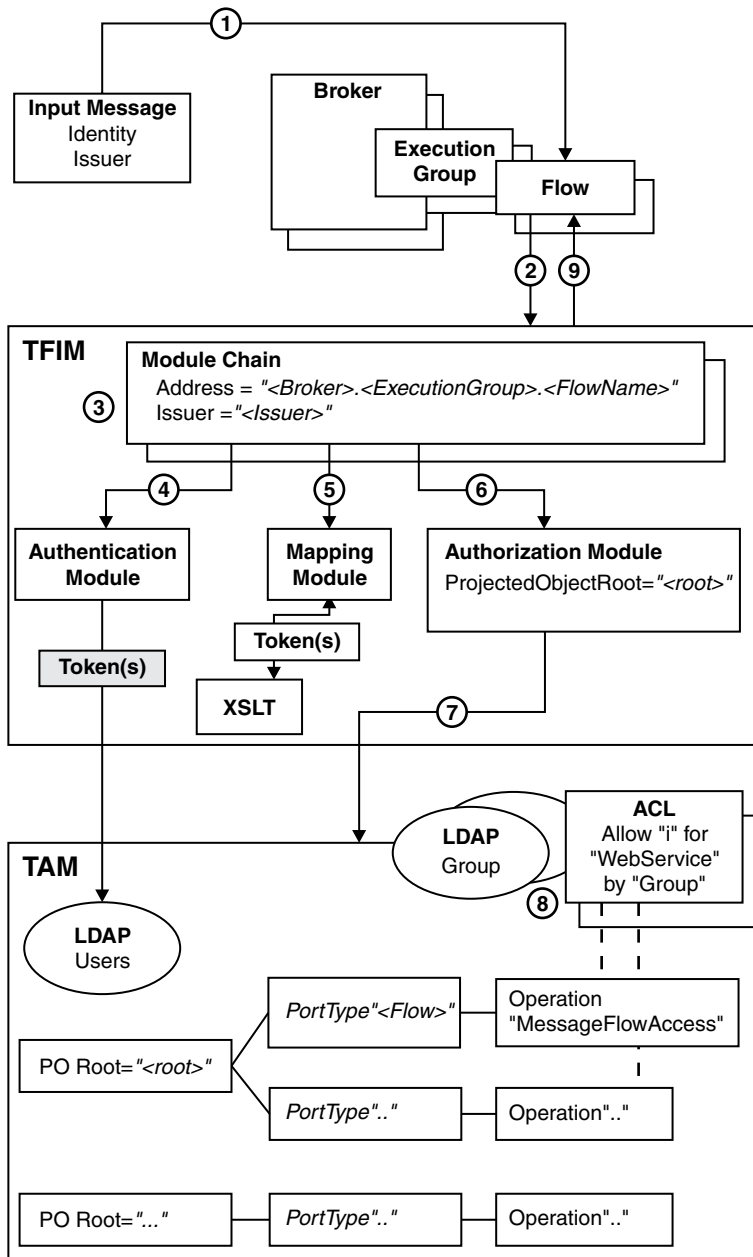


Figure 22. The image is described in the text.

The numbers in the preceding diagram correspond to the following sequence of events:

1. A message enters a message flow.
2. A WS-Trust request is issued by the broker, with these properties:
 - RequestType = Validate
 - Identity = Token(s) from input message
 - Issuer = Issuer from input message
 - AppliesTo Address = "Broker.IntegrationServer.FlowName"
 - PortType = "FlowName"
 - Operation = "MessageFlowAccess"

3. TFIM selects a module chain to process the WS-Trust request, based on the AppliesTo Address and Issuer properties of the request.
4. A module chain can perform authentication if it includes a module (such as a UsernameTokenSTSMModule or X509STSMModule) in validate mode.
5. A module chain can perform mapping by using an XSLTransformationModule in mapping mode to manipulate the identity information.
6. A module chain can perform authorization by using an AuthorizationSTSMModule in other mode. The module chain must be configured with a *Protected Object Root* value.
7. The AuthorizationSTSMModule performs the authorization check by making a request to TAM with these properties:
 - Action = "i" (invoke)
 - Action Group = "WebService"
 - Protected Object = "*ProtectedObjectRoot.FlowName.MessageFlowAccess*" where "i" and "WebService" are default values used by an AuthorizationSTSMModule; and *FlowName* and *MessageFlowAccess* are the WS-Trust request PortType and Operation values.
8. TAM processes the authorization request by:
 - a. Finding the Access Control Lists (ACLs) associated with protected object "*<ProtectedObjectRoot>.<FlowName>.MessageFlowAccess*".
 - b. Checking whether or not the ACLs grant action "i" on action group "WebService" to the user (with the user either named directly, or by membership of a named group).
9. The WS-Trust reply is returned to the broker. If this action is the result of a mapping request, the WS-Trust reply contains the mapped identity token.

For further information about how to configure TFIM and TAM, see IBM Security Systems information centers.

Related concepts:

"Identity" on page 251

In IBM Integration Bus, an identity is a security token that uniquely identifies an individual, or that provides a set of assertions that can be validated.

"Authentication and validation" on page 257

Authentication is the process of establishing the identity of a user or system and verifying that the identity is valid. Applying authentication to a SAML security token involves validating the assertions that it carries and confirming that it is being processed within its validity period.

"Authorization" on page 261

Authorization is the process of verifying that an identity token has permission to access a message flow.

"Identity mapping" on page 263

Identity mapping is the transformation of a security token from one format to another format, or the federation of an identity from one realm to an equivalent identity in another realm.

"Security profiles" on page 247

A security profile defines the security operations that are to be performed in a message flow at SecurityPEP nodes and security enabled input and output nodes.

Related tasks:

“Configuring identity authentication and security token validation” on page 312
You can configure a message flow to perform identity authentication or security token validation using Integrated Windows Authentication (IWA), Lightweight Directory Access Protocol (LDAP), a WS-Trust V1.3 compliant security token server (STS) such as Tivoli Federated Identity Manager (TFIM) Version 6.2, or HTTP requests. Support for TFIM V6.1 is also provided, for compatibility with previous versions of IBM Integration Bus.

“Configuring identity mapping” on page 333
Configure a security token server (STS), such as Tivoli Federated Identity Manager (TFIM) V6.2, to map the incoming security token and, if required, to authenticate and authorize it.

“Creating a security profile” on page 294
You can create a security profile for use with Lightweight Directory Access Protocol (LDAP) or a WS-Trust V1.3 compliant security token server (STS), such as Tivoli Federated Identity Manager (TFIM) V6.2. Support is also provided for TFIM V6.1, for compatibility with previous versions of IBM Integration Bus. You can create the security profile by using either the `mqsicreateconfigurableservice` command or an editor in the IBM Integration Explorer.

“Configuring authorization” on page 342
Configure the broker to use an external security provider to authorize an identity in a message flow. You can use either Lightweight Directory Access Protocol (LDAP) or a WS-Trust V1.3 compliant security token server (STS) such as Tivoli Federated Identity Manager (TFIM) V6.2. Support for TFIM V6.1 is also provided for compatibility with previous versions.

“Configuring authorization with TFIM V6.1” on page 355
You can configure a message flow to perform authorization on an identity by using Tivoli Federated Identity Manager (TFIM) V6.1.

“Setting up message flow security” on page 292
Set up security on a message flow to control access based on the identity of a message passing through the message flow.

Related reference:



MQInput node

Use the MQInput node to receive messages from clients that connect to the broker by using the WebSphere MQ Enterprise Transport, and that use the MQI and AMI application programming interfaces.



HTTPInput node

Use the HTTPInput node to receive an HTTP message from an HTTP client for processing by a message flow.



SOAPInput node

Use the SOAPInput node to process client SOAP messages, so that the broker operates as a SOAP Web Services provider.

Authentication, mapping, and authorization with TFIM V6.2 and TAM

You can use IBM Integration Bus, Tivoli Federated Identity Manager (TFIM) V6.2, and Tivoli Access Manager (TAM) to control authentication, mapping, and authorization.

IBM Integration Bus makes a single TFIM WS-Trust call for an input node or SecurityPEP node that is configured with a WS-Trust V1.3 STS security profile. As a result, a single module chain must be configured to perform all the required authentication, mapping, and authorization operations.

When you use a WS-Trust v1.3 STS for authentication, authorization, or mapping, a request is made to the trust service with the following parameters, which control the STS processing. If you are using TFIM V6.2, these parameters are used in the selection of the TFIM module chain:

Parameter	Value
RequestType	<p>The type of request issued to the trust service. Valid values are:</p> <p>Issue This value can be specified when mapping is the only operation that is set to WS-Trust V1.3 STS in the security profile. It is not valid if WS-Trust V1.3 STS is specified for authentication or authorization.</p> <p>The namespace qualified value is <code>http://docs.oasis-open.org/ws-sx/ws-trust/200512/Issue</code>, which shows in TFIM V6.2 as <i>Issue Oasis URI</i>.</p> <p>Validate This value must be set when the security profile also includes authentication or authorization (in addition to mapping) for the same WS-Trust V1.3 STS provider.</p> <p>The namespace qualified value is <code>http://docs.oasis-open.org/ws-sx/ws-trust/200512/Validate</code>, which shows in TFIM V6.2 as <i>Validate Oasis URI</i>.</p>
Issuer	<p>This value is determined by the effective setting of the IssuedBy property on the Basic tab of the SecurityPEP node or the Security tab of the input node.</p>
AppliesTo	<p>This value is determined by the type of node:</p> <p>MQInput or SCAInput node with MQ binding: The WebSphere MQ IRI of the node's input queue; for example: <code>wmq://msg/queue/queue_name@queue_manager_name</code></p> <p>HTTPInput, SOAPInput, or SOAPAsyncResponse node with HTTP binding: The endpoint URL; for example: <code>http://myflow/myInputNodePath</code></p> <p>SecurityPEP node with a default (blank) WS-Trust AppliesTo address: The URN for the message flow that contains the node; for example: <code>urn:/broker_name.execution_group_name.flow_name</code></p> <p>SecurityPEP node with WS-Trust AppliesTo address set on the Advanced tab of the node: The URI value configured in the property. This value is typically the URL of the target service that is used when you invoke a mapping operation to obtain the required token for the following request node; for example: <code>http://remotehost.ibm.com:9080/targetservice</code></p> <p>You can also set the AppliesTo service name and AppliesTo port type properties on the Advanced tab of the node. The WS-Trust request includes these optional elements only when they are configured. These values are typically valid QNames; for example: <code>http://myservice.mycom.com:myservicename</code></p> <p>When you set these properties in the SecurityPEP node, you must configure them in the TFIM module chain:</p> <ul style="list-style-type: none"> • In the service name and port type TFIM properties, the information to the left of the colon must match the namespace URI of the WS-Addressing namespace that is used for the PortType and ServiceName elements in the WS-Trust request set by the broker, which is: <code>http://www.w3.org/2005/08/addressing</code> • The information to the right of the colon in the service name and port type TFIM properties must match the value configured on the SecurityPEP node. You can also configure a regular expression in TFIM to specify a match.

This section describes an authorization configuration that you can use to perform the authorization operation with TFIM V6.2 and TAM.

In the security profile, set the TFIM V6.2 endpoint for the authorization operation. When you create a module chain to be used by a security enabled input node or SecurityPEP node, and resolved by *AppliesTo* information, you must include the TFIM TAMAuthorizationSTModule to invoke TAM authorization.

The TAMAuthorizationSTModule requires the following TFIM STS universal user context attributes:

PrincipalName

The username to be authorized. This username must exist in your TAM user repository.

ObjectName

The TAM object name of the resource on which an authorization check is to be made. Typically this is derived from the *AppliesTo* information that is passed by the message flow security manager from the security enabled input node or SecurityPEP node.

Action

The TAM action to be authorized; for example, *x* (eXecute).

The TAM Access Control List (ACL), which determines the authorization decision, is located in the TAM protected object space using the path that is set on the *ObjectName* attribute of the TFIM STS universal user context input to the TAMAuthorizationSTModule module.

The following diagram shows the configuration of IBM Integration Bus, TFIM V6.2, and TAM to enable authentication, mapping, and authorization of an identity in a message flow:

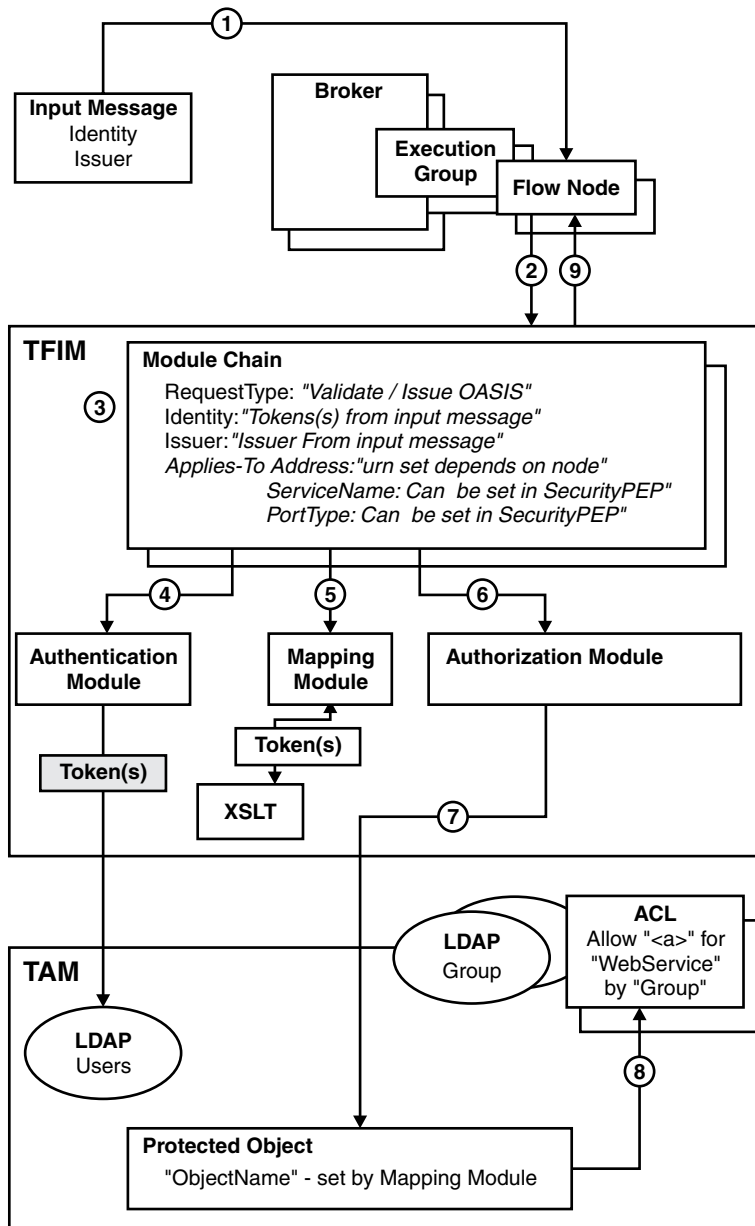


Figure 23. Diagram showing the flow of interaction between Message Broker, TFIM V6.2, and TAM.

The numbers in the preceding diagram correspond to the following sequence of events:

1. A message enters a message flow.
2. A WS-Trust request is issued by the broker, with the *RequestType*, *Issuer*, and *AppliesTo* properties set.
3. TFIM selects a module chain to process the WS-Trust request, based on the *RequestType*, *Issuer*, and *AppliesTo* properties of the request.
4. A module chain can perform authentication if it includes a module in *Validate* mode that is appropriate to the token type that is being passed in the request from the message flow input message. For example, a Username and Password token can be authenticated using a UsernameTokenSTSM module .

5. The module chain must perform some mapping by using an XSLTransformationModule in mapping mode to manipulate the identity information and to provide the required context attributes in the TFIM stsuser object for use by subsequent modules.
6. A module chain can perform authorization in TAM by using the TAMAuthorizationSTSMModule.
7. The TAMAuthorizationSTSMModule performs the authorization check by making a request to TAM with these properties:
 - Action = *a* (where *a* is the stsuser context action attribute). For example, *x* for eExecute could be set using the following code:

```
<stsuser:ContextAttributes>
  <!-- Action -->
  <stsuser:Attribute name="Action" type="urn:ibm:names:ITFIM:stsmodule:tamazn">
    <stsuser:Value>x</stsuser:Value>
  </stsuser:Attribute>
</stsuser:ContextAttributes>
```

- Action Group = *WebService*
- Protected Object = *ProtectedObjectName* (where *ProtectedObjectName* is the stsuser context action attribute). For example, *x* for eExecute could be set using the following code:

```
<stsuser:ContextAttributes>
  <!-- ObjectName -->
  <stsuser:Attribute name="ObjectName" type="urn:ibm:names:ITFIM:stsmodule:tamazn">
    <stsuser:Value>ProtectedObjectName</stsuser:Value>
  </stsuser:Attribute>
</stsuser:ContextAttributes>
```

Typically, *ProtectedObjectName* is set conditionally from the *AppliesTo* information in the request.

8. TAM processes the authorization request by:
 - a. Finding the Access Control Lists (ACLs) associated with protected object *ProtectedObjectName*
 - b. Checking whether the ACLs grant action *a* on action group *WebService* to the user (the user is named either directly or indirectly, through membership of a named group).
9. The WS-Trust reply is returned to the broker. If this action is the result of a mapping request, the WS-Trust reply contains the mapped identity token.

For further information about how to configure TFIM and TAM, see IBM Security Systems information centers.

Related concepts:

“Identity” on page 251

In IBM Integration Bus, an identity is a security token that uniquely identifies an individual, or that provides a set of assertions that can be validated.

“Authentication and validation” on page 257

Authentication is the process of establishing the identity of a user or system and verifying that the identity is valid. Applying authentication to a SAML security token involves validating the assertions that it carries and confirming that it is being processed within its validity period.

“Authorization” on page 261

Authorization is the process of verifying that an identity token has permission to access a message flow.

“Identity mapping” on page 263

Identity mapping is the transformation of a security token from one format to another format, or the federation of an identity from one realm to an equivalent identity in another realm.

“Security profiles” on page 247

A security profile defines the security operations that are to be performed in a message flow at SecurityPEP nodes and security enabled input and output nodes.

Related tasks:

“Configuring identity authentication and security token validation” on page 312

You can configure a message flow to perform identity authentication or security token validation using Integrated Windows Authentication (IWA), Lightweight Directory Access Protocol (LDAP), a WS-Trust V1.3 compliant security token server (STS) such as Tivoli Federated Identity Manager (TFIM) Version 6.2, or HTTP requests. Support for TFIM V6.1 is also provided, for compatibility with previous versions of IBM Integration Bus.

“Configuring identity mapping” on page 333

Configure a security token server (STS), such as Tivoli Federated Identity Manager (TFIM) V6.2, to map the incoming security token and, if required, to authenticate and authorize it.

“Creating a security profile” on page 294

You can create a security profile for use with Lightweight Directory Access Protocol (LDAP) or a WS-Trust V1.3 compliant security token server (STS), such as Tivoli Federated Identity Manager (TFIM) V6.2. Support is also provided for TFIM V6.1, for compatibility with previous versions of IBM Integration Bus. You can create the security profile by using either the `mqsicreateconfigurablebservice` command or an editor in the IBM Integration Explorer.

“Configuring authorization” on page 342

Configure the broker to use an external security provider to authorize an identity in a message flow. You can use either Lightweight Directory Access Protocol (LDAP) or a WS-Trust V1.3 compliant security token server (STS) such as Tivoli Federated Identity Manager (TFIM) V6.2. Support for TFIM V6.1 is also provided for compatibility with previous versions.

“Setting up message flow security” on page 292

Set up security on a message flow to control access based on the identity of a message passing through the message flow.

Related reference:



MQInput node

Use the MQInput node to receive messages from clients that connect to the broker by using the WebSphere MQ Enterprise Transport, and that use the MQI and AMI application programming interfaces.



SecurityPEP node

Use the SecurityPEP node in a message flow to invoke the message flow security manager at any point in the message flow.



HTTPInput node

Use the HTTPInput node to receive an HTTP message from an HTTP client for processing by a message flow.



SOAPInput node

Use the SOAPInput node to process client SOAP messages, so that the broker operates as a SOAP Web Services provider.

Identity and security token propagation

Identity and security token propagation enables the identity and security tokens (associated with each message) to be propagated throughout a message flow, and on to target applications through output or request nodes.

In an enterprise system, you can use different physical identities or security tokens (such as user names, certificates, and SAML assertions) to represent a single logical identity through different parts of the enterprise. The propagation of an identity or security token ensures that the logical identity is kept throughout the system by mapping between the various physical forms as necessary. For example, a message might enter the system using a certificate, but a user name token might be required for server processing of the message. Identity mapping is used to convert from the certificate to the username token, and identity propagation ensures that the mapped identity is placed in the correct place for the outbound transport.

When an output or request node propagates an identity, the mapped identity is used. If the mapped identity is not set, or if it has a token type that is not supported by the node, the source identity is used. If no identity is set, or if neither the mapped nor source identity has a token type that is supported by the node, a security exception is thrown by the node.

The output nodes that support identity propagation are:

- CICSRequest
- HTTPRequest
- IMSRequest
- MQOutput
- SAPRequest
- SCAAsyncRequest
- SCAResult
- SOAPAsyncRequest
- SOAPRequest

The following table shows the support that is provided by the message flow security manager for the propagation of the different types of security token. For more information about these security tokens, see “Identity” on page 251.

Table 4. Support for security token types - token propagation

Token type (format)	Broker security manager support	Token propagated in
Username	Username tokens are supported for propagation by the following nodes: <ul style="list-style-type: none"> • CICSRequest • HTTPRequest • IMSRequest • MQOutput • SCAAsyncRequest • SCAResult • SOAPRequest 	CICS® The request security credentials HTTP BasicAuth header IMS The request security credentials IWA The request security credentials MQ MQMD.UserIdentifier transport header

Table 4. Support for security token types - token propagation (continued)

Token type (format)	Broker security manager support	Token propagated in
Username and password	Username and password tokens are supported for propagation by the following nodes: <ul style="list-style-type: none"> • CICSRequest • HTTPRequest • IMSRequest • MQOutput • SAPRequest • SCAAsyncRequest • SCAResult • SOAPAsyncRequest • SOAPRequest 	CICS The request security credentials HTTP BasicAuth header IMS The request security credentials IWA The request security credentials MQ MQMD.UserIdentifier transport header SAP The request security credentials SOAP <ul style="list-style-type: none"> • BasicAuth header, if there is no policy set and binding • SOAP header, if a policy set and binding sets the Username token profile • Kerberos client credentials, if a policy set and binding sets the Kerberos token profile
SAML assertion	SAML tokens are supported for propagation by the following nodes: <ul style="list-style-type: none"> • SOAPRequest • SOAPAsyncRequest 	SOAP header when a policy set and binding sets the SAML token profile
X.509 certificate	X.509 tokens are supported for propagation by the following nodes: <ul style="list-style-type: none"> • SOAPRequest • SOAPAsyncRequest 	SOAP header when a policy set and binding sets the X.509 binary token profile
LTPA v2 token	LTPA v2 tokens are supported for propagation by the following nodes: <ul style="list-style-type: none"> • SOAPRequest • SOAPAsyncRequest 	SOAP header when a policy set and binding sets the LTPA token profile
Universal WSSE token	Universal WSSE tokens are not supported for propagation by any node.	

For information about how to configure a message flow to propagate a message identity, see “Configuring for identity propagation” on page 389. For more information about how one physical identity is converted to another, see “Identity mapping” on page 263.

Related concepts:

“Identity” on page 251

In IBM Integration Bus, an identity is a security token that uniquely identifies an individual, or that provides a set of assertions that can be validated.

“Authentication and validation” on page 257

Authentication is the process of establishing the identity of a user or system and verifying that the identity is valid. Applying authentication to a SAML security token involves validating the assertions that it carries and confirming that it is being processed within its validity period.

“Identity mapping” on page 263

Identity mapping is the transformation of a security token from one format to another format, or the federation of an identity from one realm to an equivalent identity in another realm.

“Authorization” on page 261

Authorization is the process of verifying that an identity token has permission to

access a message flow.

“Security profiles” on page 247

A security profile defines the security operations that are to be performed in a message flow at SecurityPEP nodes and security enabled input and output nodes.

“Integrated Windows Authentication” on page 260

Integrated Windows Authentication (IWA) refers to a set of authentication protocols, NTLM, Kerberos, and SPNEGO, that are used to provide transport-level security. You can configure IBM Integration Bus to provide an IWA-secured service on a broker running on any operating system, and to consume an IWA-secured service on a broker running on Windows, when you are using the HTTP and SOAP nodes.

Related tasks:

“Configuring identity mapping” on page 333

Configure a security token server (STS), such as Tivoli Federated Identity Manager (TFIM) V6.2, to map the incoming security token and, if required, to authenticate and authorize it.

“Creating a security profile” on page 294

You can create a security profile for use with Lightweight Directory Access Protocol (LDAP) or a WS-Trust V1.3 compliant security token server (STS), such as Tivoli Federated Identity Manager (TFIM) V6.2. Support is also provided for TFIM V6.1, for compatibility with previous versions of IBM Integration Bus. You can create the security profile by using either the `mqsicreateconfigurable` command or an editor in the IBM Integration Explorer.

“Configuring for identity propagation” on page 389

To enable a message flow to perform identity propagation, the input nodes must extract the identity and the output node must propagate it.

“Providing credentials in HTTP requests” on page 328

Use a security profile to configure HTTPRequest and SOAPRequest nodes to authenticate with a remote server.

“Setting up message flow security” on page 292

Set up security on a message flow to control access based on the identity of a message passing through the message flow.



Propagating security credentials to an SAP request

The SAPRequest node can use an identity that is present in the Properties folder of the message tree structure for the security credentials in a request, by using the Propagate property on the security profile that is defined on the node.



Propagating security credentials to IMS

The IMSRequest node can use an identity that is present in the Properties folder of the message tree structure for the security credentials in a request, by using the Propagate property on the security profile that is defined for the node.



Propagating security credentials to CICS Transaction Server for z/OS

The CICSRequest node can use an identity that is present in the Properties folder of the message tree structure for the security credentials in a request, by using the Propagate property on the security profile that is defined on the node.

Related reference:



CICSRequest node

Use the CICSRequest node to call CICS Transaction Server for z/OS programs over TCP/IP-based IP InterCommunications (IPIC) protocol.

HTTPInput node

Use the HTTPInput node to receive an HTTP message from an HTTP client for processing by a message flow.

MQInput node

Use the MQInput node to receive messages from clients that connect to the broker by using the WebSphere MQ Enterprise Transport, and that use the MQI and AMI application programming interfaces.

SOAPInput node

Use the SOAPInput node to process client SOAP messages, so that the broker operates as a SOAP Web Services provider.

SecurityPEP node

Use the SecurityPEP node in a message flow to invoke the message flow security manager at any point in the message flow.

SCAInput node

Use the SCAInput node with the SCAREply node to process messages from WebSphere Process Server.

SCAResponse node

Use the SCAResponse node to send a request to WebSphere Process Server. The node is a synchronous request and response node, and blocks after sending the request until the response is received. The node can also send one-way requests.

SAPRequest node

Use the SAPRequest node to send requests to an SAP application.

IMSRequest node

Use the IMSRequest node to send a request to run a transaction on a local or remote IMS system, and wait for a response. IMS Connect must be configured and running on the IMS system.

Security identities for integration nodes connecting to external systems

You can access external systems from the message flows that you deploy to your integration nodes, and you must therefore consider the steps that you might want to take to secure that access.

You can set a user ID and password that you want the integration node to use for access to an external system or database by using the **mqsisetdbparms** command on a configurable service; see Securing database connections.

After you have defined user IDs, you must authorize those IDs so that the integration node can access your databases from deployed message flows. See the documentation for your database provider to authorize your integration node user ID.

If you migrated your integration node from a previous release, the integration node accessed a database for its own use. You might have defined the user ID and password that is used to access that database by specifying a database connection user ID and password with the **-u** and **-p** parameters on the **mqsicreatebroker** command. Alternatively, you might have used the integration node service user ID and its password (specified with the **-i** and **-a** parameters on the same command). When you migrate the integration node, these parameters are migrated and stored,

and are used by the migrated integration node for access to databases that do not have specific ID access defined. On WebSphere Message Broker Version 8.0, you can use only the **mqsisetdbparms** command to set or change values for database access by the integration node, because the parameters are deprecated. (The service user ID and password are required on Windows, but are no longer used for database access.) To view what security credentials are set, you can use the **mqsireportdbparms** command; see “Checking the security credentials that are used by an integration node” on page 392.

If you have defined a user ID and password by using any of the previously described methods, then you do not need to create a security profile. To see how to set security authorization for a node, refer to the table of properties for that specific node.

IBM Integration Bus does not provide special commands to administer databases. Discuss your database security requirements with the database administrator for the database manager that you are using, or refer to the documentation provided by your database supplier.

Related concepts:

“Security overview” on page 203

When you are designing an IBM Integration Bus application it is important to consider the security measures that are needed to protect the information in the system.

Related reference:



mqsicreatebroker command

Use the **mqsicreatebroker** command to create a broker and its associated resources.



mqsichangebroker command

Use the **mqsichangebroker** command to change one or more of the configuration parameters of the broker.



mqsisetdbparms command

Use the **mqsisetdbparms** command to associate a specific user ID and password (or SSH identity file) with one or more resources that are accessed by the broker.



mqsireportdbparms command

Use the **mqsireportdbparms** command to list all parameters that are set for a specific broker.

Security exception processing

A security exception is raised when a message flow security failure occurs during security processing in an input node or SecurityPEP node.

Security exceptions are processed in a different way from other errors on the input node. An error is typically caught on the input node and routed down the Failure terminal for error processing, but security exceptions are not processed in the same way. By default, the broker does not allow security exceptions to be caught within the message flow, but backs the message out or returns an error (as in the case of HTTP). Security exceptions in input nodes are managed in this way to prevent a security denial of service attack filling the logs and destabilizing the system.

However, security exceptions in SecurityPEP nodes are managed in a different way. If a security operation fails in a SecurityPEP node, a security exception is raised, wrapped in a normal recoverable exception, which invokes the error handling that is provided by the message flow.

If you have designed the message flow to be in a secure area and you want to explicitly perform processing of security exceptions, you can select the **Treat Security Exceptions as normal exceptions** property on the MQInput or HTTPInput nodes. This property causes security exceptions to be processed in the same way as other exceptions in the message flow.

If you associate the Default Propagation security profile with an output or request node, the token type of the mapped or source security token must be the same as the transport default for that node; otherwise, a security exception occurs. For example, for an MQOutput node, the token type must be *Username*, for an HTTPRequest node, the token type must be *Username + Password*, and for a SOAPRequest node, the token type must be the type that is defined in either the policy set and binding or the transport binding.

For information on how to diagnose the causes of security exceptions, see “Diagnosing security problems” on page 394.

Related concepts:

“Identity” on page 251

In IBM Integration Bus, an identity is a security token that uniquely identifies an individual, or that provides a set of assertions that can be validated.

“Authentication and validation” on page 257

Authentication is the process of establishing the identity of a user or system and verifying that the identity is valid. Applying authentication to a SAML security token involves validating the assertions that it carries and confirming that it is being processed within its validity period.

“Identity mapping” on page 263

Identity mapping is the transformation of a security token from one format to another format, or the federation of an identity from one realm to an equivalent identity in another realm.

“Authorization” on page 261

Authorization is the process of verifying that an identity token has permission to access a message flow.

“Identity and security token propagation” on page 286

Identity and security token propagation enables the identity and security tokens (associated with each message) to be propagated throughout a message flow, and on to target applications through output or request nodes.

“Security profiles” on page 247

A security profile defines the security operations that are to be performed in a message flow at SecurityPEP nodes and security enabled input and output nodes.

Related tasks:

“Configuring the extraction of an identity or security token” on page 308

You can configure the SecurityPEP node or security enabled input nodes to extract the identity or security token from a message and store it in the properties tree identity fields, enabling it to be processed throughout the message flow and propagated at output or request nodes.

“Setting up message flow security” on page 292

Set up security on a message flow to control access based on the identity of a message passing through the message flow.

“Diagnosing security problems” on page 394

This topic explains how to find out why access to a secured flow has been denied.

Related reference:

 **mqsichangeproperties** command

Use the **mqsichangeproperties** command to modify broker properties and properties of broker resources.

 **mqsireportproperties** command

Use the **mqsireportproperties** command to display properties that relate to a broker, an integration server, or a configurable service.

 MQInput node

Use the MQInput node to receive messages from clients that connect to the broker by using the WebSphere MQ Enterprise Transport, and that use the MQI and AMI application programming interfaces.

 HTTPInput node

Use the HTTPInput node to receive an HTTP message from an HTTP client for processing by a message flow.

Setting up message flow security

Set up security on a message flow to control access based on the identity of a message passing through the message flow.

About this task

You can configure the broker to perform end-to-end processing of an identity carried in a message through a message flow. Administrators can configure security at message flow level, controlling access based on the identity flowed in a message. This security mechanism is independent of both the transport and the message format.

To set up security for a message flow, perform the tasks described in the following topics:

Procedure

1. “Creating a security profile” on page 294
2. “Configuring the extraction of an identity or security token” on page 308
3. “Configuring identity authentication and security token validation” on page 312
4. “Configuring identity mapping” on page 333
5. “Configuring authorization” on page 342
6. “Configuring for identity propagation” on page 389
7. “Security identities for integration nodes connecting to external systems” on page 289
8. “Diagnosing security problems” on page 394

What to do next

If the message flow is a Web service implemented by using the SOAP nodes, and the identity is to be taken from the WS-Security header tokens, you must also

create appropriate Policy sets and bindings, then configure them on the relevant SOAP nodes (in addition to the security profile). See Associating policy sets and bindings with message flows and nodes.

To work with an identity, you must configure the policy sets and bindings for the relevant capabilities:

- To work with a Username and Password identity, configure the policy sets and bindings for Username token capabilities.
- To work with an X.509 Certificate identity, configure the policy sets and bindings for X.509 certificate token capabilities.

In the policy set binding, the *Certificates* mode of the X.509 certificate authentication token must be set as *Trust Any* (rather than *Trust Store*), so that the certificate is passed to the security provider defined by the security profile. Setting *Trust Store* causes the certificate to be validated in the local broker trust store.

- To work with a SAML assertion, configure the policy sets and bindings for SAML token capabilities.
- To work with an LTPA token, configure the policy sets and bindings for LTPA token capabilities.
- To work with a Kerberos ticket, configure the policy sets and bindings for Kerberos token capabilities.

For more information, see Policy Sets and Policy Set Bindings editor: Authentication and Protection Tokens panel.

Related concepts:

“Identity” on page 251

In IBM Integration Bus, an identity is a security token that uniquely identifies an individual, or that provides a set of assertions that can be validated.

“Authentication and validation” on page 257

Authentication is the process of establishing the identity of a user or system and verifying that the identity is valid. Applying authentication to a SAML security token involves validating the assertions that it carries and confirming that it is being processed within its validity period.

“Identity mapping” on page 263

Identity mapping is the transformation of a security token from one format to another format, or the federation of an identity from one realm to an equivalent identity in another realm.

“Authorization” on page 261

Authorization is the process of verifying that an identity token has permission to access a message flow.

“Identity and security token propagation” on page 286

Identity and security token propagation enables the identity and security tokens (associated with each message) to be propagated throughout a message flow, and on to target applications through output or request nodes.

“Security profiles” on page 247

A security profile defines the security operations that are to be performed in a message flow at SecurityPEP nodes and security enabled input and output nodes.

“Message flow security overview” on page 243

IBM Integration Bus provides a security manager, which enables you to control access to individual messages in a message flow, using the identity of the message.

Related tasks:

“Configuring the extraction of an identity or security token” on page 308

You can configure the SecurityPEP node or security enabled input nodes to extract the identity or security token from a message and store it in the properties tree identity fields, enabling it to be processed throughout the message flow and propagated at output or request nodes.

“Creating a security profile”

You can create a security profile for use with Lightweight Directory Access Protocol (LDAP) or a WS-Trust V1.3 compliant security token server (STS), such as Tivoli Federated Identity Manager (TFIM) V6.2. Support is also provided for TFIM V6.1, for compatibility with previous versions of IBM Integration Bus. You can create the security profile by using either the **mqsicreateconfigurableservice** command or an editor in the IBM Integration Explorer.

“Creating a security profile for LDAP” on page 296

Create a security profile for use with Lightweight Directory Access Protocol (LDAP) or Secure LDAP (LDAPS), by using either the **mqsicreateconfigurableservice** command or an editor in the IBM Integration Explorer.

“Configuring identity authentication and security token validation” on page 312

You can configure a message flow to perform identity authentication or security token validation using Integrated Windows Authentication (IWA), Lightweight Directory Access Protocol (LDAP), a WS-Trust V1.3 compliant security token sever (STS) such as Tivoli Federated Identity Manager (TFIM) Version 6.2, or HTTP requests. Support for TFIM V6.1 is also provided, for compatibility with previous versions of IBM Integration Bus.

“Configuring identity mapping” on page 333

Configure a security token server (STS), such as Tivoli Federated Identity Manager (TFIM) V6.2, to map the incoming security token and, if required, to authenticate and authorize it.

“Configuring authorization” on page 342

Configure the broker to use an external security provider to authorize an identity in a message flow. You can use either Lightweight Directory Access Protocol (LDAP) or a WS-Trust V1.3 compliant security token server (STS) such as Tivoli Federated Identity Manager (TFIM) V6.2. Support for TFIM V6.1 is also provided for compatibility with previous versions.

“Configuring for identity propagation” on page 389

To enable a message flow to perform identity propagation, the input nodes must extract the identity and the output node must propagate it.

Creating a security profile

You can create a security profile for use with Lightweight Directory Access Protocol (LDAP) or a WS-Trust V1.3 compliant security token server (STS), such as Tivoli Federated Identity Manager (TFIM) V6.2. Support is also provided for TFIM V6.1, for compatibility with previous versions of IBM Integration Bus. You can create the security profile by using either the **mqsicreateconfigurableservice** command or an editor in the IBM Integration Explorer.

About this task

Before you can enable security on a node or a message flow, you need to create a security profile that defines the security operations that you want to perform.

You can create a security profile for use with external security providers to provide the required security enforcement and mapping. You can configure the security

profile to use different security providers for different security functions; for example, you might use LDAP for authentication and WS-Trust V1.3 STS for mapping and authorization.

If you want to extract and propagate an identity without security enforcement or mapping, you can use the supplied security profile called Default Propagation. The Default Propagation profile is a predefined profile that requests only identity propagation.

To create a security profile, see:

- “Creating a security profile for LDAP” on page 296
- “Creating a security profile for WS-Trust V1.3 (TFIM V6.2)” on page 302
- “Creating a security profile for TFIM V6.1” on page 306

Related concepts:

“Identity” on page 251

In IBM Integration Bus, an identity is a security token that uniquely identifies an individual, or that provides a set of assertions that can be validated.

“Authentication and validation” on page 257

Authentication is the process of establishing the identity of a user or system and verifying that the identity is valid. Applying authentication to a SAML security token involves validating the assertions that it carries and confirming that it is being processed within its validity period.

“Identity mapping” on page 263

Identity mapping is the transformation of a security token from one format to another format, or the federation of an identity from one realm to an equivalent identity in another realm.

“Authorization” on page 261

Authorization is the process of verifying that an identity token has permission to access a message flow.

“Identity and security token propagation” on page 286

Identity and security token propagation enables the identity and security tokens (associated with each message) to be propagated throughout a message flow, and on to target applications through output or request nodes.

“Security profiles” on page 247

A security profile defines the security operations that are to be performed in a message flow at SecurityPEP nodes and security enabled input and output nodes.

“Security exception processing” on page 290

A security exception is raised when a message flow security failure occurs during security processing in an input node or SecurityPEP node.

Related tasks:

“Configuring the extraction of an identity or security token” on page 308

You can configure the SecurityPEP node or security enabled input nodes to extract the identity or security token from a message and store it in the properties tree identity fields, enabling it to be processed throughout the message flow and propagated at output or request nodes.

“Configuring identity authentication and security token validation” on page 312

You can configure a message flow to perform identity authentication or security token validation using Integrated Windows Authentication (IWA), Lightweight Directory Access Protocol (LDAP), a WS-Trust V1.3 compliant security token sever (STS) such as Tivoli Federated Identity Manager (TFIM) Version 6.2, or HTTP requests. Support for TFIM V6.1 is also provided, for compatibility with previous versions of IBM Integration Bus.

“Configuring identity mapping” on page 333

Configure a security token server (STS), such as Tivoli Federated Identity Manager (TFIM) V6.2, to map the incoming security token and, if required, to authenticate and authorize it.

“Configuring authorization” on page 342

Configure the broker to use an external security provider to authorize an identity in a message flow. You can use either Lightweight Directory Access Protocol (LDAP) or a WS-Trust V1.3 compliant security token server (STS) such as Tivoli Federated Identity Manager (TFIM) V6.2. Support for TFIM V6.1 is also provided for compatibility with previous versions.

“Configuring for identity propagation” on page 389

To enable a message flow to perform identity propagation, the input nodes must extract the identity and the output node must propagate it.

“Setting up message flow security” on page 292

Set up security on a message flow to control access based on the identity of a message passing through the message flow.

Related reference:



mqsicreateconfigurable-service command

Use the **mqsicreateconfigurable-service** command to create an object name for a broker external resource.

Creating a security profile for LDAP:

Create a security profile for use with Lightweight Directory Access Protocol (LDAP) or Secure LDAP (LDAPS), by using either the **mqsicreateconfigurable-service** command or an editor in the IBM Integration Explorer.

Before you begin

Before you start:

Ensure that you have an LDAP server that is LDAP Version 3 compliant, for example:

- IBM Tivoli Directory Server
- Microsoft Active Directory
- OpenLDAP.

About this task

If your LDAP directory does not permit login by unrecognized user IDs, and does not grant search access rights on the subtree, you must also set up a separate authorized login ID that the broker can use for the search. For information on how to do this, see “Configuring authorization with LDAP” on page 343 or “Authenticating incoming requests with LDAP” on page 319.

Creating a security profile using mqsicreateconfigurable-service:

About this task

You can use the **mqsicreateconfigurable-service** command to create a security profile that uses LDAP for authentication, authorization, or both. The security profile ensures that each message has an authenticated ID and is authorized for the message flow.

Procedure

1. Open a command window that is configured for your environment.
2. Enter the **mqsicreateconfigurablesevice** command on the command line. For example:

```
mqsicreateconfigurablesevice WBRK_BROKER -c SecurityProfiles -o LDAP
-n authentication,authenticationConfig,authorization,authorizationConfig,propagation,rejectBlankpassword
-v "LDAP,\"ldap://ldap.acme.com:389/ou=sales,o=acme.com\",LDAP,
\"ldap://ldap.acme.com:389/cn=A11 Sales,ou=acmegroups,o=acme.com\",TRUE,TRUE
```

You must enclose the LDAP URL (which contains commas) with escaped double quotation marks (`\"` and `\"`) so that the URL commas are not confused with the comma separator of the value parameter of **mqsicreateconfigurablesevice**.

If the LDAP URL includes an element name with a space, in this case `cn=A11 Sales`, the set of values after the `-v` flag must be enclosed by double quotation marks, (`"`)

For more information about the structure of the command, refer to the **mqsicreateconfigurablesevice** command.

You can define the security-specific parts of the command in the following way:

- a. Set the **authentication** to *LDAP*. This ensures that the incoming identity is validated.
- b. Set the **authenticationConfig** using the following syntax:

```
ldap[s]://server[:port]/baseDN[?uid_attr][?base|sub]]
```

For example:

```
ldap://ldap.acme.com:389/ou=sales,o=acme.com
ldaps://localhost:636/ou=sales,o=acme?cn?base
```

ldap: (Required) Fixed protocol string.

s: (Optional) Specifies whether SSL should be used. Default is not to use SSL.

server: (Required) The name or IP address of the LDAP server to contact.

port: (Optional) The port to connect to. Default is 389 (non-SSL). For LDAP servers with SSL enabled, the port is typically 636.

baseDN

(Required) String defining the base distinguished name (DN) of all users in the directory. If users exist in different subtrees, specify a common subtree under which a search on the username uniquely resolves to the required user entry, and set the *sub* attribute.

uid_attr:

(Optional) String defining the attribute to which the incoming username maps, typically *uid*, *CN*, or email address. Default is *uid*.

base|sub:

(Optional) Defines whether to perform a base or subtree search. If *base* is defined, the authentication is faster because the DN of the user can be constructed from the *uid_attr*, *username*, and *baseDN* values. If *sub* is selected, a search must be performed before the DN can be resolved. Default is *sub*.

- c. Set the **authorization** to *LDAP*. This ensures that the incoming identity is checked for group membership in LDAP.
- d. Set the **authorizationConfig** using the following syntax:

```
ldap[s]://server[:port]/groupDN[?[member_attr]
[?[base|sub][?[x-userBaseDN=baseDN,
x-uid_attr=uid_attr]]]]
```

For example:

```
ldap://ldap.acme.com:389/cn=All Sales,ou=acmegroups,
o=acme.com?uniquemember?sub?x-userBaseDN=ou=sales%2co=ibm.com,
x-uid_attr=emailaddress
```

ldap: (Required) Fixed protocol string

s: (Optional) Specifies whether SSL is used. Default is not to use SSL.

server: (Required) The name or IP address of the LDAP server to contact.

port: (Optional) The port to connect to. Default is 389 (non-SSL). For LDAP servers with SSL enabled, the port is typically 636.

groupDN

(Required) Fully defined distinguished name of the group in which users must be members to be granted access.

member_attr:

(Optional) The attribute of the group used to filter the search.

Default is to look for both **member** and **uniquemember** attributes.

The following options are required only if authentication has not preceded the authorization, and if the authentication configuration string has not been specified. If the authentication configuration string has been specified, the following parameters are ignored and those provided by the **baseDN**, **uid_attr**, and **[base|sub]** for authentication are used instead:

base|sub:

(Optional) Defines whether to perform a base or subtree search. If base is defined, the authentication is faster because the DN of the user can be constructed from uid_attr + username + baseDN. If sub is selected, a search must be performed before the DN can be resolved. Default is sub.

baseDN

(Optional) String defining the base distinguished name of all users in the directory. Must be preceded by the string x-userBaseDN. Any commas in the BaseDN must be rendered as %2c.

x-uid_attr:

(Optional) String defining the attribute to which the incoming username should map, typically uid, CN, or email address. Default is uid. Must be preceded by the string x-uid_attr.

When you submit the command from a batch (.bat) file or command (.cmd) file, if the LDAP URL includes an extension with LDAP URL "percent hex hex" escaped characters (for example, a comma replaced by %2c, or a space replaced by %20), the percent signs must be escaped from the batch interpreter with an extra percent sign (%). For example:

```
mqsicreateconfigurableservice WBRK_BROKER -c SecurityProfiles -o LDAP_URI_FUN
-n authentication,authenticationConfig,authorization,authorizationConfig -v
"LDAP,\"ldap://ldap.acme.com:389/ou=sales,o=acme.com?emailaddress?sub\",
LDAP,\"ldap://ldap.acme.com:389/cn=All Sales,ou=acmegroups,
o=acme.com?report?base?x-BaseDN=ou=sales%2co=acme.com,
x-uid_attr=emailaddress\""
```

The selected group must be defined on the LDAP server, and all of the required users must be members of the group.

3. If you need to reconfigure the security profile after it has been created, use the **mqsichangeproperties** command. You must stop and start the integration server for the change of property value to take effect.

Creating a security profile using the IBM Integration Explorer:

About this task

You can use the IBM Integration Explorer to create a security profile for LDAP.

Procedure

1. In the IBM Integration Explorer, right-click on the broker with which you want to work, and click **Properties**.
2. In the Properties window, select the Security tab, and click **Security Profiles**. The Security Profiles window is displayed, containing a list of existing security profiles for the broker on the left, and a pane in which you can configure the profile on the right.
3. Click **Add** to create a new profile and add it to the list. You can edit the name of the security profile by highlighting it in the list and pressing **F2**.
4. Configure the security profile using the entry fields on the right side of the pane:
 - a. Select the type of **Authentication** required. This can be LDAP, TFIM, or NONE.
 - b. If you have selected LDAP for authentication, edit the following fields in the **LDAP Parameters** section:
 - LDAP Host
 - LDAP baseDN
 - LDAP uid attr
 - LDAP search Scope

The values that you enter in the **LDAP Parameters** fields create a configuration string, which is displayed in the **Authentication Config** field. For information about the valid values for the parameters, see “Creating a security profile using mqsicreateconfigurableservice” on page 296.
 - c. Select the type of **Mapping** required. This can be either TFIM or NONE.
 - d. If you have selected TFIM for mapping, type the URL of the TFIM server in the **TFIM Configuration** field of the **TFIM Parameters** section.

The value that you specify in the **TFIM Configuration** field creates a configuration string, which is displayed in the **Mapping Config** field.
 - e. Select the type of **Authorization** required. This can be LDAP, TFIM, or NONE.
 - f. If you have selected LDAP for authorization, edit the following fields in the **LDAP Parameters** section:
 - LDAP Host
 - LDAP baseDN
 - LDAP uid attr
 - LDAP search Scope
 - LDAP group baseDN
 - LDAP group member.

The values that you enter in the **LDAP Parameters** fields create a configuration string, which is displayed in the **Authorization Config** field. For information about the valid values for the parameters, see “Creating a security profile using mqsicreateconfigurableservice” on page 296.

- g. In the **Propagation** field, specify whether you require the identity to be propagated. The default is `False`.
- h. In the **Reject Empty Password** field, specify whether you want the security manager to reject a username that has an empty password token, without passing it to LDAP. The default is `False`, which means that a username is passed to LDAP even if it has an empty password token.
- i. In the **Password Value** field, select the way in which the password is displayed in the properties folder. The options are:
 - PLAIN** The password appears in the Properties folder as plain text.
 - OBFUSCATE**
The password appears in the Properties folder as base64 encoding.
 - MASK** The password appears in the Properties folder as four asterisks (****).

5. Click **Finish** to deploy the security profile to the broker.

Results

To delete an existing security profile, select the profile in the list and then click **Delete**.

Related concepts:

“Identity” on page 251

In IBM Integration Bus, an identity is a security token that uniquely identifies an individual, or that provides a set of assertions that can be validated.

“Authentication and validation” on page 257

Authentication is the process of establishing the identity of a user or system and verifying that the identity is valid. Applying authentication to a SAML security token involves validating the assertions that it carries and confirming that it is being processed within its validity period.

“Identity mapping” on page 263

Identity mapping is the transformation of a security token from one format to another format, or the federation of an identity from one realm to an equivalent identity in another realm.

“Authorization” on page 261

Authorization is the process of verifying that an identity token has permission to access a message flow.

“Identity and security token propagation” on page 286

Identity and security token propagation enables the identity and security tokens (associated with each message) to be propagated throughout a message flow, and on to target applications through output or request nodes.

“Security profiles” on page 247

A security profile defines the security operations that are to be performed in a message flow at SecurityPEP nodes and security enabled input and output nodes.

“Security exception processing” on page 290

A security exception is raised when a message flow security failure occurs during security processing in an input node or SecurityPEP node.

Related tasks:

“Configuring the extraction of an identity or security token” on page 308

You can configure the SecurityPEP node or security enabled input nodes to extract the identity or security token from a message and store it in the properties tree identity fields, enabling it to be processed throughout the message flow and propagated at output or request nodes.

“Configuring identity authentication and security token validation” on page 312
You can configure a message flow to perform identity authentication or security token validation using Integrated Windows Authentication (IWA), Lightweight Directory Access Protocol (LDAP), a WS-Trust V1.3 compliant security token server (STS) such as Tivoli Federated Identity Manager (TFIM) Version 6.2, or HTTP requests. Support for TFIM V6.1 is also provided, for compatibility with previous versions of IBM Integration Bus.

“Configuring identity mapping” on page 333

Configure a security token server (STS), such as Tivoli Federated Identity Manager (TFIM) V6.2, to map the incoming security token and, if required, to authenticate and authorize it.

“Creating a security profile” on page 294

You can create a security profile for use with Lightweight Directory Access Protocol (LDAP) or a WS-Trust V1.3 compliant security token server (STS), such as Tivoli Federated Identity Manager (TFIM) V6.2. Support is also provided for TFIM V6.1, for compatibility with previous versions of IBM Integration Bus. You can create the security profile by using either the **mqscreateconfigurable-service** command or an editor in the IBM Integration Explorer.

“Configuring authorization” on page 342

Configure the broker to use an external security provider to authorize an identity in a message flow. You can use either Lightweight Directory Access Protocol (LDAP) or a WS-Trust V1.3 compliant security token server (STS) such as Tivoli Federated Identity Manager (TFIM) V6.2. Support for TFIM V6.1 is also provided for compatibility with previous versions.

“Configuring for identity propagation” on page 389

To enable a message flow to perform identity propagation, the input nodes must extract the identity and the output node must propagate it.

“Setting up message flow security” on page 292

Set up security on a message flow to control access based on the identity of a message passing through the message flow.

Related reference:

 **mqscreateconfigurable-service** command

Use the **mqscreateconfigurable-service** command to create an object name for a broker external resource.

 **msideleteconfigurable-service** command

Use the **msideleteconfigurable-service** command to delete a configurable service, such as a JMS provider, JDBC provider, or FTP server, that you have created by using the **mqscreateconfigurable-service** command.

 **mqschangeproperties** command

Use the **mqschangeproperties** command to modify broker properties and properties of broker resources.

 **mqsireportproperties** command

Use the **mqsireportproperties** command to display properties that relate to a broker, an integration server, or a configurable service.

 MQInput node

Use the MQInput node to receive messages from clients that connect to the broker by using the WebSphere MQ Enterprise Transport, and that use the MQI and AMI application programming interfaces.

HTTPInput node

Use the HTTPInput node to receive an HTTP message from an HTTP client for processing by a message flow.

SOAPInput node

Use the SOAPInput node to process client SOAP messages, so that the broker operates as a SOAP Web Services provider.

SOAPAsyncRequest node

Use the SOAPAsyncRequest node with the SOAPAsyncResponse node to construct a pair of message flows that call a Web service asynchronously.

HTTPRequest node

Use the HTTPRequest node to interact with a web service.

MQOutput node

Use the MQOutput node to send messages to clients that connect to the broker using the WebSphere MQ Enterprise Transport and that use the MQI and AMI application programming interfaces.

Creating a security profile for WS-Trust V1.3 (TFIM V6.2):

You can create a security profile for a WS-Trust V1.3 compliant Security Token Server (STS), for example, Tivoli Federated Identity Manager (TFIM) V6.2, for any combination of the following security operations: authentication, authorization, and mapping.

About this task

You can use either the **mqsicreateconfigurablesevice** command or an editor in the IBM Integration Explorer to create the security profile:

- “Creating a profile using mqsicreateconfigurablesevice”
- “Creating a profile using the IBM Integration Explorer” on page 303

Creating a profile using mqsicreateconfigurablesevice:

About this task

To create a security profile that uses a WS-Trust V1.3 compliant Security Token Server (STS), you can use the **mqsicreateconfigurablesevice** command by setting the configuration parameter to the full URL of the STS. The URL must consist of the transport scheme, host name, port, and path. For TFIM V6.2 WS-Trust V1.3 endpoint, the path is /TrustServerWST13/services/RequestSecurityToken. For example:

```
http://stsserver.mycompany.com:9080/TrustServerWST13/services/RequestSecurityToken
```

Procedure

To create a security profile that uses WS-Trust v1.3 for mapping, enter the following command:

```
mqsicreateconfigurablesevice brokername -c SecurityProfiles  
-o profilename -n mapping,mappingConfig  
-v "WS-Trust v1.3 STS",http://stsserver.mycompany.com:9080/TrustServerWST13/services/RequestSecurityToken
```

If the URL specifies an address beginning with `https://`, an SSL secured connection is used for requests to the WS-Trust v1.3 server. For example, to create a security profile that uses an HTTPS connection to WS-Trust v1.3 for mapping, enter the following command:

```
mqsicreateconfigurableservice brokername -c SecurityProfiles
-o profilename -n mapping,mappingConfig
-v "WS-Trust v1.3 STS",https://stsserver.mycompany.com:9080/TrustServerWST13/services/RequestSecurityToken
```

In addition to specifying the security profile URL as an address beginning with `https://`, you can configure the following advanced parameters, by setting broker environment variables:

MQSI_STS_SSL_PROTOCOL

The version of the SSL protocol to be used. Valid values are:

- SSL
- SSLv3
- TLS

The initial value is SSLv3.

MQSI_STS_SSL_ALLOWED_CIPHERS

A space-separated list of the encryption ciphers that can be used. For a list of all the cipher suites that are supported by IBM Integration Bus, see the Java product information for your operating system. For operating systems that use IBM Java, see Appendix A of the IBM JSSE2 Guide:

<http://www.ibm.com/developerworks/java/jdk/security/60/secguides/jsse2Docs/JSSE2RefGuide.html>

MQSI_STS_REQUEST_TIMEOUT

The STS request timeout, specified in seconds. The initial value is 100. For information about providing environment variables to the broker, see *Setting up a command environment*.

If WS-Trust v1.3 STS is selected for more than one operation (for example, for authentication and mapping), the WS-Trust v1.3 server URL must be identical for all the operations, and is therefore specified only once.

The following example creates a security profile that uses TFIM V6.2 for authentication, mapping, and authorization:

```
mqsicreateconfigurableservice MYBROKER -c SecurityProfiles -o MyWSTrustProfile
-n authentication,mapping,authorization,propagation,mappingConfig
-v "WS-Trust v1.3 STS","WS-Trust v1.3 STS","WS-Trust v1.3 STS",TRUE,http://stsserver.mycompany.com:9080/TrustServerWST13/services/RequestSecurityToken
```

Creating a profile using the IBM Integration Explorer:

About this task

You can use the IBM Integration Explorer to create a security profile for using WS-Trust v1.3.

Procedure

1. In the IBM Integration Explorer, right-click on the broker with which you want to work, and click **Properties**.
2. In the Properties window, select the Security tab, and click **Security Profiles**. The Security Profiles window is displayed, containing a list of existing security profiles for the broker on the left and, on the right, a pane in which you can configure the profile.
3. Click **Add** to create a new profile and add it to the list. You can edit the name of the security profile by highlighting it in the list and pressing **F2**. The security profile name must not include spaces.

4. Configure the security profile using the entry fields on the right side of the pane:
 - a. Select the security provider that you require for Authentication, Mapping, and Authorization. When you select WS-Trust v1.3 STS for any of these options, the **STS URI** field in the **Security Token Service (STS) Parameters** group is enabled.
 - b. Type the URL of the WS-Trust v1.3 STS into the **STS URL** field. The STS URL must contain the following URL parts:
 - Transport scheme (http or https)
 - Host name (a fully qualified domain name)
 - Port
 - Path (for example, for TFIM V6.2: /TrustServerWST13/services/RequestSecurityToken)

For example:

`http://stsserver.mycompany.com:9080/TrustServerWST13/services/RequestSecurityToken`

The URL that you enter forms a configuration string, which is displayed in one or more of the configuration fields (**Authentication Config**, **Mapping Config**, and **Authorization Config**), depending on the security operations that are configured to use WS-Trust v1.3 STS.

For more information about the valid values for the configuration parameter, see “Creating a profile using mqsicreateconfigurable-service” on page 302.

- c. In the **Propagation** field, specify whether you require the identity to be propagated. The default is False.
- d. In the **Password Value** field, select the way in which the password is displayed in the properties folder. The password is optional, and is required only when the token type is Username + Password. The options are:

PLAIN The password is shown in the Properties folder as plain text.

OBFUSCATE

The password is shown in the Properties folder as base64 encoding.

MASK The password is shown in the Properties folder as four asterisks (****).

5. Click **Finish** to deploy the security profile to the broker.

Related concepts:

“Identity” on page 251

In IBM Integration Bus, an identity is a security token that uniquely identifies an individual, or that provides a set of assertions that can be validated.

“Authentication and validation” on page 257

Authentication is the process of establishing the identity of a user or system and verifying that the identity is valid. Applying authentication to a SAML security token involves validating the assertions that it carries and confirming that it is being processed within its validity period.

“Identity mapping” on page 263

Identity mapping is the transformation of a security token from one format to another format, or the federation of an identity from one realm to an equivalent identity in another realm.

“Authorization” on page 261

Authorization is the process of verifying that an identity token has permission to access a message flow.

“Identity and security token propagation” on page 286

Identity and security token propagation enables the identity and security tokens (associated with each message) to be propagated throughout a message flow, and on to target applications through output or request nodes.

“Security profiles” on page 247

A security profile defines the security operations that are to be performed in a message flow at SecurityPEP nodes and security enabled input and output nodes.

Related tasks:

“Configuring the extraction of an identity or security token” on page 308

You can configure the SecurityPEP node or security enabled input nodes to extract the identity or security token from a message and store it in the properties tree identity fields, enabling it to be processed throughout the message flow and propagated at output or request nodes.

“Configuring identity authentication and security token validation” on page 312

You can configure a message flow to perform identity authentication or security token validation using Integrated Windows Authentication (IWA), Lightweight Directory Access Protocol (LDAP), a WS-Trust V1.3 compliant security token server (STS) such as Tivoli Federated Identity Manager (TFIM) Version 6.2, or HTTP requests. Support for TFIM V6.1 is also provided, for compatibility with previous versions of IBM Integration Bus.

“Configuring identity mapping” on page 333

Configure a security token server (STS), such as Tivoli Federated Identity Manager (TFIM) V6.2, to map the incoming security token and, if required, to authenticate and authorize it.

“Creating a security profile” on page 294

You can create a security profile for use with Lightweight Directory Access Protocol (LDAP) or a WS-Trust V1.3 compliant security token server (STS), such as Tivoli Federated Identity Manager (TFIM) V6.2. Support is also provided for TFIM V6.1, for compatibility with previous versions of IBM Integration Bus. You can create the security profile by using either the **mqsicreateconfigurableservice** command or an editor in the IBM Integration Explorer.

“Configuring authorization” on page 342

Configure the broker to use an external security provider to authorize an identity in a message flow. You can use either Lightweight Directory Access Protocol (LDAP) or a WS-Trust V1.3 compliant security token server (STS) such as Tivoli Federated Identity Manager (TFIM) V6.2. Support for TFIM V6.1 is also provided for compatibility with previous versions.

“Configuring for identity propagation” on page 389

To enable a message flow to perform identity propagation, the input nodes must extract the identity and the output node must propagate it.

“Setting up message flow security” on page 292

Set up security on a message flow to control access based on the identity of a message passing through the message flow.

Related reference:

 **mqsicreateconfigurableservice** command

Use the **mqsicreateconfigurableservice** command to create an object name for a broker external resource.

 SecurityPEP node

Use the SecurityPEP node in a message flow to invoke the message flow security manager at any point in the message flow.

Creating a security profile for TFIM V6.1:

You can create a security profile for Tivoli Federated Identity Manager (TFIM) V6.1 for any combination of the following functions: authentication, authorization, and mapping. You can use either the **mqsicreateconfigurablesevice** command or an editor in the IBM Integration Explorer to create the security profile.

About this task

Note: Support for TFIM V6.1 is included for compatibility with previous versions of IBM Integration Bus. If possible, upgrade to TFIM V6.2 and follow the instructions in “Creating a security profile for WS-Trust V1.3 (TFIM V6.2)” on page 302.

Creating a profile using mqsicreateconfigurablesevice:

About this task

To create a security profile that uses TFIM V6.1, you can use the **mqsicreateconfigurablesevice** command by setting the configuration parameter to the URL of the TFIM server. For example: `http://tfimserver.mycompany.com:9080`

Procedure

To create a security profile that uses TFIM V6.1 for mapping, enter the following command:

```
mqsicreateconfigurablesevice brokername -c SecurityProfiles -o profilename  
-n mapping,mappingConfig -v TFIM,http://tfimserver.mycompany.com:9080
```

If the URL specifies an address beginning with `https://`, an SSL secured connection is used for requests to the TFIM server. For example, to create a security profile that uses an HTTPS connection to TFIM for mapping, enter the following command:

```
mqsicreateconfigurablesevice brokername -c SecurityProfiles -o profilename  
-n mapping,mappingConfig -v TFIM,https://tfimserver.mycompany.com:9443
```

where `https://tfimserver.mycompany.com:9443` is the address of the TFIM server. If TFIM is selected for more than one operation (for example, for authentication and mapping), the TFIM server URL must be identical for all the operations, and is therefore specified only once.

The following example creates a security profile that uses TFIM for authentication, mapping, and authorization:

```
mqsicreateconfigurablesevice MYBROKER -c SecurityProfiles -o TFIM  
-n authentication,mapping,authorization,propagation,mappingConfig  
-v TFIM,TFIM,TFIM,TRUE,http://tfimhost1.ibm.com:9080
```

Creating a profile for TFIM V6.1 using the IBM Integration Explorer:

About this task

You can use the IBM Integration Explorer to create a security profile for using TFIM V6.1.

Procedure

1. In the IBM Integration Explorer, right-click on the broker with which you want to work, and click **Properties**.

2. In the Properties window, select the Security tab, and click **Security Profiles**. The Security Profiles window is displayed, containing a list of existing security profiles for the broker on the left and, on the right, a pane in which you can configure the profile.
3. Click **Add** to create a new profile and add it to the list. You can edit the name of the security profile by highlighting it in the list and pressing **F2**.
4. Configure the security profile using the entry fields on the right side of the pane:
 - a. Select the type of **Authentication, Mapping, and Authorization** required. If you select TFIM V6.1 for any of these options, the **TFIM Configuration** field at the bottom of the pane is enabled.
 - b. If you have selected TFIM V6.1 for authentication, mapping or authorization, type the URL of the TFIM server into the **TFIM Configuration** field. The URL that you enter forms a configuration string, which is displayed in one or more of the configuration fields (**Authentication Config, Mapping Config, and Authorization Config**) depending on the entry fields that have TFIM selected.
For more information about the valid values for the configuration parameter, see “Creating a profile using mqsicreateconfigurable-service” on page 306.
 - c. In the **Propagation** field, specify whether you require the identity to be propagated. The default is **False**.
 - d. In the **Password Value** field, select the way in which the password is displayed in the properties folder. The options are:
 - PLAIN** The password is shown in the Properties folder as plain text.
 - OBFUSCATE**
The password is shown in the Properties folder as base64 encoding.
 - MASK** The password is shown in the Properties folder as four asterisks (****).
5. Click **Finish** to deploy the security profile to the broker.

Related concepts:

“Identity” on page 251

In IBM Integration Bus, an identity is a security token that uniquely identifies an individual, or that provides a set of assertions that can be validated.

“Authentication and validation” on page 257

Authentication is the process of establishing the identity of a user or system and verifying that the identity is valid. Applying authentication to a SAML security token involves validating the assertions that it carries and confirming that it is being processed within its validity period.

“Identity mapping” on page 263

Identity mapping is the transformation of a security token from one format to another format, or the federation of an identity from one realm to an equivalent identity in another realm.

“Authorization” on page 261

Authorization is the process of verifying that an identity token has permission to access a message flow.

“Identity and security token propagation” on page 286

Identity and security token propagation enables the identity and security tokens (associated with each message) to be propagated throughout a message flow, and on to target applications through output or request nodes.

“Security profiles” on page 247

A security profile defines the security operations that are to be performed in a message flow at SecurityPEP nodes and security enabled input and output nodes.

Related tasks:

“Configuring the extraction of an identity or security token”

You can configure the SecurityPEP node or security enabled input nodes to extract the identity or security token from a message and store it in the properties tree identity fields, enabling it to be processed throughout the message flow and propagated at output or request nodes.

“Configuring identity authentication and security token validation” on page 312

You can configure a message flow to perform identity authentication or security token validation using Integrated Windows Authentication (IWA), Lightweight Directory Access Protocol (LDAP), a WS-Trust V1.3 compliant security token sever (STS) such as Tivoli Federated Identity Manager (TFIM) Version 6.2, or HTTP requests. Support for TFIM V6.1 is also provided, for compatibility with previous versions of IBM Integration Bus.

“Configuring identity mapping” on page 333

Configure a security token server (STS), such as Tivoli Federated Identity Manager (TFIM) V6.2, to map the incoming security token and, if required, to authenticate and authorize it.

“Creating a security profile” on page 294

You can create a security profile for use with Lightweight Directory Access Protocol (LDAP) or a WS-Trust V1.3 compliant security token server (STS), such as Tivoli Federated Identity Manager (TFIM) V6.2. Support is also provided for TFIM V6.1, for compatibility with previous versions of IBM Integration Bus. You can create the security profile by using either the **mqsicreateconfigurable** command or an editor in the IBM Integration Explorer.

“Configuring authorization” on page 342

Configure the broker to use an external security provider to authorize an identity in a message flow. You can use either Lightweight Directory Access Protocol (LDAP) or a WS-Trust V1.3 compliant security token server (STS) such as Tivoli Federated Identity Manager (TFIM) V6.2. Support for TFIM V6.1 is also provided for compatibility with previous versions.

“Configuring for identity propagation” on page 389

To enable a message flow to perform identity propagation, the input nodes must extract the identity and the output node must propagate it.

“Setting up message flow security” on page 292

Set up security on a message flow to control access based on the identity of a message passing through the message flow.

Related reference:



mqsicreateconfigurable command

Use the **mqsicreateconfigurable** command to create an object name for a broker external resource.

Configuring the extraction of an identity or security token

You can configure the SecurityPEP node or security enabled input nodes to extract the identity or security token from a message and store it in the properties tree identity fields, enabling it to be processed throughout the message flow and propagated at output or request nodes.

Before you begin

Before you start:

Check that an appropriate security profile exists or create a new security profile. See “Creating a security profile” on page 294.

About this task

If an input node or SecurityPEP node is associated with a profile that specifies a security operation (authentication, mapping, or authorization), or specifies propagation as enabled, the node can retrieve an identity or security token from the message bit stream.

- An MQInput node, with the Identity token type security property set to Transport default, retrieves the UserIdentifier element from the message descriptor (MQMD) and puts it into the Identity Source Token element of the Properties folder. At the same time, it sets the Identity Source Type element to username and the Identity Source Issued By element to MQMD.PutAppName (the put application name).
- An HTTPInput node, with the Identity token type security property set to Transport default, retrieves the BasicAuth header from the HTTP request, decodes it, and puts it into the Identity Source Token and Password elements in the Properties folder. At the same time, it sets the Identity Source Type element to username + Password and the Identity Source Issued By element to the HTTP header UserAgent property.
- A SOAPInput node retrieves the appropriate tokens as defined by the configured WS-Security policy sets and bindings, or (if they are not set), the transport binding determines the token type; for example, HTTP transport is BasicAuth. The SOAPInput node then populates the identity source fields in the Properties folder with the retrieved tokens. With a Kerberos policy set and bindings, the token type is a Username containing the Service Principal Name (SPN) from the Kerberos ticket.
- A SecurityPEP node, with the Identity token type property set to Current token, can use the token that has been extracted by an upstream input or SecurityPEP node and stored in the Properties folder.

In some cases, the information extracted from the transport headers is not set or is insufficient to perform authentication or authorization. For example, for authentication to occur, a Username + Password type token is required; however, with WebSphere MQ, only a username is available, which means that the incoming identity has to be trusted. However, you can increase security by applying transport-level security using WebSphere MQ Extended Security Edition.

If the transport header cannot provide the required identity credentials, the information must be provided as part of the body of the incoming message. To enable the identity information to be taken from the body of the message, you must specify the location of the information by using either the **Security** tab on the HTTP, MQ, and SCA input nodes or the **Basic** tab on the SecurityPEP node, or by configuring the required policy set and bindings WS-Security profile on the SOAP node. A SOAP node with a Kerberos policy set and bindings extracts a Username token containing the Service Principal Name (SPN) of the Kerberos ticket.

Procedure

1. In **Identity Token Type**, specify the type of identity token that is in the message. The type can have one of the following values:
 - Transport Default (on the security enabled input nodes)
 - Current token (on the SecurityPEP node)
 - Username

- Username and password
- X.509 Certificate
- SAML assertion
- Kerberos GSS v5 AP_REQ (on the SecurityPEP node)
- LTPA v2 token (on the SecurityPEP node)
- Universal WSSE token (on the SecurityPEP node)

On the security enabled input nodes, the default value is *Transport Default*. On the SecurityPEP node, the default value is *Current token*, which means that the token type that exists in the identity mapped or source field in the Properties folder is used.

2. In **Identity Token Location**, specify the location in the message where the identity is specified. This string is in the form of an ESQL field reference, XPath expression, or string literal, and must resolve to a token with the type Username, Username and password, SAML assertion, Kerberos GSS v5 AP_REQ, LTPA v2 token, or X.509 Certificate. If you use a string literal, it must be enclosed in single quotes and must not contain a period (.).
3. In **Identity Password Location**, enter the location in the message where the password is specified. This string is in the form of an ESQL field reference, XPath expression, or string literal, and must resolve to a string containing a password. If you use a string literal, it must be enclosed in single quotes and must not contain a period (.). This option can be set only if the **Identity Token Type** is set to Username and password.
4. In **Identity IssuedBy Location**, specify a string or path expression to show where (in the message) information about the issuer of the identity is held. This string is in the form of an ESQL field reference, XPath expression, or string literal, defining where the identity was defined. If you use a string literal, it must be enclosed in single quotes and must not contain a period (.).
If you leave this property blank on the security enabled input nodes, the transport header value is used (if there is one). For example, for MQ the MQMD.PutApplName value is used. If you leave this property blank on the SecurityPEP node, the WS-Trust request is sent to the STS without the optional Issuer element in the WS-Trust message.
5. (Optional) Ensure that all input nodes share the same information by promoting the properties to the message flow.

What to do next

To enable the extraction of an identity in a security enabled input node or SecurityPEP node, select a security profile that has at least one security operation configured (authentication, mapping, or authorization) or propagation enabled:

1. In the IBM Integration Toolkit, right-click the BAR file, then click **Open with > Broker Archive Editor**.
2. Click the **Manage and Configure** tab.
3. Click the flow or node on which you want to set the security profile. The properties that you can configure for the message flow or for the node are displayed in the **Properties** view.
4. In the **Security Profile Name** field, select a security profile.
5. Save the BAR file.

Related concepts:

“Identity” on page 251

In IBM Integration Bus, an identity is a security token that uniquely identifies an

individual, or that provides a set of assertions that can be validated.

“Authentication and validation” on page 257

Authentication is the process of establishing the identity of a user or system and verifying that the identity is valid. Applying authentication to a SAML security token involves validating the assertions that it carries and confirming that it is being processed within its validity period.

“Identity mapping” on page 263

Identity mapping is the transformation of a security token from one format to another format, or the federation of an identity from one realm to an equivalent identity in another realm.

“Authorization” on page 261

Authorization is the process of verifying that an identity token has permission to access a message flow.

“Identity and security token propagation” on page 286

Identity and security token propagation enables the identity and security tokens (associated with each message) to be propagated throughout a message flow, and on to target applications through output or request nodes.

“Security profiles” on page 247

A security profile defines the security operations that are to be performed in a message flow at SecurityPEP nodes and security enabled input and output nodes.



Policy sets

Policy sets and bindings define and configure your WS-Security and WS-RM requirements, supported by IBM Integration Bus, for the SOAPInput, SOAPReply, SOAPRequest, SOAPAsyncRequest, and SOAPAsyncResponse nodes.

Related tasks:

“Configuring identity authentication and security token validation” on page 312

You can configure a message flow to perform identity authentication or security token validation using Integrated Windows Authentication (IWA), Lightweight Directory Access Protocol (LDAP), a WS-Trust V1.3 compliant security token server (STS) such as Tivoli Federated Identity Manager (TFIM) Version 6.2, or HTTP requests. Support for TFIM V6.1 is also provided, for compatibility with previous versions of IBM Integration Bus.

“Creating a security profile” on page 294

You can create a security profile for use with Lightweight Directory Access Protocol (LDAP) or a WS-Trust V1.3 compliant security token server (STS), such as Tivoli Federated Identity Manager (TFIM) V6.2. Support is also provided for TFIM V6.1, for compatibility with previous versions of IBM Integration Bus. You can create the security profile by using either the **mqsicreateconfigurable-service** command or an editor in the IBM Integration Explorer.

“Configuring authorization” on page 342

Configure the broker to use an external security provider to authorize an identity in a message flow. You can use either Lightweight Directory Access Protocol (LDAP) or a WS-Trust V1.3 compliant security token server (STS) such as Tivoli Federated Identity Manager (TFIM) V6.2. Support for TFIM V6.1 is also provided for compatibility with previous versions.

“Configuring for identity propagation” on page 389

To enable a message flow to perform identity propagation, the input nodes must extract the identity and the output node must propagate it.

“Setting up message flow security” on page 292

Set up security on a message flow to control access based on the identity of a message passing through the message flow.

Related reference:



MQInput node

Use the MQInput node to receive messages from clients that connect to the broker by using the WebSphere MQ Enterprise Transport, and that use the MQI and AMI application programming interfaces.



HTTPInput node

Use the HTTPInput node to receive an HTTP message from an HTTP client for processing by a message flow.



SOAPInput node

Use the SOAPInput node to process client SOAP messages, so that the broker operates as a SOAP Web Services provider.

Configuring identity authentication and security token validation

You can configure a message flow to perform identity authentication or security token validation using Integrated Windows Authentication (IWA), Lightweight Directory Access Protocol (LDAP), a WS-Trust V1.3 compliant security token server (STS) such as Tivoli Federated Identity Manager (TFIM) Version 6.2, or HTTP requests. Support for TFIM V6.1 is also provided, for compatibility with previous versions of IBM Integration Bus.

Before you begin

Before you start:

Check that an appropriate security profile exists, or create a new security profile. See “Creating a security profile” on page 294.

About this task

For information about configuring authentication, see:

- “Authenticating incoming requests with IWA on Windows” on page 313
- “Authenticating incoming requests with IWA on Linux or UNIX” on page 315
- “Authenticating incoming requests with LDAP” on page 319
- “Authenticating incoming requests with WS-Trust v1.3 STS (TFIM V6.2)” on page 322
- “Authenticating incoming requests with TFIM V6.1” on page 325
- “Providing credentials in HTTP requests” on page 328

Related concepts:

“Identity” on page 251

In IBM Integration Bus, an identity is a security token that uniquely identifies an individual, or that provides a set of assertions that can be validated.

“Authentication and validation” on page 257

Authentication is the process of establishing the identity of a user or system and verifying that the identity is valid. Applying authentication to a SAML security token involves validating the assertions that it carries and confirming that it is being processed within its validity period.

“Security profiles” on page 247

A security profile defines the security operations that are to be performed in a message flow at SecurityPEP nodes and security enabled input and output nodes.

“Security exception processing” on page 290

A security exception is raised when a message flow security failure occurs during security processing in an input node or SecurityPEP node.

Related tasks:

“Setting up message flow security” on page 292

Set up security on a message flow to control access based on the identity of a message passing through the message flow.

Authenticating incoming requests with IWA on Windows:

Set up IBM Integration Bus to use Integrated Windows Authentication (IWA) to secure inbound requests against a broker on Windows.

Before you begin

Securing an IBM Integration Bus service with IWA modifies the behavior of only the HTTPInput and SOAPInput nodes. For inbound support, IWA requires the HTTP and SOAP nodes to use an embedded (integration server) listener. IWA is not supported by broker listeners. SOAP nodes use embedded listeners by default, but HTTP nodes use broker listeners by default. For information on how to switch to an embedded listener, see *Switching from a broker-wide listener to embedded listeners*.

If you are using HTTP over SSL (HTTPS), you must set up a public key infrastructure (PKI). For more information, see “Setting up a public key infrastructure” on page 415.

About this task

Use the following commands to set up and manage inbound support for the NTLM, Kerberos, and SPNEGO protocols, which together are referred to as Integrated Windows Authentication (IWA). When IBM Integration Bus is configured to provide an IWA-secured service, the HTTPInput and SOAPInput nodes accept only incoming requests that can be authenticated against the Windows domain controller, or the Kerberos KDC, as appropriate. Any requests that cannot be authenticated are refused by IBM Integration Bus. By default IWA is disabled.

To enable IWA on a broker running on Windows, run the following command:

```
mqsichangeproperties broker_name -e IntegrationServerName -o ConnectorType -n integratedWindowsAuthentication -v "PropertyValue"
```

Where:

- *broker_name* is the name of the broker you want to modify.
- *IntegrationServerName* is the name of the integration server on that broker.
- *ConnectorType* is *HTTPSConnector* for an SSL connection, or *HTTPConnector* for a non-SSL connection.
- *PropertyValue* is *NTLM*, *Negotiate*, or *Negotiate:Kerberos*. Multiple values can be given, separated by a semicolon or a space, and these values are not case-sensitive. The order in which the values are specified, is the order in which they are returned to the client in the HTTP response. To disable IWA, set this property to a blank string.

NTLM

Specify this value to use the NTLM protocol.

Negotiate

Specify this value to use the Negotiate (SPNEGO) process. This process

allows IBM Integration Bus to negotiate the use of the NTLM or Kerberos protocols. If Kerberos is available, it is the preferred protocol.

Negotiate:Kerberos

Specify this value to use the Negotiate (SPNEGO) process to negotiate only the use of the Kerberos protocol. If the client cannot support the Kerberos protocol, IBM Integration Bus refuses the connection.

You must restart the broker, or reload the integration server, for the command to take effect.

To check what the current IWA setting is, run the following command:

```
mqsireportproperties broker_name -e IntegrationServerName -o ConnectorType -r
```

The following output is displayed within the connector properties:

- integratedWindowsAuthentication='PropertyValue'

Where *PropertyValue* is *NTLM*, *Negotiate*, or *Negotiate:Kerberos*. If multiple values are set, they are separated by a semicolon. If no value is set, IWA is disabled, and the following output is displayed within the connector properties:

- integratedWindowsAuthentication=""

Results

When IBM Integration Bus is configured to provide an IWA-secured service, successfully authenticated messages have the client's identity credentials set in the local environment tree of the message flow. In addition, if the Default Propagation security profile is configured, a subset of these identity credentials are set in the Properties folder of the message tree structure. The following table lists the identity credentials set in the local environment tree of the message flow, and the associated subset of identity credentials set in the Properties folder of the message tree structure:

Table 5. List of identity credentials

Local environment tree credentials	Properties folder credentials
username (root folder)	IdentitySourceType
> fullName (consisting of realm\username)	
> username	IdentitySourceToken
> realm	IdentitySourceIssuedBy
> package	
> spn	
> sid	

Examples

Enable the Negotiate (SPNEGO) protocol for an SSL connection:

```
mqsichangeproperties IB9NODE -e default -o HTTPSConnector  
-n integratedWindowsAuthentication -v "Negotiate"
```

Enable the NTLM and Negotiate (SPNEGO) protocols for a non-SSL connection:

```
mqsichangeproperties IB9NODE -e default -o HTTPConnector  
-n integratedWindowsAuthentication -v "NTLM;Negotiate"
```


Disable all protocols for a non-SSL connection:

```
mqsichangeproperties IB9NODE -e default -o HTTPConnector  
-n integratedWindowsAuthentication -v ""
```

What to do next

Note: If you are authenticating by using the Kerberos protocol, and IBM Integration Bus is receiving messages from a non-Windows client, the client must include the Windows KDC in its Kerberos configuration file (krb5.conf).

Related concepts:

“Integrated Windows Authentication” on page 260

Integrated Windows Authentication (IWA) refers to a set of authentication protocols, NTLM, Kerberos, and SPNEGO, that are used to provide transport-level security. You can configure IBM Integration Bus to provide an IWA-secured service on a broker running on any operating system, and to consume an IWA-secured service on a broker running on Windows, when you are using the HTTP and SOAP nodes.

Related reference:

 HTTPInput node

Use the HTTPInput node to receive an HTTP message from an HTTP client for processing by a message flow.

 SOAPInput node

Use the SOAPInput node to process client SOAP messages, so that the broker operates as a SOAP Web Services provider.

 **mqsichangeproperties** command

Use the **mqsichangeproperties** command to modify broker properties and properties of broker resources.

 **mqsireportproperties** command

Use the **mqsireportproperties** command to display properties that relate to a broker, an integration server, or a configurable service.

Authenticating incoming requests with IWA on Linux or UNIX:

Set up IBM Integration Bus to use Integrated Windows Authentication (IWA) to secure inbound requests against a broker on Linux or UNIX.

Before you begin

Securing an IBM Integration Bus service with IWA modifies the behavior of only the HTTPInput and SOAPInput nodes. For inbound support, IWA requires the HTTP and SOAP nodes to use an embedded (integration server) listener. IWA is not supported by broker listeners. SOAP nodes use embedded listeners by default, but HTTP nodes use broker listeners by default. For information on how to switch to an embedded listener, see [Switching from a broker-wide listener to embedded listeners](#).

If you are using HTTP over SSL (HTTPS), you must set up a public key infrastructure (PKI). For more information, see [“Setting up a public key infrastructure”](#) on page 415.

About this task

Use the following commands to set up and manage inbound support for the Kerberos and SPNEGO protocols, which together are referred to as Integrated Windows Authentication (IWA). When IBM Integration Bus is configured to provide an IWA-secured service, the HTTPInput and SOAPInput nodes accept only incoming requests that can be authenticated against the Kerberos KDC. Any requests that cannot be authenticated are refused by IBM Integration Bus. To configure IBM Integration Bus on Linux or UNIX to use IWA, you must both configure Kerberos on your system, and enable IWA. By default IWA is disabled.

Procedure

1. If your Key Distribution Center (KDC) is on a Windows system, export a keytab that contains the private key of the service principal from the KDC. For example:

```
ktpass -out c:\Windows\krb5.keytab -princ SomePrincipal@YourDomain -crypto RC4-HMAC-NT mapUser Username -pass Password -mapOp set
```

where

out *filename*

Specifies the name and path of the keytab file to be generated.

princ *principal_name*

Specifies the principal name.

crypto *encryption_type*

Specifies the encryption type.

mapUser *username*

Maps the name of a Kerberos principal to a local account.

pass *password*

Specifies the password to use for this principal name.

mapOp *attribute*

Defines how the mapping attribute is set. The attribute alternatives are either add or set.

If your Key Distribution Center (KDC) is on a Linux or UNIX system, refer to the KDC documentation for instructions on creating a keytab for the service principal.

2. Copy the keytab file to the server that hosts the service. You can copy the file to the server by exporting the keytab file and transferring it to the server, for instance by using FTP. The Kerberos configuration file contains a reference to the keytab file in the form of a file URL (such as: /home/user/my.keytab). Because the reference is in the configuration file on the server, the server service can take on the Kerberos principal that is defined in the keytab.
3. Create a Kerberos configuration file that specifies the location of the keytab file on the local workstation.

You can use more than one service principle name per broker per Kerberos realm. Use your workstation default Kerberos configuration file when you are using Kerberos for security. The location for the configuration file differs depending on the system. The usual locations are:

- Linux: /etc/krb5.conf
- UNIX: /etc/krb5/krb5.conf
- z/OS: /krb5/krb5.conf

Different Kerberos configuration files can be configured for use by the broker and integration servers.

The following sample Kerberos configuration file shows typical values for the variables. The variables *default_realm*, *default_keytab_name*, and the names in the

realms are among the values you change in the configuration file, depending on your network and location of the configuration file.

```
[libdefaults]
default_realm = MYREALM.EXAMPLE.COM
default_keytab_name = FILE:c:\Windows\krb5.keytab
default_tkt_enctypes = rc4-hmac
default_tgs_enctypes = rc4-hmac
dns_lookup_realm = false
dns_lookup_kdc = false
ticket_lifetime = 24h
renew_lifetime = 7d
forwardable = true
[realms]
MYREALM.EXAMPLE.COM = {
kdc = kdc.myrealm.example.com
admin_server = kdc.myrealm.example.com
}
```

4. Use one of these **mqsichangeproperties** commands to specify the location of your new configuration file.

- For a broker level Kerberos configuration:

```
mqsichangeproperties brokerName -o BrokerRegistry -n kerberosConfigFile -v kerberosConfigLocation
```

- For an integration server level Kerberos configuration:

```
mqsichangeproperties brokerName -e integrationServerName -o ComIbmJVMManager -n kerberosConfigFile -v kerberosConfigLocation
```

5. To enable IWA on a broker running on Linux or UNIX, run the following command:

```
mqsichangeproperties broker_name -e IntegrationServerName -o ConnectorType
-n integratedWindowsAuthentication -v "PropertyValue"
```

Where:

- *broker_name* is the name of the broker to modify.
- *IntegrationServerName* is the name of the integration server on that broker.
- *ConnectorType* is *HTTPSConnector* for an SSL connection, or *HTTPConnector* for a non-SSL connection.
- *PropertyValue* is *Negotiate:Kerberos*, and is not case-sensitive. To disable IWA, set this property to a blank string.

Negotiate:Kerberos

Specify this value to use the Negotiate (SPNEGO) process to negotiate only the use of the Kerberos protocol. If the client cannot support the Kerberos protocol, IBM Integration Bus refuses the connection.

You must restart the broker, or reload the integration server, for the command to take effect.

To check what the current IWA setting is, run the following command:

```
mqsireportproperties broker_name -e IntegrationServerName -o ConnectorType -r
```

The following output is displayed within the connector properties:

- `integratedWindowsAuthentication='PropertyValue'`

Where *PropertyValue* is *Negotiate:Kerberos*. If no value is set, IWA is disabled, and the following output is displayed within the connector properties:

- `integratedWindowsAuthentication=""`

Results

When IBM Integration Bus is configured to provide an IWA-secured service, successfully authenticated messages will have the client's identity credentials set in the local environment tree of the message flow. In addition, if the Default Propagation security profile is configured, a subset of these identity credentials are set in the Properties folder of the message tree structure. The following table lists the identity credentials set in the local environment tree of the message flow, and the associated subset of identity credentials set in the Properties folder of the message tree structure:

Table 6. List of identity credentials

Local environment tree credentials	Properties folder credentials
username (root folder)	IdentitySourceType
> fullName (consisting of realm\username)	
> username	IdentitySourceToken
> realm	IdentitySourceIssuedBy
> package	
> spn	
> sid	

Examples

Enable the Kerberos protocol for a non-SSL connection:

```
mqsichangeproperties IB9NODE -e default -o HTTPConnector  
-n integratedWindowsAuthentication -v "Negotiate:Kerberos"
```

Disable IWA for an SSL connection:

```
mqsichangeproperties IB9NODE -e default -o HTTPSConnector  
-n integratedWindowsAuthentication -v ""
```

Related concepts:

“Integrated Windows Authentication” on page 260

Integrated Windows Authentication (IWA) refers to a set of authentication protocols, NTLM, Kerberos, and SPNEGO, that are used to provide transport-level security. You can configure IBM Integration Bus to provide an IWA-secured service on a broker running on any operating system, and to consume an IWA-secured service on a broker running on Windows, when you are using the HTTP and SOAP nodes.

Related reference:



HTTPInput node

Use the HTTPInput node to receive an HTTP message from an HTTP client for processing by a message flow.



SOAPInput node

Use the SOAPInput node to process client SOAP messages, so that the broker operates as a SOAP Web Services provider.



mqsichangeproperties command

Use the **mqsichangeproperties** command to modify broker properties and properties of broker resources.



mqsiereportproperties command

Use the **mqsiereportproperties** command to display properties that relate to a broker, an integration server, or a configurable service.

Authenticating incoming requests with LDAP:

Configure a message flow to perform identity authentication using Lightweight Directory Access Protocol (LDAP).

Before you begin

Before you start:

Before you can configure a message flow to perform identity authentication using LDAP, you need to check that an appropriate security profile exists, or create a new security profile. See “Creating a security profile for LDAP” on page 296.

About this task

To authenticate the identity of a user or system, the broker attempts to connect to the LDAP server using the username and password associated with the identity. To do this, the broker needs the following information:

- To resolve the username to an LDAP entry, the broker needs to know the base distinguished name (base DN) of the accepted login IDs. This is required to enable the broker to differentiate between different entries with the same name.
- If the identities do not all have a common base DN, but can be uniquely resolved from a subtree, the DN can be specified in the broker configuration. When a subtree search has been specified, the broker must first connect to the LDAP server and search for the given username in order to obtain the full username distinguished name (DN) to be used for authentication. If your LDAP directory does not permit login of unrecognized IDs, and does not grant search access rights on the subtree, you must set up a separate authorized login ID that the broker can use for the search. Use the **mqsi setdbparms** command to specify a username and password. For example:

```
mqsi setdbparms -n ldap::LDAP -u username -p password
```

or

```
mqsi setdbparms -n ldap::<servername> -u username -p password
```

where *<servername>* is your base LDAP server name, for example, ldap.mydomain.com.

If you specify ldap::LDAP, it creates a default setting for the broker, which the broker attempts to use if you have not explicitly used the **mqsi setdbparms** command to create a login ID for a specific *<servername>*. All servers that do not have an explicit ldap::servername entry then start using the credentials in the ldap::LDAP entry. This means that any servers that were previously using anonymous bind by default will start using the details in ldap::LDAP.

The username that you specify in the **-u** parameter must be recognized by the LDAP server as a complete user name. In most cases this means that you need to specify the full DN of the user. Alternatively, by specifying a username to be anonymous, you can force the broker to bind anonymously to this LDAP server. This might be useful if you have specified a non-anonymous bind as your default (ldap::LDAP). For example:

```
mqsi setdbparms -n ldap::<servername> -u anonymous -p password
```

In this case, the value specified for *password* is ignored.

Steps for enabling LDAP authentication:

Procedure

To enable an existing message flow to perform identity authentication, use the Broker Archive editor to select a security profile that uses LDAP for authentication. You can set a security profile on a message flow or on individual input nodes. If no security profile is set for the input nodes, the setting is inherited from the setting on the message flow.

1. Switch to the Integration Development perspective.
2. In the Application Development view, right-click the BAR file and then click **Open with > Broker Archive Editor**.
3. Click the **Manage and Configure** tab.
4. Click the flow or node on which you want to set the security profile. The properties that you can configure for the message flow or for the node are displayed in the **Properties** view.
5. In the **Security Profile Name** field, select a security profile that uses LDAP for authentication.
6. Save the BAR file.

What to do next

For a SOAPInput node to use the identity in the WS-Security header (rather than an underlying transport identity) an appropriate policy set and bindings must also be defined and specified. For more information, see Policy sets.

If the message identity does not contain enough information for authentication, the information must be taken from the message body. For example, if a password is required for authentication but the message came from WebSphere MQ with only a username, the password information must be taken from the message body. For more information, see “Configuring the extraction of an identity or security token” on page 308.

Related concepts:

“Identity” on page 251

In IBM Integration Bus, an identity is a security token that uniquely identifies an individual, or that provides a set of assertions that can be validated.

“Authentication and validation” on page 257

Authentication is the process of establishing the identity of a user or system and verifying that the identity is valid. Applying authentication to a SAML security token involves validating the assertions that it carries and confirming that it is being processed within its validity period.

“Identity mapping” on page 263

Identity mapping is the transformation of a security token from one format to another format, or the federation of an identity from one realm to an equivalent identity in another realm.

“Authorization” on page 261

Authorization is the process of verifying that an identity token has permission to access a message flow.

“Identity and security token propagation” on page 286

Identity and security token propagation enables the identity and security tokens (associated with each message) to be propagated throughout a message flow, and

on to target applications through output or request nodes.

“Security profiles” on page 247

A security profile defines the security operations that are to be performed in a message flow at SecurityPEP nodes and security enabled input and output nodes.

“Security exception processing” on page 290

A security exception is raised when a message flow security failure occurs during security processing in an input node or SecurityPEP node.

Related tasks:

“Configuring the extraction of an identity or security token” on page 308

You can configure the SecurityPEP node or security enabled input nodes to extract the identity or security token from a message and store it in the properties tree identity fields, enabling it to be processed throughout the message flow and propagated at output or request nodes.

“Configuring identity mapping” on page 333

Configure a security token server (STS), such as Tivoli Federated Identity Manager (TFIM) V6.2, to map the incoming security token and, if required, to authenticate and authorize it.

“Creating a security profile” on page 294

You can create a security profile for use with Lightweight Directory Access Protocol (LDAP) or a WS-Trust V1.3 compliant security token server (STS), such as Tivoli Federated Identity Manager (TFIM) V6.2. Support is also provided for TFIM V6.1, for compatibility with previous versions of IBM Integration Bus. You can create the security profile by using either the **mqscreateconfigurable** command or an editor in the IBM Integration Explorer.

“Configuring authorization” on page 342

Configure the broker to use an external security provider to authorize an identity in a message flow. You can use either Lightweight Directory Access Protocol (LDAP) or a WS-Trust V1.3 compliant security token server (STS) such as Tivoli Federated Identity Manager (TFIM) V6.2. Support for TFIM V6.1 is also provided for compatibility with previous versions.

“Configuring identity authentication and security token validation” on page 312

You can configure a message flow to perform identity authentication or security token validation using Integrated Windows Authentication (IWA), Lightweight Directory Access Protocol (LDAP), a WS-Trust V1.3 compliant security token sever (STS) such as Tivoli Federated Identity Manager (TFIM) Version 6.2, or HTTP requests. Support for TFIM V6.1 is also provided, for compatibility with previous versions of IBM Integration Bus.

“Configuring for identity propagation” on page 389

To enable a message flow to perform identity propagation, the input nodes must extract the identity and the output node must propagate it.

“Setting up message flow security” on page 292

Set up security on a message flow to control access based on the identity of a message passing through the message flow.

Related reference:

 **mqscreateconfigurable** command

Use the **mqscreateconfigurable** command to create an object name for a broker external resource.

 **mqsdeleteconfigurable** command

Use the **mqsdeleteconfigurable** command to delete a configurable service, such as a JMS provider, JDBC provider, or FTP server, that you have created by using the **mqscreateconfigurable** command.

mqsichangeproperties command

Use the **mqsichangeproperties** command to modify broker properties and properties of broker resources.

mqsireportproperties command

Use the **mqsireportproperties** command to display properties that relate to a broker, an integration server, or a configurable service.

MQInput node

Use the MQInput node to receive messages from clients that connect to the broker by using the WebSphere MQ Enterprise Transport, and that use the MQI and AMI application programming interfaces.

HTTPInput node

Use the HTTPInput node to receive an HTTP message from an HTTP client for processing by a message flow.

SOAPInput node

Use the SOAPInput node to process client SOAP messages, so that the broker operates as a SOAP Web Services provider.

SOAPAsyncRequest node

Use the SOAPAsyncRequest node with the SOAPAsyncResponse node to construct a pair of message flows that call a Web service asynchronously.

HTTPRequest node

Use the HTTPRequest node to interact with a web service.

MQOutput node

Use the MQOutput node to send messages to clients that connect to the broker using the WebSphere MQ Enterprise Transport and that use the MQI and AMI application programming interfaces.

Authenticating incoming requests with WS-Trust v1.3 STS (TFIM V6.2):

You can configure supported message flow input nodes or SecurityPEP nodes to perform identity authentication or security token validation using a WS-Trust v1.3 compliant security token server (STS), such as Tivoli Federated Identity Manager (TFIM) V6.2.

Before you begin

Before you start:

Before you can configure identity authentication or token validation, you need to check that an appropriate security profile exists, or create a new security profile. See “Creating a security profile for WS-Trust V1.3 (TFIM V6.2)” on page 302.

About this task

When the security profile is configured to use WS-Trust V1.3 STS for authentication, the broker security manager issues trust requests and processes trust responses according to the WS-Trust V1.3 standard.

When you use a WS-Trust v1.3 STS for authentication, a request is made to the trust service with the following parameters, which control the STS processing. If you are using TFIM V6.2, the following parameters are used in the selection of the TFIM module chain:

- RequestType
- Issuer
- AppliesTo

For more information about these parameters, see: “Authentication, mapping, and authorization with TFIM V6.2 and TAM” on page 280.

The WS-Trust v1.3 specification, published by OASIS, is available at:

<http://docs.oasis-open.org/ws-sx/ws-trust/200512/ws-trust-1.3-os.html>

Steps for enabling WS-Trust v1.3 authentication:

Procedure

To enable an existing message flow to perform authentication or token validation, use the Broker Archive editor to select a security profile that uses a WS-Trust v1.3 STS for authentication and to associate it with the node or message flow. If a security profile is specified on either a message flow or a node, the profile must be available when the message flow is deployed; otherwise, a deployment error occurs.

1. In the Message Broker Toolkit, right-click the BAR file, then click **Open with > Broker Archive Editor**.
2. Click the **Manage and Configure** tab.
3. Click the flow or node on which you want to set the security profile. The properties that you can configure for the message flow or for the node are displayed in the **Properties** view.
4. In the **Security Profile Name** field, select a security profile that configures WS-Trust v1.3 STS for authentication.
5. Save the BAR file.

What to do next

For a SOAPInput node to use the identity in the WS-Security header (rather than an underlying transport identity) an appropriate policy set and bindings must also be defined and specified. For more information, see Policy sets.

If the message identity (or security token) does not contain enough information for authentication, the information must be taken from the message body. For example, if a password is required for authentication but the message came from WebSphere MQ with only a username, the password information must be taken from the message body. For more information, see “Configuring the extraction of an identity or security token” on page 308.

Related concepts:

“Identity” on page 251

In IBM Integration Bus, an identity is a security token that uniquely identifies an individual, or that provides a set of assertions that can be validated.

“Authentication and validation” on page 257

Authentication is the process of establishing the identity of a user or system and verifying that the identity is valid. Applying authentication to a SAML security token involves validating the assertions that it carries and confirming that it is being processed within its validity period.

“Security profiles” on page 247

A security profile defines the security operations that are to be performed in a message flow at SecurityPEP nodes and security enabled input and output nodes.

“Security exception processing” on page 290

A security exception is raised when a message flow security failure occurs during security processing in an input node or SecurityPEP node.

Related tasks:

“Configuring the extraction of an identity or security token” on page 308

You can configure the SecurityPEP node or security enabled input nodes to extract the identity or security token from a message and store it in the properties tree identity fields, enabling it to be processed throughout the message flow and propagated at output or request nodes.

“Creating a security profile” on page 294

You can create a security profile for use with Lightweight Directory Access Protocol (LDAP) or a WS-Trust V1.3 compliant security token server (STS), such as Tivoli Federated Identity Manager (TFIM) V6.2. Support is also provided for TFIM V6.1, for compatibility with previous versions of IBM Integration Bus. You can create the security profile by using either the **mqsicreateconfigurableservice** command or an editor in the IBM Integration Explorer.

“Configuring authorization” on page 342

Configure the broker to use an external security provider to authorize an identity in a message flow. You can use either Lightweight Directory Access Protocol (LDAP) or a WS-Trust V1.3 compliant security token server (STS) such as Tivoli Federated Identity Manager (TFIM) V6.2. Support for TFIM V6.1 is also provided for compatibility with previous versions.

“Configuring identity authentication and security token validation” on page 312

You can configure a message flow to perform identity authentication or security token validation using Integrated Windows Authentication (IWA), Lightweight Directory Access Protocol (LDAP), a WS-Trust V1.3 compliant security token sever (STS) such as Tivoli Federated Identity Manager (TFIM) Version 6.2, or HTTP requests. Support for TFIM V6.1 is also provided, for compatibility with previous versions of IBM Integration Bus.

“Setting up message flow security” on page 292

Set up security on a message flow to control access based on the identity of a message passing through the message flow.

Related reference:



mqsicreateconfigurableservice command

Use the **mqsicreateconfigurableservice** command to create an object name for a broker external resource.



mqsdeleteconfigurableservice command

Use the **mqsdeleteconfigurableservice** command to delete a configurable service, such as a JMS provider, JDBC provider, or FTP server, that you have created by using the **mqsicreateconfigurableservice** command.



mqschangeproperties command

Use the **mqschangeproperties** command to modify broker properties and properties of broker resources.

mqsiereportproperties command

Use the **mqsiereportproperties** command to display properties that relate to a broker, an integration server, or a configurable service.

MQInput node

Use the MQInput node to receive messages from clients that connect to the broker by using the WebSphere MQ Enterprise Transport, and that use the MQI and AMI application programming interfaces.

HTTPInput node

Use the HTTPInput node to receive an HTTP message from an HTTP client for processing by a message flow.

SOAPInput node

Use the SOAPInput node to process client SOAP messages, so that the broker operates as a SOAP Web Services provider.

SOAPAsyncRequest node

Use the SOAPAsyncRequest node with the SOAPAsyncResponse node to construct a pair of message flows that call a Web service asynchronously.

HTTPRequest node

Use the HTTPRequest node to interact with a web service.

MQOutput node

Use the MQOutput node to send messages to clients that connect to the broker using the WebSphere MQ Enterprise Transport and that use the MQI and AMI application programming interfaces.

SecurityPEP node

Use the SecurityPEP node in a message flow to invoke the message flow security manager at any point in the message flow.

Authenticating incoming requests with TFIM V6.1:

You can configure a message flow to perform identity authentication by using Tivoli Federated Identity Manager (TFIM) V6.1.

Before you begin

Before you start:

Before you can configure a message flow to perform identity authentication, you need to check that an appropriate security profile exists, or create a new security profile. See “Creating a security profile for TFIM V6.1” on page 306.

About this task

Note: Support for TFIM V6.1 is included for compatibility with previous versions of IBM Integration Bus. If possible, upgrade to TFIM V6.2 and follow the instructions in “Authenticating incoming requests with WS-Trust v1.3 STS (TFIM V6.2)” on page 322.

When you use TFIM V6.1 for authentication, a request is made to the TFIM trust service with the following three parameters, which select the module chain:

- Issuer = Properties.IdentitySourceIssuedBy

- Applies To = The Fully Qualified Name of the Flow: <Brokername>.<Integration Server Name>.<Message Flow Name>
- Token = Properties.IdentitySourceToken

For more information about these parameters, see “Authentication, mapping, and authorization with TFIM V6.1 and TAM” on page 277.

For further information about how to configure TFIM, see the IBM Tivoli Federated Identity Manager product documentation.

Steps for enabling TFIM authentication:

Procedure

To enable an existing message flow to perform identity authentication, use the Broker Archive editor to select a security profile that uses TFIM for authentication. You can set a security profile on a message flow or on individual input nodes. If no security profile is set for the input nodes, the setting is inherited from the setting on the message flow.

1. Switch to the Integration Development perspective.
2. In the Application Development view, right-click the BAR file, then click **Open with > Broker Archive Editor**.
3. Click the **Manage and Configure** tab.
4. Click the flow or node on which you want to set the security profile. The properties that you can configure for the message flow or for the node are displayed in the **Properties** view.
5. In the **Security Profile Name** field, select a security profile that uses TFIM for authentication.
6. Save the BAR file.

What to do next

For a SOAPInput node to use the identity in the WS-Security header (rather than an underlying transport identity) an appropriate policy set and bindings must also be defined and specified. For more information, see Policy sets.

If the message identity does not contain enough information for authentication, the information must be taken from the message body. For example, if a password is required for authentication but the message came from WebSphere MQ with only a username, the password information must be taken from the message body. For more information, see “Configuring the extraction of an identity or security token” on page 308.

Related concepts:

“Identity” on page 251

In IBM Integration Bus, an identity is a security token that uniquely identifies an individual, or that provides a set of assertions that can be validated.

“Authentication and validation” on page 257

Authentication is the process of establishing the identity of a user or system and verifying that the identity is valid. Applying authentication to a SAML security token involves validating the assertions that it carries and confirming that it is being processed within its validity period.

“Identity mapping” on page 263

Identity mapping is the transformation of a security token from one format to another format, or the federation of an identity from one realm to an equivalent identity in another realm.

“Authorization” on page 261

Authorization is the process of verifying that an identity token has permission to access a message flow.

“Identity and security token propagation” on page 286

Identity and security token propagation enables the identity and security tokens (associated with each message) to be propagated throughout a message flow, and on to target applications through output or request nodes.

“Security profiles” on page 247

A security profile defines the security operations that are to be performed in a message flow at SecurityPEP nodes and security enabled input and output nodes.

“Security exception processing” on page 290

A security exception is raised when a message flow security failure occurs during security processing in an input node or SecurityPEP node.

Related tasks:

“Configuring the extraction of an identity or security token” on page 308

You can configure the SecurityPEP node or security enabled input nodes to extract the identity or security token from a message and store it in the properties tree identity fields, enabling it to be processed throughout the message flow and propagated at output or request nodes.

“Configuring identity mapping” on page 333

Configure a security token server (STS), such as Tivoli Federated Identity Manager (TFIM) V6.2, to map the incoming security token and, if required, to authenticate and authorize it.

“Creating a security profile” on page 294

You can create a security profile for use with Lightweight Directory Access Protocol (LDAP) or a WS-Trust V1.3 compliant security token server (STS), such as Tivoli Federated Identity Manager (TFIM) V6.2. Support is also provided for TFIM V6.1, for compatibility with previous versions of IBM Integration Bus. You can create the security profile by using either the **mqsicreateconfigurable-service** command or an editor in the IBM Integration Explorer.

“Configuring authorization” on page 342

Configure the broker to use an external security provider to authorize an identity in a message flow. You can use either Lightweight Directory Access Protocol (LDAP) or a WS-Trust V1.3 compliant security token server (STS) such as Tivoli Federated Identity Manager (TFIM) V6.2. Support for TFIM V6.1 is also provided for compatibility with previous versions.

“Configuring identity authentication and security token validation” on page 312

You can configure a message flow to perform identity authentication or security token validation using Integrated Windows Authentication (IWA), Lightweight Directory Access Protocol (LDAP), a WS-Trust V1.3 compliant security token sever (STS) such as Tivoli Federated Identity Manager (TFIM) Version 6.2, or HTTP requests. Support for TFIM V6.1 is also provided, for compatibility with previous versions of IBM Integration Bus.

“Configuring for identity propagation” on page 389

To enable a message flow to perform identity propagation, the input nodes must extract the identity and the output node must propagate it.

“Setting up message flow security” on page 292

Set up security on a message flow to control access based on the identity of a message passing through the message flow.

Related reference:



mqsicreateconfigurable-service command

Use the **mqsicreateconfigurable-service** command to create an object name for a broker external resource.



mqsdeleteconfigurable-service command

Use the **mqsdeleteconfigurable-service** command to delete a configurable service, such as a JMS provider, JDBC provider, or FTP server, that you have created by using the **mqsicreateconfigurable-service** command.



mqschange-properties command

Use the **mqschange-properties** command to modify broker properties and properties of broker resources.



mqsireport-properties command

Use the **mqsireport-properties** command to display properties that relate to a broker, an integration server, or a configurable service.



MQInput node

Use the MQInput node to receive messages from clients that connect to the broker by using the WebSphere MQ Enterprise Transport, and that use the MQI and AMI application programming interfaces.



HTTPInput node

Use the HTTPInput node to receive an HTTP message from an HTTP client for processing by a message flow.



SOAPInput node

Use the SOAPInput node to process client SOAP messages, so that the broker operates as a SOAP Web Services provider.



SOAPAsyncRequest node

Use the SOAPAsyncRequest node with the SOAPAsyncResponse node to construct a pair of message flows that call a Web service asynchronously.



HTTPRequest node

Use the HTTPRequest node to interact with a web service.



MQOutput node

Use the MQOutput node to send messages to clients that connect to the broker using the WebSphere MQ Enterprise Transport and that use the MQI and AMI application programming interfaces.

Providing credentials in HTTP requests:

Use a security profile to configure HTTPRequest and SOAPRequest nodes to authenticate with a remote server.

About this task

Basic authentication is a common extension in the HTTP protocol that allows a client to provide identity information to a remote web server in the form of a username and password sent in the HTTP header data. Security profiles in IBM Integration Bus provide a way for message flow designers to provide these credentials without building the HTTP headers in a Compute node.

If identity propagation is enabled for the selected security profile, the HTTPRequest and SOAPRequest nodes automatically pick up username and password credentials, if present, from the Properties tree. See “Configuring for identity propagation” on page 389. The predefined security profile Default Propagation has this setting enabled.

If your broker runs on Windows, you can choose to use additional protocols to pass identity information with the HTTP request, which avoids passing the user and password in the clear. The identity used is obtained from the Properties tree in the same way as basic authentication. These protocols include NT Lan Manager (NTLM), Kerberos, and Simple and Protected Negotiation (SPNEGO), collectively known as Integrated Windows Authentication (IWA), and are controlled at the integration server level. For more information, see “Providing credentials for outbound requests by using IWA” on page 330.

To enable basic authentication, select an appropriate security profile for the output node or the message flow in the Broker Archive editor. The credentials are picked up from the following Properties tree locations if set:

```
Properties.IdentityMappedType  
Properties.IdentityMappedToken  
Properties.IdentityMappedPassword
```

If the mapped identity fields are not set, the credentials are picked up from the following Properties tree locations:

```
Properties.IdentitySourceType  
Properties.IdentitySourceToken  
Properties.IdentitySourcePassword
```

For basic authentication both a username and password are required, therefore the appropriate Type field must be set to the string usernameAndPassword. For example:

```
SET OutputRoot.Properties.IdentitySourceType='usernameAndPassword';  
SET OutputRoot.Properties.IdentitySourceToken = 'myUser';  
SET OutputRoot.Properties.IdentitySourcePassword = 'myPassw0rd';  
SET OutputRoot.Properties.IdentitySourceIssuedBy = 'myDomain';
```

These fields are interpreted by a subsequent HTTPRequest or SOAPRequest node and converted into a basic authentication HTTP header.

You can also propagate credentials from an input message by setting a security profile which includes propagation on an input node, and then using the input node properties Identity token type, Identity Token location and Identity password location. These three properties take an XPath expression that specifies the location in the input message to retrieve the appropriate token or password from. When configured correctly, these properties place the identity information in the Properties.IdentitySourceType, Properties.IdentitySourceToken and Properties.IdentitySourcePassword fields. HTTPRequest or SOAPRequest nodes then use these values directly, with an appropriate security policy.

You can override the configuration of the security profile by selecting the build option **Override configurable property values** in the Broker Archive editor.

Related concepts:

“Identity” on page 251

In IBM Integration Bus, an identity is a security token that uniquely identifies an individual, or that provides a set of assertions that can be validated.

“Authentication and validation” on page 257

Authentication is the process of establishing the identity of a user or system and verifying that the identity is valid. Applying authentication to a SAML security token involves validating the assertions that it carries and confirming that it is being processed within its validity period.

“Identity mapping” on page 263

Identity mapping is the transformation of a security token from one format to another format, or the federation of an identity from one realm to an equivalent identity in another realm.

“Identity and security token propagation” on page 286

Identity and security token propagation enables the identity and security tokens (associated with each message) to be propagated throughout a message flow, and on to target applications through output or request nodes.

“Security profiles” on page 247

A security profile defines the security operations that are to be performed in a message flow at SecurityPEP nodes and security enabled input and output nodes.

Related tasks:

“Configuring for identity propagation” on page 389

To enable a message flow to perform identity propagation, the input nodes must extract the identity and the output node must propagate it.

“Creating a security profile” on page 294

You can create a security profile for use with Lightweight Directory Access Protocol (LDAP) or a WS-Trust V1.3 compliant security token server (STS), such as Tivoli Federated Identity Manager (TFIM) V6.2. Support is also provided for TFIM V6.1, for compatibility with previous versions of IBM Integration Bus. You can create the security profile by using either the `mqsicreateconfigurableservice` command or an editor in the IBM Integration Explorer.

Related reference:



HTTPRequest node

Use the HTTPRequest node to interact with a web service.



SOAPRequest node

Use the SOAPRequest node to send a SOAP request to the remote Web service.



SOAPAsyncRequest node

Use the SOAPAsyncRequest node with the SOAPAsyncResponse node to construct a pair of message flows that call a Web service asynchronously.



Broker Archive editor

Use the Broker Archive editor to create and manage broker archive (BAR) files.

Providing credentials for outbound requests by using IWA:

Set up IBM Integration Bus to consume a remote service that is secured with Integrated Windows Authentication (IWA). Only an IBM Integration Bus running on Windows can consume an IWA-secured service.

Before you begin

Your IBM Integration Bus must be running on the Windows operating system. If IBM Integration Bus is running on a different operating system, an IWA-secured remote service cannot be consumed.

Ensure that you have a security profile specified that is configured for identity propagation. The supplied Default Propagation security profile is sufficient for all authentication types. For more information, see “Creating a security profile” on page 294.

Your message flow must include an HTTPRequest node, or a SOAPRequest node, or both. If your message flow includes an HTTPRequest node, you must set the **HTTP version** property to *1.1* and select *Enable HTTP/1.1 keep-alive* on the **HTTP Settings** tab in the **Properties** view of the node.

A security identity is required for outbound authentication. By default, the identity credentials of the broker user ID (the **serviceUserId** parameter that is specified by the **mqsicreatebroker** command) is sent to the remote service to use for authentication. If you require a specific security identity to be propagated, you must set the appropriate identity credentials in the Properties tree. For more information, see “Providing credentials in HTTP requests” on page 328.

About this task

Use the following commands to set up and manage outbound support for the NTLM, Kerberos, SPNEGO, and SPNEGO-2 protocols, which together are referred to as Integrated Windows Authentication (IWA). By default IWA is not enabled.

To consume a remote service that is secured with IWA, run the following command:

```
mqsichangeproperties broker_name -e IntegrationServerName -o ComIbmSocketConnectionManager  
-n allowedAuthTypes -v "PropertyValue"
```

Where:

- *broker_name* is the name of the broker you want to modify.
- *IntegrationServerName* is the name of the integration server on that broker.
- *PropertyValue* is one of the following values:

IWA Allow the broker to authenticate by using any IWA protocol.

NTLM Allow the broker to authenticate by using the NTLM protocol.

Negotiate

Allow the broker to authenticate by using the SPNEGO process to negotiate the use of the NTLM or Kerberos protocols.

Nego2 Allow the broker to authenticate by using the SPNEGO-2 process to negotiate the use of the NTLM or Kerberos protocols.

Basic Allow authentication with Basic Authentication.

All Allow authentication with any supported protocol from this list.

None Do not authenticate.

Multiple values can be given, separated by a semicolon or a space, and these values are not case-sensitive. IBM Integration Bus selects one value from the list of supported IWA protocols by the server, in the following order: Nego2, Negotiate, NTLM.

When any security protocol other than Basic Authentication is enabled, the HTTPRequest and SOAPRequest nodes do not pre-emptively authenticate to the service. Instead the nodes wait for a 401 response from the server indicating which authentication mechanisms the server supports, and the nodes then use the highest

supported protocol selected in the order listed above. Once connected, this protocol will be used to authenticate pre-emptively until the flow is stopped or the `allowedAuthTypes` property is changed. To configure any of the protocols to be used pre-emptively, run the following command:

```
mqsichangeproperties broker_name -e IntegrationServerName -o ComIbmSocketConnectionManager  
-n preemptiveAuthType -v "PropertyValue"
```

Where:

- *broker_name* is the name of the broker you want to modify.
- *IntegrationServerName* is the name of the integration server on that broker.
- *PropertyValue* is one of the following values:
 - Basic* Pre-emptively authenticate by using Basic Authentication.
 - NTLM* Pre-emptively authenticate by using the NTLM protocol.
 - Negotiate*
Pre-emptively authenticate by using the SPNEGO process to negotiate the use of the NTLM or Kerberos protocols.
 - Nego2* Pre-emptively authenticate by using the SPNEGO-2 process to negotiate the use of the NTLM or Kerberos protocols.

For more advanced scenarios, the following optional configuration properties can also be used with the `ComIbmSocketConnectionManager` object:

allowNtlmNegotiation='TRUE'

Set to 'FALSE' to prevent NTLM from being negotiated with the SPNEGO and SPNEGO-2 protocols. The default value is 'TRUE'.

negotiateMutualAuth='FALSE'

Set to 'TRUE' if you require mutual authentication when the Kerberos protocol is negotiated. The default value is 'FALSE'.

Note: When IBM Integration Bus is authenticating by using Kerberos, the broker automatically generates a service principal name (SPN) for the service that is based on the host name for the request. For example, if the URL for the service is `http://iib.iibservice/testservice/service1.svc` the SPN is assumed to be `HTTP/iib.iibservice`. If the service exists at a different SPN, use the following local environment overrides to provide an explicit SPN for the service:

```
HTTP SET OutputLocalEnvironment.Destination.HTTP.ServicePrincipalName =  
'HTTP/iib.iibservice2.com:7800';
```

```
SOAP SET
```

```
OutputLocalEnvironment.Destination.SOAP.Request.Transport.HTTP.ServicePrincipalName  
= 'HTTP/iib.iibservice2.com:7800';
```

To check the current outbound authentication setting, run the following command:

```
mqsireportproperties broker_name -e IntegrationServerName  
-o ComIbmSocketConnectionManager -r
```

The following output is displayed within the connector properties:

- `allowedAuthTypes='PropertyValue'`

Where *PropertyValue* is *NTLM*, *Negotiate*, *Nego2*, *None*, or *Basic*. If multiple values are set, they are separated by a semicolon.

Examples

Enable all IWA protocols:

```
mqsichangeproperties IB9NODE -e default -o ComIbmSocketConnectionManager  
-n allowedAuthTypes -v "IWA"
```

Enable NTLM and Negotiate (SPNEGO) protocols:

```
mqsichangeproperties IB9NODE -e default -o ComIbmSocketConnectionManager  
-n allowedAuthTypes -v "NTLM;Negotiate"
```

Disable all outbound security protocols:

```
mqsichangeproperties IB9NODE -e default -o ComIbmSocketConnectionManager  
-n allowedAuthTypes -v "None"
```

Related concepts:

“Integrated Windows Authentication” on page 260

Integrated Windows Authentication (IWA) refers to a set of authentication protocols, NTLM, Kerberos, and SPNEGO, that are used to provide transport-level security. You can configure IBM Integration Bus to provide an IWA-secured service on a broker running on any operating system, and to consume an IWA-secured service on a broker running on Windows, when you are using the HTTP and SOAP nodes.

Related reference:

 **mqsichangeproperties** command

Use the **mqsichangeproperties** command to modify broker properties and properties of broker resources.

 **mqsireportproperties** command

Use the **mqsireportproperties** command to display properties that relate to a broker, an integration server, or a configurable service.

 HTTPRequest node

Use the HTTPRequest node to interact with a web service.

 SOAPRequest node

Use the SOAPRequest node to send a SOAP request to the remote Web service.

 **mqsicreatebroker** command - Windows systems

Use the **mqsicreatebroker** command to create a broker on a Windows system.

Configuring identity mapping

Configure a security token server (STS), such as Tivoli Federated Identity Manager (TFIM) V6.2, to map the incoming security token and, if required, to authenticate and authorize it.

Before you begin

Before you start:

Before you can configure a message flow to perform identity mapping, you need to check that an appropriate security profile exists, or create a new security profile. For information about security profiles, see “Creating a security profile” on page 294.

About this task

IBM Integration Bus provides support for identity mapping (also known as identity federation) and token issuance and exchange. Identity mapping is the process of mapping an identity in one realm to another identity in a different realm. For example, you might map *User001* from the eSellers realm to *eSellerUser01* in the eShipping realm. Token issuance and exchange involves the mapping of a token of one type to a token of a different type. For example, an incoming Username and Password token from a client over MQ might be mapped into an equivalent SAML assertion, to be propagated to a Web Services call. Alternatively, you might exchange a SAML 1.1 assertion from a client application for an equivalent SAML 2.0 assertion for an updated backend server.

For information about configuring identity mapping with either a WS-Trust V1.3 STS (for example, TFIM V6.2) or with TFIM V6.1, see:

- “Configuring identity mapping with a WS-Trust V1.3 STS (TFIM V6.2)” on page 336
- “Configuring identity mapping with TFIM V6.1” on page 339

Related concepts:

“Identity” on page 251

In IBM Integration Bus, an identity is a security token that uniquely identifies an individual, or that provides a set of assertions that can be validated.

“Authentication and validation” on page 257

Authentication is the process of establishing the identity of a user or system and verifying that the identity is valid. Applying authentication to a SAML security token involves validating the assertions that it carries and confirming that it is being processed within its validity period.

“Identity mapping” on page 263

Identity mapping is the transformation of a security token from one format to another format, or the federation of an identity from one realm to an equivalent identity in another realm.

“Authorization” on page 261

Authorization is the process of verifying that an identity token has permission to access a message flow.

“Identity and security token propagation” on page 286

Identity and security token propagation enables the identity and security tokens (associated with each message) to be propagated throughout a message flow, and on to target applications through output or request nodes.

“Security profiles” on page 247

A security profile defines the security operations that are to be performed in a message flow at SecurityPEP nodes and security enabled input and output nodes.

“Security exception processing” on page 290

A security exception is raised when a message flow security failure occurs during security processing in an input node or SecurityPEP node.

Related tasks:

“Configuring the extraction of an identity or security token” on page 308

You can configure the SecurityPEP node or security enabled input nodes to extract the identity or security token from a message and store it in the properties tree identity fields, enabling it to be processed throughout the message flow and propagated at output or request nodes.

“Configuring identity authentication and security token validation” on page 312
You can configure a message flow to perform identity authentication or security token validation using Integrated Windows Authentication (IWA), Lightweight Directory Access Protocol (LDAP), a WS-Trust V1.3 compliant security token server (STS) such as Tivoli Federated Identity Manager (TFIM) Version 6.2, or HTTP requests. Support for TFIM V6.1 is also provided, for compatibility with previous versions of IBM Integration Bus.

“Creating a security profile” on page 294

You can create a security profile for use with Lightweight Directory Access Protocol (LDAP) or a WS-Trust V1.3 compliant security token server (STS), such as Tivoli Federated Identity Manager (TFIM) V6.2. Support is also provided for TFIM V6.1, for compatibility with previous versions of IBM Integration Bus. You can create the security profile by using either the **mqsicreateconfigurableservice** command or an editor in the IBM Integration Explorer.

“Configuring authorization” on page 342

Configure the broker to use an external security provider to authorize an identity in a message flow. You can use either Lightweight Directory Access Protocol (LDAP) or a WS-Trust V1.3 compliant security token server (STS) such as Tivoli Federated Identity Manager (TFIM) V6.2. Support for TFIM V6.1 is also provided for compatibility with previous versions.

“Configuring for identity propagation” on page 389

To enable a message flow to perform identity propagation, the input nodes must extract the identity and the output node must propagate it.

“Setting up message flow security” on page 292

Set up security on a message flow to control access based on the identity of a message passing through the message flow.

Related reference:



MQInput node

Use the MQInput node to receive messages from clients that connect to the broker by using the WebSphere MQ Enterprise Transport, and that use the MQI and AMI application programming interfaces.



HTTPInput node

Use the HTTPInput node to receive an HTTP message from an HTTP client for processing by a message flow.



SOAPInput node

Use the SOAPInput node to process client SOAP messages, so that the broker operates as a SOAP Web Services provider.



SOAPAsyncRequest node

Use the SOAPAsyncRequest node with the SOAPAsyncResponse node to construct a pair of message flows that call a Web service asynchronously.



HTTPRequest node

Use the HTTPRequest node to interact with a web service.



MQOutput node

Use the MQOutput node to send messages to clients that connect to the broker using the WebSphere MQ Enterprise Transport and that use the MQI and AMI application programming interfaces.

Configuring identity mapping with a WS-Trust V1.3 STS (TFIM V6.2):

Configure a WS-Trust V1.3 compliant security token server (STS), such as Tivoli Federated Identity Manager (TFIM) V6.2, to map the incoming security token and, if required, to authenticate and authorize it.

Before you begin

Before you start:

Before you can configure a message flow to perform identity mapping, you need to check that an appropriate security profile exists, or create a new security profile. For information about security profiles, see “Creating a security profile” on page 294.

About this task

To configure TFIM V6.2 to map an incoming security token, you must create a custom module chain in TFIM, which performs the security operations. The TFIM configuration controls the token type that is returned from the mapping.

When you use a WS-Trust V1.3 STS for identity mapping, a request is made to the security token server with the following parameters, which control the STS processing:

- RequestType
- Issuer
- AppliesTo

If you are using TFIM V6.2, these parameters are used in the selection of the module chain.

The security manager invokes the WS-Trust v1.3 provider only once, even if it is set for additional security operations (such as authentication or authorization). As a result, when you are using TFIM V6.2, you must configure a single module chain to perform all the required authentication, mapping, and authorization operations.

When the security profile includes a mapping operation, the STS (for example, TFIM V6.2) must return a security token in its response. The STS can return the original unmodified token if no token exchange is required.

For more information about these parameters, see: “Authentication, mapping, and authorization with TFIM V6.2 and TAM” on page 280.

The WS-Trust v1.3 specification, published by OASIS, is available at:
<http://docs.oasis-open.org/ws-sx/ws-trust/200512/ws-trust-1.3-os.html>

For information on how to configure TFIM, see the IBM Tivoli Federated Identity Manager product documentation.

Follow these steps to enable an existing message flow to perform identity mapping.

Procedure

Using the Broker Archive editor, select a security profile that has mapping enabled. You can set a security profile on a message flow or on individual input nodes. If no security profile is set for the input nodes, the setting is inherited from the setting on the message flow.

1. In the Message Broker Toolkit, right-click the BAR file, then click **Open with > Broker Archive Editor**.
2. Click the **Manage and Configure** tab.
3. Click the message flow or node on which you want to set the security profile. The properties that you can configure for the message flow or for the node are displayed in the **Properties** view.
4. In the **Security Profile Name** field, enter the name of a security profile that has mapping enabled.
5. Save the BAR file.

Related concepts:

“Identity” on page 251

In IBM Integration Bus, an identity is a security token that uniquely identifies an individual, or that provides a set of assertions that can be validated.

“Authentication and validation” on page 257

Authentication is the process of establishing the identity of a user or system and verifying that the identity is valid. Applying authentication to a SAML security token involves validating the assertions that it carries and confirming that it is being processed within its validity period.

“Identity mapping” on page 263

Identity mapping is the transformation of a security token from one format to another format, or the federation of an identity from one realm to an equivalent identity in another realm.

“Authorization” on page 261

Authorization is the process of verifying that an identity token has permission to access a message flow.

“Identity and security token propagation” on page 286

Identity and security token propagation enables the identity and security tokens (associated with each message) to be propagated throughout a message flow, and on to target applications through output or request nodes.

“Security profiles” on page 247

A security profile defines the security operations that are to be performed in a message flow at SecurityPEP nodes and security enabled input and output nodes.

“Security exception processing” on page 290

A security exception is raised when a message flow security failure occurs during security processing in an input node or SecurityPEP node.

Related tasks:

“Configuring the extraction of an identity or security token” on page 308

You can configure the SecurityPEP node or security enabled input nodes to extract the identity or security token from a message and store it in the properties tree identity fields, enabling it to be processed throughout the message flow and propagated at output or request nodes.

“Configuring identity authentication and security token validation” on page 312
You can configure a message flow to perform identity authentication or security token validation using Integrated Windows Authentication (IWA), Lightweight Directory Access Protocol (LDAP), a WS-Trust V1.3 compliant security token server (STS) such as Tivoli Federated Identity Manager (TFIM) Version 6.2, or HTTP requests. Support for TFIM V6.1 is also provided, for compatibility with previous versions of IBM Integration Bus.

“Creating a security profile” on page 294

You can create a security profile for use with Lightweight Directory Access Protocol (LDAP) or a WS-Trust V1.3 compliant security token server (STS), such as Tivoli Federated Identity Manager (TFIM) V6.2. Support is also provided for TFIM V6.1, for compatibility with previous versions of IBM Integration Bus. You can create the security profile by using either the **mqsicreateconfigurable** service command or an editor in the IBM Integration Explorer.

“Configuring authorization” on page 342

Configure the broker to use an external security provider to authorize an identity in a message flow. You can use either Lightweight Directory Access Protocol (LDAP) or a WS-Trust V1.3 compliant security token server (STS) such as Tivoli Federated Identity Manager (TFIM) V6.2. Support for TFIM V6.1 is also provided for compatibility with previous versions.

“Configuring for identity propagation” on page 389

To enable a message flow to perform identity propagation, the input nodes must extract the identity and the output node must propagate it.

“Setting up message flow security” on page 292

Set up security on a message flow to control access based on the identity of a message passing through the message flow.

Related reference:



MQInput node

Use the MQInput node to receive messages from clients that connect to the broker by using the WebSphere MQ Enterprise Transport, and that use the MQI and AMI application programming interfaces.



HTTPInput node

Use the HTTPInput node to receive an HTTP message from an HTTP client for processing by a message flow.



SOAPInput node

Use the SOAPInput node to process client SOAP messages, so that the broker operates as a SOAP Web Services provider.



SOAPAsyncRequest node

Use the SOAPAsyncRequest node with the SOAPAsyncResponse node to construct a pair of message flows that call a Web service asynchronously.



HTTPRequest node

Use the HTTPRequest node to interact with a web service.



MQOutput node

Use the MQOutput node to send messages to clients that connect to the broker using the WebSphere MQ Enterprise Transport and that use the MQI and AMI application programming interfaces.

Configuring identity mapping with TFIM V6.1:

Configure Tivoli Federated Identity Manager (TFIM) V6.1 to map the incoming security token and, if required, to authenticate and authorize it.

Before you begin

Before you start:

Before you can configure a message flow to perform identity mapping, you need to check that an appropriate security profile exists, or create a new security profile. For information about security profiles, see “Creating a security profile” on page 294.

About this task

Note: Support for TFIM V6.1 is included for compatibility with previous versions of IBM Integration Bus. If possible, upgrade to TFIM V6.2 and follow the instructions in “Configuring identity mapping with a WS-Trust V1.3 STS (TFIM V6.2)” on page 336.

To configure TFIM V6.1 to map the incoming security token, you need to create a custom module chain in TFIM, which performs the security operations. The TFIM configuration controls the token type that is returned from the mapping.

When you use TFIM for mapping, a request is made to the TFIM trust service with the following three parameters, which select the module chain:

- Issuer = Properties.IdentitySourceIssuedBy
- AppliesTo = The fully qualified name of the flow: *Brokername.Integration Server Name.Message Flow Name*
- Token = Properties.IdentitySourceToken

The security manager invokes the security provider only once, even if it is set for additional security operations (such as authentication or authorization). As a result, when you are using TFIM V6.1, you must configure a single module chain to perform all the required authentication, mapping, and authorization operations.

For information on how to configure TFIM, see the IBM Tivoli Federated Identity Manager product documentation.

Follow these steps to enable an existing message flow to perform identity mapping.

Procedure

Using the Broker Archive editor, select a security profile that has mapping enabled. You can set a security profile on a message flow or on individual input nodes. If no security profile is set for the input nodes, the setting is inherited from the setting on the message flow.

1. In the Message Broker Toolkit, right-click the BAR file, then click **Open with > Broker Archive Editor**.
2. Click the **Manage and Configure** tab.

3. Click the flow or node on which you want to set the security profile. The properties that you can configure for the message flow or for the node are displayed in the **Properties** view.
4. In the **Security Profile Name** field, enter the name of a security profile that has mapping enabled.
5. Save the BAR file.

Related concepts:

“Identity” on page 251

In IBM Integration Bus, an identity is a security token that uniquely identifies an individual, or that provides a set of assertions that can be validated.

“Authentication and validation” on page 257

Authentication is the process of establishing the identity of a user or system and verifying that the identity is valid. Applying authentication to a SAML security token involves validating the assertions that it carries and confirming that it is being processed within its validity period.

“Identity mapping” on page 263

Identity mapping is the transformation of a security token from one format to another format, or the federation of an identity from one realm to an equivalent identity in another realm.

“Authorization” on page 261

Authorization is the process of verifying that an identity token has permission to access a message flow.

“Identity and security token propagation” on page 286

Identity and security token propagation enables the identity and security tokens (associated with each message) to be propagated throughout a message flow, and on to target applications through output or request nodes.

“Security profiles” on page 247

A security profile defines the security operations that are to be performed in a message flow at SecurityPEP nodes and security enabled input and output nodes.

“Security exception processing” on page 290

A security exception is raised when a message flow security failure occurs during security processing in an input node or SecurityPEP node.

Related tasks:

“Configuring the extraction of an identity or security token” on page 308

You can configure the SecurityPEP node or security enabled input nodes to extract the identity or security token from a message and store it in the properties tree identity fields, enabling it to be processed throughout the message flow and propagated at output or request nodes.

“Configuring identity authentication and security token validation” on page 312

You can configure a message flow to perform identity authentication or security token validation using Integrated Windows Authentication (IWA), Lightweight Directory Access Protocol (LDAP), a WS-Trust V1.3 compliant security token server (STS) such as Tivoli Federated Identity Manager (TFIM) Version 6.2, or HTTP requests. Support for TFIM V6.1 is also provided, for compatibility with previous versions of IBM Integration Bus.

“Creating a security profile” on page 294

You can create a security profile for use with Lightweight Directory Access Protocol (LDAP) or a WS-Trust V1.3 compliant security token server (STS), such as Tivoli Federated Identity Manager (TFIM) V6.2. Support is also provided for TFIM V6.1, for compatibility with previous versions of IBM Integration Bus. You can create the security profile by using either the **mqsicreateconfigurableservice** command or an editor in the IBM Integration Explorer.

“Creating a security profile for LDAP” on page 296

Create a security profile for use with Lightweight Directory Access Protocol (LDAP) or Secure LDAP (LDAPS), by using either the **mqsicreateconfigurable-service** command or an editor in the IBM Integration Explorer.

“Creating a security profile for TFIM V6.1” on page 306

You can create a security profile for Tivoli Federated Identity Manager (TFIM) V6.1 for any combination of the following functions: authentication, authorization, and mapping. You can use either the **mqsicreateconfigurable-service** command or an editor in the IBM Integration Explorer to create the security profile.

“Configuring authorization” on page 342

Configure the broker to use an external security provider to authorize an identity in a message flow. You can use either Lightweight Directory Access Protocol (LDAP) or a WS-Trust V1.3 compliant security token server (STS) such as Tivoli Federated Identity Manager (TFIM) V6.2. Support for TFIM V6.1 is also provided for compatibility with previous versions.

“Configuring for identity propagation” on page 389

To enable a message flow to perform identity propagation, the input nodes must extract the identity and the output node must propagate it.

“Setting up message flow security” on page 292

Set up security on a message flow to control access based on the identity of a message passing through the message flow.

Related reference:



mqsicreateconfigurable-service command

Use the **mqsicreateconfigurable-service** command to create an object name for a broker external resource.



mqsideleteconfigurable-service command

Use the **mqsideleteconfigurable-service** command to delete a configurable service, such as a JMS provider, JDBC provider, or FTP server, that you have created by using the **mqsicreateconfigurable-service** command.



mqsichangeproperties command

Use the **mqsichangeproperties** command to modify broker properties and properties of broker resources.



mqsireportproperties command

Use the **mqsireportproperties** command to display properties that relate to a broker, an integration server, or a configurable service.



MQInput node

Use the MQInput node to receive messages from clients that connect to the broker by using the WebSphere MQ Enterprise Transport, and that use the MQI and AMI application programming interfaces.



HTTPInput node

Use the HTTPInput node to receive an HTTP message from an HTTP client for processing by a message flow.



SOAPInput node

Use the SOAPInput node to process client SOAP messages, so that the broker operates as a SOAP Web Services provider.



SOAPAsyncRequest node

Use the SOAPAsyncRequest node with the SOAPAsyncResponse node to construct a pair of message flows that call a Web service asynchronously.



HTTPRequest node

Use the HTTPRequest node to interact with a web service.



MQOutput node

Use the MQOutput node to send messages to clients that connect to the broker using the WebSphere MQ Enterprise Transport and that use the MQI and AMI application programming interfaces.

Configuring authorization

Configure the broker to use an external security provider to authorize an identity in a message flow. You can use either Lightweight Directory Access Protocol (LDAP) or a WS-Trust V1.3 compliant security token server (STS) such as Tivoli Federated Identity Manager (TFIM) V6.2. Support for TFIM V6.1 is also provided for compatibility with previous versions.

Before you begin

Before you start:

Check that an appropriate security profile exists, or create a new security profile. See “Creating a security profile” on page 294.

About this task

For information about configuring authorization for LDAP, WS-Trust V1.3 (TFIM V6.2), or TFIM V6.1, see:

- “Configuring authorization with LDAP” on page 343
- “Configuring authorization with a WS-Trust v1.3 STS (TFIM V6.2)” on page 347
- “Configuring authorization with TFIM V6.1” on page 355

What to do next

Related concepts:

“Identity” on page 251

In IBM Integration Bus, an identity is a security token that uniquely identifies an individual, or that provides a set of assertions that can be validated.

“Authorization” on page 261

Authorization is the process of verifying that an identity token has permission to access a message flow.

“Security profiles” on page 247

A security profile defines the security operations that are to be performed in a message flow at SecurityPEP nodes and security enabled input and output nodes.

Related tasks:

“Configuring identity authentication and security token validation” on page 312

You can configure a message flow to perform identity authentication or security token validation using Integrated Windows Authentication (IWA), Lightweight Directory Access Protocol (LDAP), a WS-Trust V1.3 compliant security token sever (STS) such as Tivoli Federated Identity Manager (TFIM) Version 6.2, or HTTP requests. Support for TFIM V6.1 is also provided, for compatibility with previous versions of IBM Integration Bus.

“Configuring authorization with LDAP”

This topic describes how to configure a message flow to perform authorization on an identity using Lightweight Directory Access Protocol (LDAP).

“Configuring authorization with a WS-Trust v1.3 STS (TFIM V6.2)” on page 347

You can configure supported message flow input nodes or SecurityPEP nodes to perform authorization of an identity or security token by using a WS-Trust v1.3 compliant security token server (STS), such as Tivoli Federated Identity Manager (TFIM) V6.2.

“Configuring authorization with TFIM V6.1” on page 355

You can configure a message flow to perform authorization on an identity by using Tivoli Federated Identity Manager (TFIM) V6.1.

“Configuring identity mapping” on page 333

Configure a security token server (STS), such as Tivoli Federated Identity Manager (TFIM) V6.2, to map the incoming security token and, if required, to authenticate and authorize it.

“Creating a security profile” on page 294

You can create a security profile for use with Lightweight Directory Access Protocol (LDAP) or a WS-Trust V1.3 compliant security token server (STS), such as Tivoli Federated Identity Manager (TFIM) V6.2. Support is also provided for TFIM V6.1, for compatibility with previous versions of IBM Integration Bus. You can create the security profile by using either the `mqs:createconfigurable` command or an editor in the IBM Integration Explorer.

“Configuring for identity propagation” on page 389

To enable a message flow to perform identity propagation, the input nodes must extract the identity and the output node must propagate it.

“Setting up message flow security” on page 292

Set up security on a message flow to control access based on the identity of a message passing through the message flow.

Configuring authorization with LDAP:

This topic describes how to configure a message flow to perform authorization on an identity using Lightweight Directory Access Protocol (LDAP).

Before you begin

Before you start:

Before you can configure a message flow to perform authorization, you need to check that an appropriate security profile exists, or create a new security profile. See “Creating a security profile for LDAP” on page 296.

About this task

When LDAP is used for authorization, the broker needs to determine whether the incoming username is a member of the given group. To do this, the broker requires the following information:

- To resolve the username to an LDAP entry, the broker needs to know the base distinguished name (Base DN) of the accepted login IDs. This is required to enable the broker to differentiate between different entries with the same name.
- To get an entry list from a group name, the group name must be the distinguished name of the group, not just a common name. An LDAP search is made for the group, and the username is checked by finding an entry matching the distinguished name of the user.

- If your LDAP directory does not permit login by unrecognized IDs, and does not grant search access rights on the subtree, you must set up a separate authorized login ID that the broker can use for the search. Use the **mqsisetdbparms** command to specify a username and password:

```
mqsisetdbparms -n ldap::LDAP -u username -p password
```

or

```
mqsisetdbparms -n ldap::<servername> -u username -p password
```

where *<servername>* is your base LDAP server name. For example:
ldap.mydomain.com.

If you specify ldap::LDAP, it creates a default setting for the broker, which the broker attempts to use if you have not explicitly used the **mqsisetdbparms** command to create a login ID for a specific *<servername>*. All servers that do not have an explicit ldap::servername entry then start using the credentials in the ldap::LDAP entry. This means that any servers that were previously using anonymous bind by default will start using the details in ldap::LDAP.

The username that you specify in the **-u** parameter must be recognized by the LDAP server as a complete user name. In most cases this means that you need to specify the full DN of the user. Alternatively, by specifying a username to be anonymous, you can force the broker to bind anonymously to this LDAP server. This might be useful if you have specified a non-anonymous bind as your default (ldap::LDAP). For example:

```
mqsisetdbparms -n ldap::<servername> -u anonymous -p password
```

In this case, the value specified for *password* is ignored.

Steps for enabling LDAP authorization:

Procedure

To enable an existing message flow to perform authorization using LDAP, use the Broker Archive editor to select a security profile that has authorization enabled. You can set a security profile on a message flow or on individual input nodes. If no security profile is set for the input nodes, the setting is inherited from the setting on the message flow.

1. Switch to the Integration Development perspective.
2. In the Application Development view, right-click the BAR file and then click **Open with > Broker Archive Editor**.
3. Click the **Manage and Configure** tab.
4. Click the flow or node on which you want to set the security profile. The properties that you can configure for the message flow or for the node are displayed in the **Properties** view.
5. In the **Security Profile Name** field, select a security profile that uses LDAP for authorization.
6. Save the BAR file.

What to do next

For a SOAPInput node to use the identity in the WS-Security header (rather than an underlying transport identity) an appropriate policy set and bindings must also be defined and specified. For more information, see Policy sets.

Related concepts:

“Identity” on page 251

In IBM Integration Bus, an identity is a security token that uniquely identifies an individual, or that provides a set of assertions that can be validated.

“Authentication and validation” on page 257

Authentication is the process of establishing the identity of a user or system and verifying that the identity is valid. Applying authentication to a SAML security token involves validating the assertions that it carries and confirming that it is being processed within its validity period.

“Identity mapping” on page 263

Identity mapping is the transformation of a security token from one format to another format, or the federation of an identity from one realm to an equivalent identity in another realm.

“Authorization” on page 261

Authorization is the process of verifying that an identity token has permission to access a message flow.

“Identity and security token propagation” on page 286

Identity and security token propagation enables the identity and security tokens (associated with each message) to be propagated throughout a message flow, and on to target applications through output or request nodes.

“Security profiles” on page 247

A security profile defines the security operations that are to be performed in a message flow at SecurityPEP nodes and security enabled input and output nodes.

“Security exception processing” on page 290

A security exception is raised when a message flow security failure occurs during security processing in an input node or SecurityPEP node.

Related tasks:

“Configuring the extraction of an identity or security token” on page 308

You can configure the SecurityPEP node or security enabled input nodes to extract the identity or security token from a message and store it in the properties tree identity fields, enabling it to be processed throughout the message flow and propagated at output or request nodes.

“Configuring identity authentication and security token validation” on page 312

You can configure a message flow to perform identity authentication or security token validation using Integrated Windows Authentication (IWA), Lightweight Directory Access Protocol (LDAP), a WS-Trust V1.3 compliant security token server (STS) such as Tivoli Federated Identity Manager (TFIM) Version 6.2, or HTTP requests. Support for TFIM V6.1 is also provided, for compatibility with previous versions of IBM Integration Bus.

“Configuring authorization” on page 342

Configure the broker to use an external security provider to authorize an identity in a message flow. You can use either Lightweight Directory Access Protocol (LDAP) or a WS-Trust V1.3 compliant security token server (STS) such as Tivoli Federated Identity Manager (TFIM) V6.2. Support for TFIM V6.1 is also provided for compatibility with previous versions.

“Configuring identity mapping” on page 333

Configure a security token server (STS), such as Tivoli Federated Identity Manager (TFIM) V6.2, to map the incoming security token and, if required, to authenticate and authorize it.

“Creating a security profile” on page 294

You can create a security profile for use with Lightweight Directory Access Protocol (LDAP) or a WS-Trust V1.3 compliant security token server (STS), such as Tivoli Federated Identity Manager (TFIM) V6.2. Support is also provided for TFIM V6.1, for compatibility with previous versions of IBM Integration Bus. You can create the security profile by using either the **mqscreateconfigurable-service** command or an editor in the IBM Integration Explorer.

“Creating a security profile for LDAP” on page 296

Create a security profile for use with Lightweight Directory Access Protocol (LDAP) or Secure LDAP (LDAPS), by using either the **mqscreateconfigurable-service** command or an editor in the IBM Integration Explorer.

“Configuring for identity propagation” on page 389

To enable a message flow to perform identity propagation, the input nodes must extract the identity and the output node must propagate it.

“Setting up message flow security” on page 292

Set up security on a message flow to control access based on the identity of a message passing through the message flow.

Related reference:

 **mqscreateconfigurable-service** command

Use the **mqscreateconfigurable-service** command to create an object name for a broker external resource.

 **mqsdeleteconfigurable-service** command

Use the **mqsdeleteconfigurable-service** command to delete a configurable service, such as a JMS provider, JDBC provider, or FTP server, that you have created by using the **mqscreateconfigurable-service** command.

 **mqschange-properties** command

Use the **mqschange-properties** command to modify broker properties and properties of broker resources.

 **mqsreport-properties** command

Use the **mqsreport-properties** command to display properties that relate to a broker, an integration server, or a configurable service.

 MQInput node

Use the MQInput node to receive messages from clients that connect to the broker by using the WebSphere MQ Enterprise Transport, and that use the MQI and AMI application programming interfaces.

 HTTPInput node

Use the HTTPInput node to receive an HTTP message from an HTTP client for processing by a message flow.

 SOAPInput node

Use the SOAPInput node to process client SOAP messages, so that the broker operates as a SOAP Web Services provider.

 SOAPAsyncRequest node

Use the SOAPAsyncRequest node with the SOAPAsyncResponse node to construct a pair of message flows that call a Web service asynchronously.

 HTTPRequest node

Use the HTTPRequest node to interact with a web service.



MQOutput node

Use the MQOutput node to send messages to clients that connect to the broker using the WebSphere MQ Enterprise Transport and that use the MQI and AMI application programming interfaces.

Configuring authorization with a WS-Trust v1.3 STS (TFIM V6.2):

You can configure supported message flow input nodes or SecurityPEP nodes to perform authorization of an identity or security token by using a WS-Trust v1.3 compliant security token server (STS), such as Tivoli Federated Identity Manager (TFIM) V6.2.

Before you begin

Before you start:

Before you configure a message flow to perform authorization with a WS-Trust v1.3 STS:

- Check that an appropriate security profile exists, or create a new security profile. See “Creating a security profile for WS-Trust V1.3 (TFIM V6.2)” on page 302.

About this task

The message flow security manager issues an authorization request to the WS-Trust service with the following parameters, which select the TFIM module chain to be used:

- RequestType
- Issuer
- AppliesTo

For more information about these parameters, see: “Authentication, mapping, and authorization with TFIM V6.2 and TAM” on page 280.

In addition to configuring Message Broker to perform authorization with a WS-Trust compliant STS, you must configure TAM. For information about how to do this, see the following topics:

- “Creating a module chain in TFIM V6.2” on page 350
- “Configuring TAM for authorization using TFIM V6.2” on page 352

Steps for enabling authorization using a WS-Trust v1.3 STS provider:

Procedure

To enable an existing message flow to enforce authorization using a WS-Trust v1.3 STS provider, use the Broker Archive editor to select a security profile that has authorization set for that provider. You can set a security profile on a message flow or on individual input nodes or SecurityPEP nodes. If you leave the **Security Profile** property blank, the node inherits the **Security Profile** property that is set at the message flow level. If you leave the property blank at both levels, security is turned off for the node.

1. In the Message Broker Toolkit, right-click the BAR file, then click **Open with > Broker Archive Editor**.
2. Click the **Manage and Configure** tab.

3. Click the flow or node on which you want to set the security profile. The properties that you can configure for the message flow or for the node are displayed in the **Properties** view.
4. In the **Security Profile Name** field, select a security profile that has authorization set for *WS-Trust V1.3 STS*.
5. Save the BAR file.

What to do next

For a SOAPInput node to use the token in the WS-Security header (rather than an underlying transport identity) an appropriate policy set and bindings must also be defined and specified. For more information, see Policy sets.

The WS-Trust v1.3 specification is available at: <http://docs.oasis-open.org/ws-sx/ws-trust/200512/ws-trust-1.3-os.html>.

Related concepts:

“Identity” on page 251

In IBM Integration Bus, an identity is a security token that uniquely identifies an individual, or that provides a set of assertions that can be validated.

“Authentication and validation” on page 257

Authentication is the process of establishing the identity of a user or system and verifying that the identity is valid. Applying authentication to a SAML security token involves validating the assertions that it carries and confirming that it is being processed within its validity period.

“Identity mapping” on page 263

Identity mapping is the transformation of a security token from one format to another format, or the federation of an identity from one realm to an equivalent identity in another realm.

“Authorization” on page 261

Authorization is the process of verifying that an identity token has permission to access a message flow.

“Identity and security token propagation” on page 286

Identity and security token propagation enables the identity and security tokens (associated with each message) to be propagated throughout a message flow, and on to target applications through output or request nodes.

“Security profiles” on page 247

A security profile defines the security operations that are to be performed in a message flow at SecurityPEP nodes and security enabled input and output nodes.

“Security exception processing” on page 290

A security exception is raised when a message flow security failure occurs during security processing in an input node or SecurityPEP node.

Related tasks:

“Configuring the extraction of an identity or security token” on page 308

You can configure the SecurityPEP node or security enabled input nodes to extract the identity or security token from a message and store it in the properties tree identity fields, enabling it to be processed throughout the message flow and propagated at output or request nodes.

“Configuring identity authentication and security token validation” on page 312

You can configure a message flow to perform identity authentication or security token validation using Integrated Windows Authentication (IWA), Lightweight Directory Access Protocol (LDAP), a WS-Trust V1.3 compliant security token sever (STS) such as Tivoli Federated Identity Manager (TFIM) Version 6.2, or HTTP requests. Support for TFIM V6.1 is also provided, for compatibility with previous

versions of IBM Integration Bus.

“Configuring identity mapping” on page 333

Configure a security token server (STS), such as Tivoli Federated Identity Manager (TFIM) V6.2, to map the incoming security token and, if required, to authenticate and authorize it.

“Creating a security profile” on page 294

You can create a security profile for use with Lightweight Directory Access Protocol (LDAP) or a WS-Trust V1.3 compliant security token server (STS), such as Tivoli Federated Identity Manager (TFIM) V6.2. Support is also provided for TFIM V6.1, for compatibility with previous versions of IBM Integration Bus. You can create the security profile by using either the **mqsicreateconfigurableservice** command or an editor in the IBM Integration Explorer.

“Configuring for identity propagation” on page 389

To enable a message flow to perform identity propagation, the input nodes must extract the identity and the output node must propagate it.

“Setting up message flow security” on page 292

Set up security on a message flow to control access based on the identity of a message passing through the message flow.

Related reference:

 **mqsicreateconfigurableservice** command

Use the **mqsicreateconfigurableservice** command to create an object name for a broker external resource.

 **mqsdeleteconfigurableservice** command

Use the **mqsdeleteconfigurableservice** command to delete a configurable service, such as a JMS provider, JDBC provider, or FTP server, that you have created by using the **mqsicreateconfigurableservice** command.

 **mqsichangeproperties** command

Use the **mqsichangeproperties** command to modify broker properties and properties of broker resources.

 **mqsireportproperties** command

Use the **mqsireportproperties** command to display properties that relate to a broker, an integration server, or a configurable service.

 MQInput node

Use the MQInput node to receive messages from clients that connect to the broker by using the WebSphere MQ Enterprise Transport, and that use the MQI and AMI application programming interfaces.

 HTTPInput node

Use the HTTPInput node to receive an HTTP message from an HTTP client for processing by a message flow.

 SOAPInput node

Use the SOAPInput node to process client SOAP messages, so that the broker operates as a SOAP Web Services provider.

 SOAPAsyncRequest node

Use the SOAPAsyncRequest node with the SOAPAsyncResponse node to construct a pair of message flows that call a Web service asynchronously.



HTTPRequest node

Use the HTTPRequest node to interact with a web service.



MQOutput node

Use the MQOutput node to send messages to clients that connect to the broker using the WebSphere MQ Enterprise Transport and that use the MQI and AMI application programming interfaces.

Creating a module chain in TFIM V6.2:

This topic describes how to create a module chain in Tivoli Federated Identity Manager (TFIM) V6.2.

About this task

When you use a WS-Trust v1.3 security token server (STS) for authentication, authorization, or mapping (or any combination of those operations), a single WS-Trust request is made to the trust service with the required parameters, which control the STS processing.

To enable IBM Integration Bus to use TFIM V6.2 for authorization, you need to configure TFIM to process the single WS-Trust request from the broker security manager. To configure TFIM, you must create a module chain to handle the request:

Procedure

1. Create a *Custom* module chain, and ensure that the chain performs all the actions that are specified in the broker security profile (Authenticate, Map, Authorize).
2. Set the *RequestType*, *Issuer* and *AppliesTo* properties of the module chain, so that it is invoked for the requests from the security enabled input node or SecurityPEP node. The parameters that are passed by the broker to TFIM are shown in the table in “Authentication, mapping, and authorization with TFIM V6.2 and TAM” on page 280.

What to do next

If your module chain includes an authorization module, and if the module specifies TAM, you must configure TAM to process the authorization requests from TFIM. For more information about how to do this, see “Configuring TAM for authorization using TFIM V6.2” on page 352.

Related concepts:

“Identity” on page 251

In IBM Integration Bus, an identity is a security token that uniquely identifies an individual, or that provides a set of assertions that can be validated.

“Authentication and validation” on page 257

Authentication is the process of establishing the identity of a user or system and verifying that the identity is valid. Applying authentication to a SAML security token involves validating the assertions that it carries and confirming that it is being processed within its validity period.

“Authorization” on page 261

Authorization is the process of verifying that an identity token has permission to access a message flow.

“Authentication, mapping, and authorization with TFIM V6.2 and TAM” on page 280

You can use IBM Integration Bus, Tivoli Federated Identity Manager (TFIM) V6.2, and Tivoli Access Manager (TAM) to control authentication, mapping, and authorization.

“Security profiles” on page 247

A security profile defines the security operations that are to be performed in a message flow at SecurityPEP nodes and security enabled input and output nodes.

“Security exception processing” on page 290

A security exception is raised when a message flow security failure occurs during security processing in an input node or SecurityPEP node.

Related tasks:

“Configuring TAM for authorization using TFIM V6.2” on page 352

Configure Tivoli Access Manager (TAM) to enable authorization using Tivoli Federated Identity Manager (TFIM) V6.2.

“Configuring identity authentication and security token validation” on page 312

You can configure a message flow to perform identity authentication or security token validation using Integrated Windows Authentication (IWA), Lightweight Directory Access Protocol (LDAP), a WS-Trust V1.3 compliant security token server (STS) such as Tivoli Federated Identity Manager (TFIM) Version 6.2, or HTTP requests. Support for TFIM V6.1 is also provided, for compatibility with previous versions of IBM Integration Bus.

“Configuring authorization with a WS-Trust v1.3 STS (TFIM V6.2)” on page 347

You can configure supported message flow input nodes or SecurityPEP nodes to perform authorization of an identity or security token by using a WS-Trust v1.3 compliant security token server (STS), such as Tivoli Federated Identity Manager (TFIM) V6.2.

“Creating a security profile” on page 294

You can create a security profile for use with Lightweight Directory Access Protocol (LDAP) or a WS-Trust V1.3 compliant security token server (STS), such as Tivoli Federated Identity Manager (TFIM) V6.2. Support is also provided for TFIM V6.1, for compatibility with previous versions of IBM Integration Bus. You can create the security profile by using either the **mqscreateconfigurable** command or an editor in the IBM Integration Explorer.

“Setting up message flow security” on page 292

Set up security on a message flow to control access based on the identity of a message passing through the message flow.

Related reference:

 **mqscreateconfigurable** command

Use the **mqscreateconfigurable** command to create an object name for a broker external resource.

 **mqsdeleteconfigurable** command

Use the **mqsdeleteconfigurable** command to delete a configurable service, such as a JMS provider, JDBC provider, or FTP server, that you have created by using the **mqscreateconfigurable** command.

 **mqschangeproperties** command

Use the **mqschangeproperties** command to modify broker properties and properties of broker resources.

 **mqsreportproperties** command

Use the **mqsreportproperties** command to display properties that relate to a

broker, an integration server, or a configurable service.



MQInput node

Use the MQInput node to receive messages from clients that connect to the broker by using the WebSphere MQ Enterprise Transport, and that use the MQI and AMI application programming interfaces.



HTTPInput node

Use the HTTPInput node to receive an HTTP message from an HTTP client for processing by a message flow.



SOAPInput node

Use the SOAPInput node to process client SOAP messages, so that the broker operates as a SOAP Web Services provider.



SOAPAsyncRequest node

Use the SOAPAsyncRequest node with the SOAPAsyncResponse node to construct a pair of message flows that call a Web service asynchronously.



HTTPRequest node

Use the HTTPRequest node to interact with a web service.



MQOutput node

Use the MQOutput node to send messages to clients that connect to the broker using the WebSphere MQ Enterprise Transport and that use the MQI and AMI application programming interfaces.

Configuring TAM for authorization using TFIM V6.2:

Configure Tivoli Access Manager (TAM) to enable authorization using Tivoli Federated Identity Manager (TFIM) V6.2.

About this task

To configure TAM for a TFIM V6.2 TAMAuthorizationSTSModule, complete the following steps using the pdadmin utility.

Procedure

1. Check that the **action group** used by the TFIM authorization module is available. The action group used is *WebService*:
action group list
If *WebService* is not listed, create it:
action group create WebService
2. Display the **action** in the action group used by the TFIM authorization module. The action used is "i":
action list WebService
If action "i" <label> 0 is not listed, create it. The value of <label> can vary:
action create i <label> 0 WebService
3. Create the Access Control List (ACL) that will be used to grant access to one or more message flows. First, create the ACL and give the administrators access to it. In this example, iv-admin is the administration group and sec_master is the main administrator:

```
acl create <AclName>
acl modify <AclName> set Group iv-admin TcmdbsvaBRxl[WebService]i
acl modify <AclName> set User sec_master TcmdbsvaBRxl[WebService]i
```

4. Grant access to all authenticated users, or specific groups, by adding them to the ACL. Grant any authenticated identity access:

```
acl modify <AclName> set Any-other Trx[WebService]i
```

To add a specific group:

```
acl modify <AclName> set group <GroupName> Trx[WebService]i
```

In these strings, each occurrence of Trx[] is an action, and corresponds to the value of the stsuser Action context attribute that is passed into the TAMAuthorizationSTSModule. For more information, see “Authentication, mapping, and authorization with TFIM V6.2 and TAM” on page 280.

5. Create a protected object space path in TAM to correspond to the value of the stsuser ObjectName context attribute that is passed into the TAMAuthorizationSTSModule using the following command syntax:

```
objectspace create /<ObjectName>
```

For more information, see “Authentication, mapping, and authorization with TFIM V6.2 and TAM” on page 280.

6. Attach the ACL to the protected object space path that you have created. Each node in the object space inherits ACLs from its parent, and a lower level ACL can override a higher level one. Use the following command syntax to attach an ACL to a node in the object space path:

```
acl attach /<ObjectSpacePath> <AclName>
```

What to do next

For further information about configuring TAM, see IBM Security Systems information centers.

Related concepts:

“Identity” on page 251

In IBM Integration Bus, an identity is a security token that uniquely identifies an individual, or that provides a set of assertions that can be validated.

“Authentication and validation” on page 257

Authentication is the process of establishing the identity of a user or system and verifying that the identity is valid. Applying authentication to a SAML security token involves validating the assertions that it carries and confirming that it is being processed within its validity period.

“Identity mapping” on page 263

Identity mapping is the transformation of a security token from one format to another format, or the federation of an identity from one realm to an equivalent identity in another realm.

“Authorization” on page 261

Authorization is the process of verifying that an identity token has permission to access a message flow.

“Identity and security token propagation” on page 286

Identity and security token propagation enables the identity and security tokens (associated with each message) to be propagated throughout a message flow, and on to target applications through output or request nodes.

“Security profiles” on page 247

A security profile defines the security operations that are to be performed in a message flow at SecurityPEP nodes and security enabled input and output nodes.

“Security exception processing” on page 290

A security exception is raised when a message flow security failure occurs during security processing in an input node or SecurityPEP node.

“Authentication, mapping, and authorization with TFIM V6.2 and TAM” on page 280

You can use IBM Integration Bus, Tivoli Federated Identity Manager (TFIM) V6.2, and Tivoli Access Manager (TAM) to control authentication, mapping, and authorization.

Related tasks:

“Creating a module chain in TFIM V6.2” on page 350

This topic describes how to create a module chain in Tivoli Federated Identity Manager (TFIM) V6.2.

“Configuring the extraction of an identity or security token” on page 308

You can configure the SecurityPEP node or security enabled input nodes to extract the identity or security token from a message and store it in the properties tree identity fields, enabling it to be processed throughout the message flow and propagated at output or request nodes.

“Configuring identity authentication and security token validation” on page 312

You can configure a message flow to perform identity authentication or security token validation using Integrated Windows Authentication (IWA), Lightweight Directory Access Protocol (LDAP), a WS-Trust V1.3 compliant security token sever (STS) such as Tivoli Federated Identity Manager (TFIM) Version 6.2, or HTTP requests. Support for TFIM V6.1 is also provided, for compatibility with previous versions of IBM Integration Bus.

“Configuring identity mapping” on page 333

Configure a security token server (STS), such as Tivoli Federated Identity Manager (TFIM) V6.2, to map the incoming security token and, if required, to authenticate and authorize it.

“Creating a security profile” on page 294

You can create a security profile for use with Lightweight Directory Access Protocol (LDAP) or a WS-Trust V1.3 compliant security token server (STS), such as Tivoli Federated Identity Manager (TFIM) V6.2. Support is also provided for TFIM V6.1, for compatibility with previous versions of IBM Integration Bus. You can create the security profile by using either the **mqsicreateconfigurable-service** command or an editor in the IBM Integration Explorer.

“Configuring for identity propagation” on page 389

To enable a message flow to perform identity propagation, the input nodes must extract the identity and the output node must propagate it.

“Setting up message flow security” on page 292

Set up security on a message flow to control access based on the identity of a message passing through the message flow.

Related reference:

 **mqsicreateconfigurable-service** command

Use the **mqsicreateconfigurable-service** command to create an object name for a broker external resource.

 **mqsideleteconfigurable-service** command

Use the **mqsideleteconfigurable-service** command to delete a configurable service, such as a JMS provider, JDBC provider, or FTP server, that you have created by using the **mqsicreateconfigurable-service** command.

 **mqsichangeproperties** command

Use the **mqsichangeproperties** command to modify broker properties and

properties of broker resources.

mqsiereportproperties command

Use the **mqsiereportproperties** command to display properties that relate to a broker, an integration server, or a configurable service.

MQInput node

Use the MQInput node to receive messages from clients that connect to the broker by using the WebSphere MQ Enterprise Transport, and that use the MQI and AMI application programming interfaces.

HTTPInput node

Use the HTTPInput node to receive an HTTP message from an HTTP client for processing by a message flow.

SOAPInput node

Use the SOAPInput node to process client SOAP messages, so that the broker operates as a SOAP Web Services provider.

SOAPAsyncRequest node

Use the SOAPAsyncRequest node with the SOAPAsyncResponse node to construct a pair of message flows that call a Web service asynchronously.

HTTPRequest node

Use the HTTPRequest node to interact with a web service.

MQOutput node

Use the MQOutput node to send messages to clients that connect to the broker using the WebSphere MQ Enterprise Transport and that use the MQI and AMI application programming interfaces.

Configuring authorization with TFIM V6.1:

You can configure a message flow to perform authorization on an identity by using Tivoli Federated Identity Manager (TFIM) V6.1.

Before you begin

Before you start:

Before you configure a message flow to perform authorization with TFIM V6.1:

- Check that an appropriate security profile exists, or create a new security profile. See “Creating a security profile for TFIM V6.1” on page 306.
- Define the required users and groups in TFIM.

About this task

Note: Support for TFIM V6.1 is included for compatibility with previous versions of IBM Integration Bus. If possible, upgrade to TFIM V6.2 and follow the instructions in “Configuring authorization with a WS-Trust v1.3 STS (TFIM V6.2)” on page 347.

The broker security manager issues an authorization request to the TFIM trust service with the following three parameters, which select the TFIM module chain to be used:

- Issuer = Properties.IdentitySourceIssuedBy

- Applies To = The Fully Qualified Name of the Flow: <Brokername>.<Integration Server Name>.<Message Flow Name>
- Token = Properties.IdentitySourceToken

Authorization is performed with TFIM using an instance of the TFIM AuthorizationSTSModule in the selected module chain. The TFIM AuthorizationSTSModule must be set with Mode = Other. This AuthorizationSTSModule authorizes a user by checking an Access Control List (ACL) from Tivoli Access Manager (TAM). TFIM performs the authorization check by verifying that the action "i" (invoke) has been granted in an ACL for the WebService action group.

The ACL is found starting from the root of the TAM object space using a path formed from the Authorization module **Web service protected object name** parameter, followed by the **Port Type** and **Operation Name** from the authorization request. When the broker makes an authorization request to TFIM, the **Port Type** and **Operation Name** parameters have the following values:

- PortType:<Message flow name>
- Operation "MessageFlowAccess"

Therefore, the ACL is found at this location in the TAM object space:

```
/<WSProtectedObjectName>.<MessageFlowName>."MessageFlowAccess"
```

For more information about this process and the parameters, see “Authentication, mapping, and authorization with TFIM V6.1 and TAM” on page 277.

Steps for enabling TFIM authorization:

Procedure

To enable an existing message flow to perform authorization with TFIM, use the Broker Archive editor to select a security profile that has authorization enabled. You can set a security profile on a message flow or on individual input nodes. If no security profile is set for the input nodes, the setting is inherited from the setting on the message flow.

1. Switch to the Integration Development perspective.
2. In the Application Development view, right-click the BAR file, then click **Open with > Broker Archive Editor**.
3. Click the **Manage and Configure** tab.
4. Click the flow or node on which you want to set the security profile. The properties that you can configure for the message flow or for the node are displayed in the **Properties** view.
5. In the **Security Profile Name** field, select a security profile that has authorization enabled.
6. Save the BAR file.

What to do next

For a SOAPInput node to use the identity in the WS-Security header (rather than an underlying transport identity) an appropriate policy set and bindings must also be defined and specified. For more information, see Policy sets.

In addition to configuring IBM Integration Bus to perform authorization with TFIM, you must configure TFIM and TAM. For information about how to do this, see the following topics:

- “Creating a module chain in TFIM V6.1” on page 359
- “Configuring TAM for authorization using TFIM V6.1” on page 361.

For further information on how to configure TFIM, see the IBM Tivoli Federated Identity Manager product documentation.

Related concepts:

“Identity” on page 251

In IBM Integration Bus, an identity is a security token that uniquely identifies an individual, or that provides a set of assertions that can be validated.

“Authentication and validation” on page 257

Authentication is the process of establishing the identity of a user or system and verifying that the identity is valid. Applying authentication to a SAML security token involves validating the assertions that it carries and confirming that it is being processed within its validity period.

“Identity mapping” on page 263

Identity mapping is the transformation of a security token from one format to another format, or the federation of an identity from one realm to an equivalent identity in another realm.

“Authorization” on page 261

Authorization is the process of verifying that an identity token has permission to access a message flow.

“Identity and security token propagation” on page 286

Identity and security token propagation enables the identity and security tokens (associated with each message) to be propagated throughout a message flow, and on to target applications through output or request nodes.

“Security profiles” on page 247

A security profile defines the security operations that are to be performed in a message flow at SecurityPEP nodes and security enabled input and output nodes.

“Security exception processing” on page 290

A security exception is raised when a message flow security failure occurs during security processing in an input node or SecurityPEP node.

Related tasks:

“Configuring the extraction of an identity or security token” on page 308

You can configure the SecurityPEP node or security enabled input nodes to extract the identity or security token from a message and store it in the properties tree identity fields, enabling it to be processed throughout the message flow and propagated at output or request nodes.

“Configuring identity authentication and security token validation” on page 312

You can configure a message flow to perform identity authentication or security token validation using Integrated Windows Authentication (IWA), Lightweight Directory Access Protocol (LDAP), a WS-Trust V1.3 compliant security token server (STS) such as Tivoli Federated Identity Manager (TFIM) Version 6.2, or HTTP requests. Support for TFIM V6.1 is also provided, for compatibility with previous versions of IBM Integration Bus.

“Configuring identity mapping” on page 333

Configure a security token server (STS), such as Tivoli Federated Identity Manager (TFIM) V6.2, to map the incoming security token and, if required, to authenticate and authorize it.

“Creating a security profile” on page 294

You can create a security profile for use with Lightweight Directory Access Protocol (LDAP) or a WS-Trust V1.3 compliant security token server (STS), such as Tivoli Federated Identity Manager (TFIM) V6.2. Support is also provided for TFIM V6.1, for compatibility with previous versions of IBM Integration Bus. You can create the security profile by using either the **mqscreateconfigurable-service** command or an editor in the IBM Integration Explorer.

“Configuring for identity propagation” on page 389

To enable a message flow to perform identity propagation, the input nodes must extract the identity and the output node must propagate it.

“Setting up message flow security” on page 292

Set up security on a message flow to control access based on the identity of a message passing through the message flow.

Related reference:

 **mqscreateconfigurable-service** command

Use the **mqscreateconfigurable-service** command to create an object name for a broker external resource.

 **mqsdeleteconfigurable-service** command

Use the **mqsdeleteconfigurable-service** command to delete a configurable service, such as a JMS provider, JDBC provider, or FTP server, that you have created by using the **mqscreateconfigurable-service** command.

 **mqschange-properties** command

Use the **mqschange-properties** command to modify broker properties and properties of broker resources.

 **mqsreport-properties** command

Use the **mqsreport-properties** command to display properties that relate to a broker, an integration server, or a configurable service.

 MQInput node

Use the MQInput node to receive messages from clients that connect to the broker by using the WebSphere MQ Enterprise Transport, and that use the MQI and AMI application programming interfaces.

 HTTPInput node

Use the HTTPInput node to receive an HTTP message from an HTTP client for processing by a message flow.

 SOAPInput node

Use the SOAPInput node to process client SOAP messages, so that the broker operates as a SOAP Web Services provider.

 SOAPAsyncRequest node

Use the SOAPAsyncRequest node with the SOAPAsyncResponse node to construct a pair of message flows that call a Web service asynchronously.

 HTTPRequest node

Use the HTTPRequest node to interact with a web service.

 MQOutput node

Use the MQOutput node to send messages to clients that connect to the broker using the WebSphere MQ Enterprise Transport and that use the MQI and AMI application programming interfaces.

Creating a module chain in TFIM V6.1:

This topic describes how to create a module chain in Tivoli Federated Identity Manager (TFIM) V6.1.

About this task

To enable IBM Integration Bus to use TFIM V6.1 for authorization, you need to configure TFIM to process the security request from the message flow. To do this you need to create a module chain in TFIM to handle the request:

Procedure

1. Create a *Custom* module chain, and ensure that the chain performs all the actions required (Authenticate, Map, Authorize).
2. Set the *Issuer* and *AppliesTo* properties of the module chain, so that it is invoked for the requests from the message flow. When the broker makes a request to TFIM, the **Port Type** and **Operation Name** parameters have the following values:
 - PortType:<Message flow name>
 - Operation "MessageFlowAccess"

The *RequestType* is always set to *Validate*.

3. To perform authorization in a module chain, add an instance of the Authorization module in *other* mode, which allows the module parameter **Web Service protected object name** to be set for the Tivoli Access Manager (TAM) configuration.

What to do next

When you have created the module chain in TFIM, see “Configuring TAM for authorization using TFIM V6.1” on page 361 for information on how to configure TAM to process authorization requests from TFIM.

Related concepts:

“Identity” on page 251

In IBM Integration Bus, an identity is a security token that uniquely identifies an individual, or that provides a set of assertions that can be validated.

“Authentication and validation” on page 257

Authentication is the process of establishing the identity of a user or system and verifying that the identity is valid. Applying authentication to a SAML security token involves validating the assertions that it carries and confirming that it is being processed within its validity period.

“Authorization” on page 261

Authorization is the process of verifying that an identity token has permission to access a message flow.

“Authentication, mapping, and authorization with TFIM V6.1 and TAM” on page 277

Use IBM Integration Bus, Tivoli Federated Identity Manager (TFIM) V6.1, and Tivoli Access Manager (TAM) to control authentication, mapping, and authorization.

“Security profiles” on page 247

A security profile defines the security operations that are to be performed in a message flow at SecurityPEP nodes and security enabled input and output nodes.

“Security exception processing” on page 290

A security exception is raised when a message flow security failure occurs during

security processing in an input node or SecurityPEP node.

Related tasks:

“Configuring identity authentication and security token validation” on page 312
You can configure a message flow to perform identity authentication or security token validation using Integrated Windows Authentication (IWA), Lightweight Directory Access Protocol (LDAP), a WS-Trust V1.3 compliant security token sever (STS) such as Tivoli Federated Identity Manager (TFIM) Version 6.2, or HTTP requests. Support for TFIM V6.1 is also provided, for compatibility with previous versions of IBM Integration Bus.

“Configuring TAM for authorization using TFIM V6.1” on page 361
This topic describes how to configure Tivoli Access Manager (TAM) to enable authorization using Tivoli Federated Identity Manager (TFIM) V6.1.

“Creating a security profile” on page 294
You can create a security profile for use with Lightweight Directory Access Protocol (LDAP) or a WS-Trust V1.3 compliant security token server (STS), such as Tivoli Federated Identity Manager (TFIM) V6.2. Support is also provided for TFIM V6.1, for compatibility with previous versions of IBM Integration Bus. You can create the security profile by using either the **mqsicreateconfigurablesevice** command or an editor in the IBM Integration Explorer.

“Setting up message flow security” on page 292
Set up security on a message flow to control access based on the identity of a message passing through the message flow.

Related reference:



mqsicreateconfigurablesevice command

Use the **mqsicreateconfigurablesevice** command to create an object name for a broker external resource.



mqsideleteconfigurablesevice command

Use the **mqsideleteconfigurablesevice** command to delete a configurable service, such as a JMS provider, JDBC provider, or FTP server, that you have created by using the **mqsicreateconfigurablesevice** command.



mqsichangeproperties command

Use the **mqsichangeproperties** command to modify broker properties and properties of broker resources.



mqsireportproperties command

Use the **mqsireportproperties** command to display properties that relate to a broker, an integration server, or a configurable service.



MQInput node

Use the **MQInput** node to receive messages from clients that connect to the broker by using the WebSphere MQ Enterprise Transport, and that use the MQI and AMI application programming interfaces.



HTTPInput node

Use the **HTTPInput** node to receive an HTTP message from an HTTP client for processing by a message flow.



SOAPInput node

Use the **SOAPInput** node to process client SOAP messages, so that the broker operates as a SOAP Web Services provider.



SOAPAsyncRequest node

Use the SOAPAsyncRequest node with the SOAPAsyncResponse node to construct a pair of message flows that call a Web service asynchronously.



HTTPRequest node

Use the HTTPRequest node to interact with a web service.



MQOutput node

Use the MQOutput node to send messages to clients that connect to the broker using the WebSphere MQ Enterprise Transport and that use the MQI and AMI application programming interfaces.

Configuring TAM for authorization using TFIM V6.1:

This topic describes how to configure Tivoli Access Manager (TAM) to enable authorization using Tivoli Federated Identity Manager (TFIM) V6.1.

About this task

To configure TAM to process an authorization request from TFIM, complete the following steps. The examples relate to the TAM Version 6.01 pdadmin utility:

Procedure

1. Check that the **action group** used by the TFIM authorization module is available. The action group used is *WebService*:

```
action group list
```

 If *WebService* is not listed, create it:

```
action group create WebService
```
2. Display the **action** in the action group used by the TFIM authorization module. The action used is "i":

```
action list WebService
```

 If action "i" <label> 0 is not listed, create it. The value of <label> can vary:

```
action create i <label> 0 WebService
```
3. Create the Access Control List (ACL) that will be used to grant access to one or more message flows. First, create the ACL and give the administrators access to it. In this example, iv-admin is the administration group and sec_master is the main administrator:

```
acl create <Ac1Name>
acl modify <Ac1Name> set Group iv-admin TcmdsavaBRx1[WebService]i
acl modify <Ac1Name> set User sec_master TcmdsavaBRx1[WebService]i
```
4. Grant access to all authenticated users, or specific groups, by adding them to the ACL. Grant any authenticated identity access:

```
acl modify <Ac1Name> set Any-other Trx[WebService]i
```

 To add a specific group:

```
acl modify <Ac1Name> set group <GroupName> Trx[WebService]i
```
5. Define protected object spaces in TAM for authorization of message flows:
 - a. Create the *application container object* as the root of the protected object space. This is the name that is used to link an instance of a TFIM AuthorizationSTSModule (within a module chain) into the TAM object space. The container object name is specified to match the **Web Service protected object name** parameter on a TFIM Authorization module.

```
objectspace create /<ContainerObjectName> <Description> 14
```

- b. Create the container objects in the tree for each broker message flow that is being authorized. The message flow name is used by TFIM to locate a point in the TAM Object Space tree for Authorization, through the attached ACL. The message flow name is passed as the **PortType** in the WS-Trust request to TFIM. Use the following command to create the object tree node representing each flow to be authorized:

```
object create /<ContainerObjectName>/<FlowName> <Description> 11 ispolicyattachable yes
```

The **ispolicyattachable** parameter applies to all levels, so you can attach an ACL at any level.

- c. Create the leaf object that represents the authorized object to grant access to the message flow. This is the fixed string *MessageFlowAccess*, which the broker sends to TFIM through the TFIM **OperationName** extension to the WS-Trust request. A fixed name (*MessageFlowAccess*) is used instead of a true operation name, because the broker does not necessarily know at the input node which operation a flow is going to perform. The command syntax is:

```
object create /<ContainerObjectName>/<FlowName>/MessageFlowAccess <Description> 12 ispolicyattachable yes
```

where *<FlowName>* has been created in a previous step.

6. Attach the ACL to the relevant node in the protected object space tree. Each node in the object space inherits ACLs from its parent, and a lower level ACL can override a higher level one. Use the following command syntax to attach an ACL to a node in the object space:

```
acl attach /<ObjectSpacePath> <Ac1Name>
```

To attach an ACL to the leaf node:

```
acl attach /<ContainerObjectName>/<FlowName>/MessageFlowAccess <Ac1Name>
```

What to do next

For further information about configuring TAM, see IBM Security Systems information centers.

Related concepts:

“Identity” on page 251

In IBM Integration Bus, an identity is a security token that uniquely identifies an individual, or that provides a set of assertions that can be validated.

“Authentication and validation” on page 257

Authentication is the process of establishing the identity of a user or system and verifying that the identity is valid. Applying authentication to a SAML security token involves validating the assertions that it carries and confirming that it is being processed within its validity period.

“Identity mapping” on page 263

Identity mapping is the transformation of a security token from one format to another format, or the federation of an identity from one realm to an equivalent identity in another realm.

“Authorization” on page 261

Authorization is the process of verifying that an identity token has permission to access a message flow.

“Identity and security token propagation” on page 286

Identity and security token propagation enables the identity and security tokens (associated with each message) to be propagated throughout a message flow, and on to target applications through output or request nodes.

“Security profiles” on page 247

A security profile defines the security operations that are to be performed in a message flow at SecurityPEP nodes and security enabled input and output nodes.

“Security exception processing” on page 290

A security exception is raised when a message flow security failure occurs during security processing in an input node or SecurityPEP node.

Related tasks:

“Configuring the extraction of an identity or security token” on page 308

You can configure the SecurityPEP node or security enabled input nodes to extract the identity or security token from a message and store it in the properties tree identity fields, enabling it to be processed throughout the message flow and propagated at output or request nodes.

“Configuring identity authentication and security token validation” on page 312

You can configure a message flow to perform identity authentication or security token validation using Integrated Windows Authentication (IWA), Lightweight Directory Access Protocol (LDAP), a WS-Trust V1.3 compliant security token server (STS) such as Tivoli Federated Identity Manager (TFIM) Version 6.2, or HTTP requests. Support for TFIM V6.1 is also provided, for compatibility with previous versions of IBM Integration Bus.

“Configuring identity mapping” on page 333

Configure a security token server (STS), such as Tivoli Federated Identity Manager (TFIM) V6.2, to map the incoming security token and, if required, to authenticate and authorize it.

“Creating a security profile” on page 294

You can create a security profile for use with Lightweight Directory Access Protocol (LDAP) or a WS-Trust V1.3 compliant security token server (STS), such as Tivoli Federated Identity Manager (TFIM) V6.2. Support is also provided for TFIM V6.1, for compatibility with previous versions of IBM Integration Bus. You can create the security profile by using either the **mqscreateconfigurable** service command or an editor in the IBM Integration Explorer.

“Configuring for identity propagation” on page 389

To enable a message flow to perform identity propagation, the input nodes must extract the identity and the output node must propagate it.

“Setting up message flow security” on page 292

Set up security on a message flow to control access based on the identity of a message passing through the message flow.

Related reference:



mqscreateconfigurable service command

Use the **mqscreateconfigurable** service command to create an object name for a broker external resource.



msideleteconfigurable service command

Use the **msideleteconfigurable** service command to delete a configurable service, such as a JMS provider, JDBC provider, or FTP server, that you have created by using the **mqscreateconfigurable** service command.



mqschangeproperties command

Use the **mqschangeproperties** command to modify broker properties and properties of broker resources.



mqsreportproperties command

Use the **mqsreportproperties** command to display properties that relate to a broker, an integration server, or a configurable service.

MQInput node

Use the MQInput node to receive messages from clients that connect to the broker by using the WebSphere MQ Enterprise Transport, and that use the MQI and AMI application programming interfaces.

HTTPInput node

Use the HTTPInput node to receive an HTTP message from an HTTP client for processing by a message flow.

SOAPInput node

Use the SOAPInput node to process client SOAP messages, so that the broker operates as a SOAP Web Services provider.

SOAPAsyncRequest node

Use the SOAPAsyncRequest node with the SOAPAsyncResponse node to construct a pair of message flows that call a Web service asynchronously.

HTTPRequest node

Use the HTTPRequest node to interact with a web service.

MQOutput node

Use the MQOutput node to send messages to clients that connect to the broker using the WebSphere MQ Enterprise Transport and that use the MQI and AMI application programming interfaces.

Securing message flows by using SSL configuration

You can secure message flows by configuring nodes within the message flows to use SSL and certificates.

After you establish a public key infrastructure configuration for your whole broker or for some of its integration servers, you can use the configuration within your message flows. For information on establishing a public key infrastructure, see “Setting up a public key infrastructure” on page 415. When you configure message flow nodes that make or accept TCP connections, you can set SSL encryption and certificates on those connections.

For more information about using SSL and certificates to secure your message flows, see:

- Configuring the broker to use SSL with JMS nodes
- Configuring SOAPInput and SOAPReply nodes to use SSL (HTTPS)
- Configuring SOAPRequest and SOAPAsyncRequest nodes to use SSL (HTTPS)
- Configuring HTTPInput and HTTPReply nodes to use SSL (HTTPS)
- Configuring the HTTPRequest and HTTPAsyncRequest nodes to use SSL (HTTPS)
- Enabling SSL on the WebSphere MQ Java Client
- Securing the connection to CICS Transaction Server for z/OS by using SSL
- Securing the connection to IMS by using SSL
- SSL and the TCP/IP nodes
- Using security exits

Related tasks:

“Implementing SSL authentication” on page 415

Use SSL authentication to enhance security in your broker environment.

Configuring the broker to use SSL with JMS nodes:

Configure your broker to work with a JMS provider that supports JMS clients that can connect by using the Secure Sockets Layer (SSL) protocol.

Before you begin

Before you start: Create a keystore file to store the broker's certificates: "Setting up a public key infrastructure" on page 415.

About this task

The JMS 1.1 Specification states that JMS does not provide features for controlling or configuring message integrity or message privacy. JMS providers typically support these additional features, and provide their own administration tools to configure these services. Clients can get the appropriate security configuration as part of the administered objects that they use.

If you want to apply SSL security to the JMS connections created by the three built-in nodes JMSInput, JMSOutput, and JMSReply, check the documentation supplied by your chosen JMS provider. The configuration of the JNDI administered objects that are used by the JMS nodes is specific to each JMS provider.

The three built-in nodes JMSInput, JMSOutput, and JMSReply are referred to in this topic by the generic term JMS nodes; apply the information and instructions here to the specific type of node that you are using.

One example of a JMS provider that provides SSL support for connecting JMS clients is TIBCO Enterprise Message Service (EMS). The following sections describe the authentication model used for JMS nodes, with specific reference to TIBCO EMS, and provide information about how to connect JMS nodes to a TIBCO EMS JMS Server securely by using SSL:

1. "SSL authentication model for the JMS nodes"
2. "Configuring your JMS nodes to use SSL-enabled JNDI administered objects"

SSL authentication model for the JMS nodes:

About this task

The JMS provider TIBCO EMS supports Java clients that can use either the Java Secure Sockets Extension (JSSE) Java package, or an SSL implementation supplied by Entrust. For details about the services provided, see the documentation provided with your chosen package.

TIBCO EMS supports a number of different authentication scenarios, but JMS nodes can use only client authentication to the server. In this scenario, the TIBCO EMS server requests the client's digital certificate during an SSL handshake, and checks its issuer against the server's list of trusted Certificate Authorities. If the authority is not in the server's list, further communications are prevented with the JMS node.

Therefore, you must configure the EMS server to explicitly enable client authentication of the SSL certificates in its configuration file; configure the JNDI administered SSL JMS connection factories for the same level of support.

Configuring your JMS nodes to use SSL-enabled JNDI administered objects:

About this task

The JMS nodes use JNDI to look up a connection factory object that is used to create JMS connections to a TIBCO EMS server.

Procedure

1. Configure the JMS node property Connection factory name to specify a pre-configured connection factory that is enabled for SSL connectivity. Make sure that you have set the appropriate parameters in the corresponding SSL JMS connection factory definition:

- Enable client authentication
- Specify the SSL protocol in the server URL
- Set other parameters to define the support you require.

See the provider's documentation for information about how to generate this JNDI administered object:

2. Configure the JMS node property Location JNDI Bindings with the URL that points to the JNDI bindings containing the JNDI administered objects for SSL connectivity.

For TIBCO EMS , this URL takes the following format:

```
tibjmsnaming://server_name:ssl_port
```

- *server_name* is the host name of the computer where the server is installed.
- *ssl_port* is the server port for SSL connectivity; typically, this is port 7243 for a TIBCO EMS server.

3. Make the TIBCO EMS client JAR files available to the broker to which you deploy the message flow that includes your JMS nodes. Use the **mqsicreateconfigurablesevice** or the **mqsichangeproperties** command to set the JMSProviders configurable service property jarsURL to point to the directory that contains the JMS provider's client JAR files and the SSL vendor's JAR files.

If you are using JSSE for the SSL support, the following JAR files are typically located in the jarsURL directory:

- jsse.jar
- net.jar
- jcert.jar
- tibcrypt.jar

You can find standard non-SSL client JAR files in the same location.

Configuring SOAPInput and SOAPReply nodes to use SSL (HTTPS):

Configure the SOAP nodes to communicate with other applications that use HTTPS by creating a keystore file, and configuring the broker to use SSL.

Before you begin

Before you start: Set up a public key infrastructure (PKI) at broker or integration server level: "Setting up a public key infrastructure" on page 415.

About this task

Follow these steps to configure the SOAPInput and SOAPReply nodes to communicate with other applications using HTTP over SSL:

1. If you are using the broker listener: Configure the broker to use SSL
2. If you are using the integration server (embedded) listener: Configure an integration server to use SSL
3. Test your configuration

If you configured your broker and integration servers such that the broker listener is used for some integration servers, and the integration server listener for other integration servers, you must complete step 1 for the first set of integration servers and step 2 for each integration server in the second set.

For information about which listener to use for HTTP messages, see HTTP listeners.

Configuring the broker to use SSL:

About this task

Complete the following steps:

Procedure

1. Turn on SSL support in the broker, by setting a value for **enableSSLConnector**

```
mqsichangeproperties broker_name
  -b httplistener -o HTTPListener
  -n enableSSLConnector -v true
```

2. Optional: If you do not want to use the default port 7083 for HTTPS messages, specify the **port** on which the broker listens:

```
mqsichangeproperties broker_name
  -b httplistener -o HTTPSConnector
  -n port -v Port to listen on for https
```

On UNIX systems, only processes that run under a privileged user account (in most cases, root) can bind to ports lower than 1024. For the broker to listen on these ports, the user ID under which the broker is started must be root.

3. Optional: Enable Client Authentication (mutual authentication):

```
mqsichangeproperties broker_name -b httplistener -o HTTPSConnector
  -n clientAuth -v true
```

4. Restart the broker after changing one or more of the HTTP listener properties.
5. Optional: Use the following commands to display HTTP listener properties:

```
mqsireportproperties broker_name -b httplistener -o AllReportableEntityNames -a
mqsireportproperties broker_name -b httplistener -o HTTPListener -a
mqsireportproperties broker_name -b httplistener -o HTTPSConnector -a
```

Configuring an integration server to use SSL:

About this task

Complete the following steps:

Procedure

1. Optional: Specify a specific port on which the integration server listens for HTTPS requests, or leave the value unset to use the next available port number.

```
mqsichangeproperties broker_name
  -e integration_server_name -o HTTPSConnector
  -n explicitlySetPortNumber -v port_number
```

On UNIX systems, only processes that run under a privileged user account (in most cases, root) can bind to ports lower than 1024. For the integration server to listen on these ports, the user ID under which the broker is started must be root.

If you do not complete this step, the first available port in the default range (7843 - 7884) is used.

- Optional: Enable Client Authentication (mutual authentication):

```
mqsichangeproperties broker_name  
-e integration_server_name -o HTTPSConnector  
-n clientAuth -v true
```

- Optional: Change the SSL protocol. The default protocol for the HTTPInput node is TLS. Run the following command to change it to SSL:

```
mqsichangeproperties broker_name  
-e integration_server_name -o HTTPSConnector  
-n sslProtocol -v SSL
```

- Restart the broker after changing one or more of the listener properties.
- Optional: Use the following command to display HTTPS properties:

```
mqsireportproperties broker_name  
-e integration_server_name -o HTTPSConnector -r
```

Testing your configuration:

About this task

Use the SOAP Nodes sample to test your configuration. You can view information about samples only when you use the information center that is integrated with the IBM Integration Toolkit or the online information center. You can run samples only when you use the information center that is integrated with the IBM Integration Toolkit.

Procedure

- Import the SOAP Nodes sample.
- Enable SSL in the SOAPNodesSampleConsumerFlow message flow by completing the following steps:
 - Open the Invoke_submitPO subflow.
 - Change the HTTPTransport properties for the SOAPRequest node. In the **Web Service URL** field, make the following changes:
 - Change http to https
 - Change the port number to the port number of your HTTPSConnector port.

Note: The default value of the HTTPSConnector port is 7843 but you can use the following command to verify the port number that is configured in your deployment.

```
mqsireportproperties broker_name -e integration_server_name -o HTTPSConnector -n port
```

If there are no other HTTPS services in your deployment, the **mqsireportproperties** command returns a 0 and you should be able to use the default value of 7843 for the value of your HTTPSConnector port.

- If you set up your public key infrastructure by following the instructions that are detailed in “Setting up a public key infrastructure” on page 415, all other properties should be correct. Otherwise, you must select the appropriate protocol and change other SSL properties to match your configuration.

3. Enable SSL in the SOAPNodesSampleProvider message flow by opening the properties for the SOAPInput node and selecting **Use HTTPS** in the HTTP Transport properties panel.
4. If OrderService_SOAPNodesSampleProviderFlow and submitPO_OrderService_SOAPNodesSampleConsumerFlow are not already subflows, you must convert them to subflows. For more information about converting between message flows and subflows, see [Converting between message flows and subflows](#).
5. Refresh the BAR file and deploy.
6. Test the sample. For example, you can test the sample by completing the following steps:

- a. From a web browser, enter the following URL.

`https://localhost:port_number/acmeOrders/WADDR/ProcessOrders`

where *port_number* is the port number of your HTTPSConnector port.

- b. When you are prompted to accept the certificate, click **Yes**.

If you are using a self-signed certificate in your PKI infrastructure, the sample is working correctly if the browser window displays the message *There is a problem with this web site's security certificate*.

If you are using a certificate from a certificate authority in your PKI infrastructure, the sample is working correctly if the browser displays a lock symbol next to the URL.

Note: You can ignore any error messages that are associated with the sample.

Related concepts:

“Security overview” on page 203

When you are designing an IBM Integration Bus application it is important to consider the security measures that are needed to protect the information in the system.

Related tasks:

“Implementing SSL authentication” on page 415

Use SSL authentication to enhance security in your broker environment.

“Configuring SOAPRequest and SOAPAsyncRequest nodes to use SSL (HTTPS)” on page 370

Configure the SOAPRequest and SOAPAsyncRequest nodes to communicate with other applications that use HTTP over SSL.

Related reference:

 **mqsichangeproperties** command

Use the **mqsichangeproperties** command to modify broker properties and properties of broker resources.

 **mqsireportproperties** command

Use the **mqsireportproperties** command to display properties that relate to a broker, an integration server, or a configurable service.

 SOAPInput node

Use the SOAPInput node to process client SOAP messages, so that the broker operates as a SOAP Web Services provider.

 SOAPReply node

Use the SOAPReply node to send SOAP messages from the broker to the originating client in response to a message received by a SOAPInput node.

SOAPRequest node

Use the SOAPRequest node to send a SOAP request to the remote Web service.

SOAPAsyncRequest node

Use the SOAPAsyncRequest node with the SOAPAsyncResponse node to construct a pair of message flows that call a Web service asynchronously.

Configuring SOAPRequest and SOAPAsyncRequest nodes to use SSL (HTTPS):

Configure the SOAPRequest and SOAPAsyncRequest nodes to communicate with other applications that use HTTP over SSL.

Before you begin

Before you start: Set up a public key infrastructure (PKI) at broker or integration server level: “Setting up a public key infrastructure” on page 415.

Configuring the nodes:

Procedure

On the HTTP Transport page of the properties for the node, set the following properties:

1. In the Web Service URL property, type the URL of the web service you want to call.
2. Optional: Select the Enable certificate revocation list checking property. For more information on Certificate Revocation List (CRL) checking see, “Working with certificate revocation lists” on page 426.
3. Set other SSL properties as appropriate.

What to do next

Test your configuration.

Testing your configuration:

About this task

Use the SOAP Nodes sample to test your configuration. You can view information about samples only when you use the information center that is integrated with the IBM Integration Toolkit or the online information center. You can run samples only when you use the information center that is integrated with the IBM Integration Toolkit.

Procedure

1. Import the SOAP Nodes sample.
2. Enable SSL in the SOAPNodesSampleConsumerFlow message flow by completing the following steps:
 - a. Open the Invoke_submitPO subflow.
 - b. Change the HTTPTransport properties for the SOAPRequest node. In the **Web Service URL** field, make the following changes:
 - Change http to https
 - Change the port number to the port number of your HTTPSConnector port.

Note: The default value of the HTTPSConnector port is 7843 but you can use the following command to verify the port number that is configured in your deployment.

```
mqsireportproperties broker_name -e integration_server_name -o HTTPSConnector -n port
```

If there are no other HTTPS services in your deployment, the **mqsireportproperties** command returns a 0 and you should be able to use the default value of 7843 for the value of your HTTPSConnector port.

- c. If you set up your public key infrastructure by following the instructions that are detailed in “Setting up a public key infrastructure” on page 415, all other properties should be correct. Otherwise, you must select the appropriate protocol and change other SSL properties to match your configuration.
3. Enable SSL in the SOAPNodesSampleProvider message flow by opening the properties for the SOAPInput node and selecting **Use HTTPS** in the HTTP Transport properties panel.
4. If OrderService_SOAPNodesSampleProviderFlow and submitPO_OrderService_SOAPNodesSampleConsumerFlow are not already subflows, you must convert them to subflows. For more information about converting between message flows and subflows, see *Converting between message flows and subflows*.
5. Refresh the BAR file and deploy.
6. Test the sample. For example, you can test the sample by completing the following steps:
 - a. From a web browser, enter the following URL.

```
https://localhost:port_number/acmeOrders/WADDR/ProcessOrders
```

where *port_number* is the port number of your HTTPSConnector port.

- b. When you are prompted to accept the certificate, click **Yes**.

If you are using a self-signed certificate in your PKI infrastructure, the sample is working correctly if the browser window displays the message *There is a problem with this web site's security certificate*.

If you are using a certificate from a certificate authority in your PKI infrastructure, the sample is working correctly if the browser displays a lock symbol next to the URL.

Note: You can ignore any error messages that are associated with the sample.

Related concepts:

“Security overview” on page 203

When you are designing an IBM Integration Bus application it is important to consider the security measures that are needed to protect the information in the system.

Related tasks:

“Implementing SSL authentication” on page 415

Use SSL authentication to enhance security in your broker environment.

“Configuring SOAPRequest and SOAPAsyncRequest nodes to use SSL (HTTPS)” on page 370

Configure the SOAPRequest and SOAPAsyncRequest nodes to communicate with other applications that use HTTP over SSL.

Related reference:

mqsichangeproperties command

Use the **mqsichangeproperties** command to modify broker properties and properties of broker resources.

mqsireportproperties command

Use the **mqsireportproperties** command to display properties that relate to a broker, an integration server, or a configurable service.

SOAPInput node

Use the SOAPInput node to process client SOAP messages, so that the broker operates as a SOAP Web Services provider.

SOAPReply node

Use the SOAPReply node to send SOAP messages from the broker to the originating client in response to a message received by a SOAPInput node.

SOAPRequest node

Use the SOAPRequest node to send a SOAP request to the remote Web service.

SOAPAsyncRequest node

Use the SOAPAsyncRequest node with the SOAPAsyncResponse node to construct a pair of message flows that call a Web service asynchronously.

Configuring HTTPInput and HTTPReply nodes to use SSL (HTTPS):

Configure the HTTPInput and HTTPReply nodes to communicate with other applications that use HTTPS by creating a keystore file, configuring the broker or integration server to use SSL, and creating a message flow to process HTTPS requests.

Before you begin

Before you start: Set up a public key infrastructure (PKI) at broker level by following the instructions in “Setting up a public key infrastructure” on page 415.

About this task

Follow these steps to configure the HTTPInput and HTTPReply nodes to communicate with other applications using HTTP over SSL:

1. If you are using the broker listener: Configure the broker to use SSL
2. If you are using the integration server listener: Configure the integration server to use SSL
3. Create a message flow
4. Test your configuration

If you have configured your broker and integration servers such that the broker listener is used for some integration servers, and the integration server listener for other integration servers, you must complete step 1 for the first set of integration servers and step 2 for each integration server in the second set.

For information about which listener to use for HTTPS messages, see HTTP listeners.

Configuring the broker to use SSL:

About this task

Complete the following steps:

Procedure

1. Turn on SSL support in the broker, by setting a value for **enableSSLConnector**

```
mqsichangeproperties broker_name  
-b httplistener -o HTTPListener  
-n enableSSLConnector -v true
```

2. Optional: If you do not want to use the default port 7083 for HTTPS messages, specify the **port** on which the broker listens:

```
mqsichangeproperties broker_name  
-b httplistener -o HTTPSConnector  
-n port -v Port to listen on for https
```

On UNIX systems, only processes that run under a privileged user account (in most cases, root) can bind to ports lower than 1024. For the broker to listen on these ports, the user ID under which the broker is started must be root.

3. Optional: Enable Client Authentication (mutual authentication):

```
mqsichangeproperties broker_name -b httplistener -o HTTPSConnector  
-n clientAuth -v true
```

4. Restart the broker after changing one or more of the HTTP listener properties.

5. Optional: Use the following commands to display HTTP listener properties:

```
mqsireportproperties broker_name -b httplistener -o AllReportableEntityNames -a  
mqsireportproperties broker_name -b httplistener -o HTTPListener -a  
mqsireportproperties broker_name -b httplistener -o HTTPSConnector -a
```

Configuring an integration server to use SSL:

About this task

Complete the following steps:

Procedure

1. Optional: Specify a specific port on which the integration server listens for HTTPS requests, or leave the value unset to use the next available port number.

```
mqsichangeproperties broker_name  
-e integration_server_name -o HTTPSConnector  
-n explicitlySetPortNumber -v port_number
```

On UNIX systems, only processes that run under a privileged user account (in most cases, root) can bind to ports lower than 1024. For the integration server to listen on these ports, the user ID under which the broker is started must be root.

If you do not complete this step, the first available port in the default range (7843 - 7884) is used.

2. Optional: Enable Client Authentication (mutual authentication):

```
mqsichangeproperties broker_name  
-e integration_server_name -o HTTPSConnector  
-n clientAuth -v true
```

3. Optional: Change the SSL protocol. The default protocol for the HTTPInput node is TLS. Run the following command to change it to SSL:

```
mqsichangeproperties broker_name  
-e integration_server_name -o HTTPSConnector  
-n sslProtocol -v SSL
```

- Restart the broker after changing one or more of the listener properties.
- Optional: Use the following command to display HTTPS properties:

```
mqsireportproperties broker_name  
-e integration_server_name -o HTTPSConnector -r
```

Creating a message flow to process HTTPS requests:

About this task

You can create a simple message flow to use HTTPS by connecting an HTTPInput node to an HTTPReply node. The two most important properties to set on the HTTPInput node are:

- Path suffix for URL; for example, /* or /testHTTPS.
- Use HTTPS.

/* means that the HTTPInput node matches against any request that is sent to the HTTP listener on a designated port. This option is useful for testing purposes, but is not suitable for production systems.

You can now deploy the message flow to the broker. If you have completed all the documented steps, message BIP3132 is written to the local system log (on Windows, the event log), stating that the HTTPS listener has been started.

You can now test the system.

Testing your configuration:

About this task

The simplest method of testing whether HTTPS is configured correctly is to use a Web browser to make a request to the broker over HTTPS.

Start a Web browser and enter the following URL:

```
https://localhost:7083/testHTTPS
```

Change values in the URL to reflect the changes that you have made in your broker configuration; for example, the port number. When a window is displayed asking you to accept the certificate, select **Yes**. The browser refreshes the window and displays an empty HTML page:

- In Mozilla browsers, the empty HTML page looks like the following example:

```
<html>  
  <body/>  
</html>
```

- In Internet Explorer, the following information is displayed:

```
XML document must have a top level element. Error processing resource  
'https://localhost:7083/testHTTPS'
```

These responses mean that a blank page was returned, indicating that the setup worked correctly. To add content to the empty page, you can add a Compute node to the flow.

You can use another HTTPS client to process HTTPS requests. Read the documentation for the client to find out how to configure it to make client connections over SSL.

You can also use another HTTPS client, such as a Java or .NET client, instead of the Web browser. Depending on the type of client, you might need to export the certificate (which was created with keytool) from the keystore file associated with the HTTP listener, then import it into the keystore for the client. Read the client documentation to find out how to configure the client to make client connections over SSL.

Related concepts:

“Security overview” on page 203

When you are designing an IBM Integration Bus application it is important to consider the security measures that are needed to protect the information in the system.

 HTTP listeners

You can choose between broker-wide listeners and integration server (embedded) listeners to manage HTTP messages in your HTTP or SOAP flows. Learn about the two types of listener, how ports are assigned to them, and how you can switch from one to the other for individual integration servers.

Related tasks:

“Implementing SSL authentication” on page 415

Use SSL authentication to enhance security in your broker environment.

“Configuring the HTTPRequest and HTTPAsyncRequest nodes to use SSL (HTTPS)”

Configure the HTTPRequest or HTTPAsyncRequest nodes to communicate with other applications that use HTTP over SSL.

Related reference:

 **mqsichangeproperties** command

Use the **mqsichangeproperties** command to modify broker properties and properties of broker resources.

 **mqsireportproperties** command

Use the **mqsireportproperties** command to display properties that relate to a broker, an integration server, or a configurable service.

 HTTPInput node

Use the HTTPInput node to receive an HTTP message from an HTTP client for processing by a message flow.

 HTTPReply node

Use the HTTPReply node to return a response from the message flow to an HTTP client. This node generates the response to an HTTP client from which the input message was received by the HTTPInput node, and waits for confirmation that it has been sent.

 HTTPRequest node

Use the HTTPRequest node to interact with a web service.

Configuring the HTTPRequest and HTTPAsyncRequest nodes to use SSL (HTTPS):

Configure the HTTPRequest or HTTPAsyncRequest nodes to communicate with other applications that use HTTP over SSL.

Before you begin

Before you start: Set up a public key infrastructure (PKI) at broker level: “Setting up a public key infrastructure” on page 415

About this task

This topic describes the steps that you need to follow when configuring an HTTPRequest node on a Windows system. The steps that you must follow on other operating systems are almost identical.

To enable an HTTPRequest node to communicate using HTTP over SSL, an HTTPS server application is required. The information provided in this topic shows how to use the HTTPInput node for SSL as the server application, but the same details also apply when you are using any other server application. You specify the appropriate SSL client Authentication key alias field (Key Alias) on the HTTPRequest node or HTTPAsyncRequest node when the server key store contains multiple certificates for the server.

Complete the following sub-tasks:

1. “Creating a message flow to make HTTPS requests”
2. “Testing your example” on page 377.

Creating a message flow to make HTTPS requests:

About this task

The following message flow creates a generic message flow for converting a WebSphere MQ message into an HTTP Request:

Procedure

1. Create a message flow with the nodes MQInput->HTTPRequest->Compute->MQOutput.
2. On the MQInput node, set the queue name to HTTPS.IN1 and create the WebSphere MQ queue.
3. On the MQOutput node, set the queue name to HTTPS.OUT1 and create the WebSphere MQ queue.
4. On the HTTPRequest node, set the Web Service URL to point to the HTTP server to call. To call the HTTPInput node, use `https://localhost:7083/testHTTPS`.
5. Optional: On the SSL settings properties tab, select Enable certificate revocation list checking. For more information on Certificate Revocation List (CRL) checking see, “Working with certificate revocation lists” on page 426.
6. On the Advanced properties tab of the HTTPRequest node, set the **Response message location in tree** property to `OutputRoot.BLOB`.
7. On the Compute node, add the following ESQL code:

```
CREATE COMPUTE MODULE test_https_Compute
CREATE FUNCTION Main() RETURNS BOOLEAN
BEGIN
  -- CALL CopyMessageHeaders();
  CALL CopyEntireMessage();
  set OutputRoot.HTTPResponseHeader = null;
  RETURN TRUE;
END;
```

```

CREATE PROCEDURE CopyMessageHeaders() BEGIN
    DECLARE I INTEGER;
    DECLARE J INTEGER;
    SET I = 1;
    SET J = CARDINALITY(InputRoot.*[]);
    WHILE I < J DO
        SET OutputRoot.*[I] = InputRoot.*[I];
        SET I = I + 1;
    END WHILE;
END;

CREATE PROCEDURE CopyEntireMessage() BEGIN
    SET OutputRoot = InputRoot;
END;
END MODULE;

```

What to do next

The message flow is now ready to be deployed to the broker and tested.

Testing your example:

About this task

To test that the example works, complete the following steps:

Procedure

1. Follow the instructions in “Configuring HTTPInput and HTTPReply nodes to use SSL (HTTPS)” on page 372, including testing the example.
2. Deploy the HTTPRequest message flow.
3. Put a message to the WebSphere MQ queue HTTPS.IN1. If successful, a message appears on the output queue. If the process fails, an error appears in the local error log (which is the event log on Windows).

Related concepts:

“Security overview” on page 203

When you are designing an IBM Integration Bus application it is important to consider the security measures that are needed to protect the information in the system.

Related tasks:

“Implementing SSL authentication” on page 415

Use SSL authentication to enhance security in your broker environment.

“Configuring HTTPInput and HTTPReply nodes to use SSL (HTTPS)” on page 372

Configure the HTTPInput and HTTPReply nodes to communicate with other applications that use HTTPS by creating a keystore file, configuring the broker or integration server to use SSL, and creating a message flow to process HTTPS requests.



Creating a message flow

Create a message flow to specify how to process messages in the broker. You can create one or more message flows and deploy them to one or more brokers.



Defining message flow content

You can define message flow content by adding and configuring message flow nodes and connecting them to form flows.



Creating ESQL for a node

Create ESQL code to customize the behavior of a Compute, Database, DatabaseInput, or Filter node in an ESQL file.



Deploying resources

Deploy message flow applications to integration servers by sending a broker archive (BAR) file to a broker, which unpacks and stores the contents ready for when your message flows are started.

Related reference:



`mqsichangeproperties` command

Use the `mqsichangeproperties` command to modify broker properties and properties of broker resources.



`mqsireportproperties` command

Use the `mqsireportproperties` command to display properties that relate to a broker, an integration server, or a configurable service.



HTTPRequest node

Use the HTTPRequest node to interact with a web service.



HTTPAsyncRequest node

Use the HTTPAsyncRequest node with the HTTPAsyncResponse node to construct a pair of message flows that interact with a Web service asynchronously.



HTTPInput node

Use the HTTPInput node to receive an HTTP message from an HTTP client for processing by a message flow.



HTTPReply node

Use the HTTPReply node to return a response from the message flow to an HTTP client. This node generates the response to an HTTP client from which the input message was received by the HTTPInput node, and waits for confirmation that it has been sent.

Securing the connection to CICS Transaction Server for z/OS by using SSL:

Configure the CICSRequest node to communicate with CICS Transaction Server for z/OS over the Secure Sockets Layer (SSL) protocol by updating a CICSConnection configurable service or the CICSRequest node to use SSL.

Before you begin

Before you start:

Ensure that you have completed the following tasks:

1. The CICSRequest node does not support a separate truststore, so the keystore file must provide both personal and signer certificates. If client-authentication (CLIENTAUTH) is enabled in the TCPIPService in CICS, the broker keystore file must also contain a personal certificate that is trusted by CICS. To set up a public key infrastructure (PKI) at broker or integration server level, follow the instructions in “Setting up a public key infrastructure” on page 415.
2. Define the COMMAREA data structure as a message set, as described in Defining a CICS Transaction Server for z/OS data structure.

3. Configure IP InterCommunications (IPIC) protocol on CICS, as described in Preparing the environment for the CICSRequest node.

About this task

To configure the CICSRequest node to use SSL, complete the following steps:

Procedure

1. For client-authenticated (CLIENTAUTH) SSL connections, CICS expects the SSL client certificate to be mapped to a RACF user ID. Therefore the SSL client certificate must be mapped to a RACF user ID before attempting to establish the SSL connection to CICS. If the client certificate is not mapped to a RACF user ID, the broker might display a ECI_ERR_NO_CICS response. You can map a client certificate to a RACF user ID by using the RACF command **RACDCERT**, which stores the client certificate in the RACF database and associates a user ID with it, or by using RACF certificate name filtering. Client certificates can be mapped one-to-one with a user ID, or a mapping from one to the other can be provided to allow a many-to-one mapping. You can achieve this mapping by using one of the following methods:

- **Associating a client certificate with a RACF user ID**

- a. Copy the certificate that you want to process into an MVS™ sequential file. The file must have variable length, blocked records (RECFM=VB), and be accessible from TSO.
- b. Run the **RACDCERT** command in TSO by using the following syntax:

```
RACDCERT ADD('datasetname') TRUST [ ID(userid) ]
```

Where:

- *datasetname* is the name of the data set containing the client certificate.
- *userid* is the user ID to be associated with the certificate. This parameter is optional. If omitted, the certificate is associated with the user issuing the **RACDCERT** command.

When you issue the **RACDCERT** command, RACF creates a profile in the DIGTCERT class. This profile associates the certificate with the user ID. You can then use the profile to translate a certificate to a user ID without giving a password. For full details of RACF commands, see z/OS Security Server RACF Command Language Reference.

- **RACF certificate name filtering**

With certificate name filtering, client certificates are not stored in the RACF database. The association between one or more certificates and a RACF user ID is achieved by defining a filter rule that matches the distinguished name of the certificate owner or issuer (CA). A sample filter rule might look like the following example:

```
RACDCERT ID(DEPT3USR) MAP SDNFILTER  
(OU=DEPT1.OU=DEPT2.O=IBM.L=LOC.SP=NY.C=US)
```

This sample filter rule would associate user ID DEPT3USR with all certificates when the distinguished name of the certificate owner contains the organizational unit DEPT1 and DEPT2, the organization IBM, the locality LOC, the state/province NY, and the country US.

2. Turn on SSL support in the broker by setting the `cicsServer` property on the CICSConnection configurable service, as shown in the following example. This example changes the CICSRequest node that is configured to use the

myCICSConnection configurable service for the CICS instance that is running at *mycicsregion.com* port 56789. After you run this command, the CICSRequest node connects to CICS over SSL.

```
mqsichangeproperties IB9NODE -c CICSConnection -o myCICSConnection -n cicsServer -v ssl://mycicsregion.com:56789
```

Alternatively you can configure the CICS server property directly on the CICSRequest node.

What to do next

Next: When you have configured the broker or the CICSRequest node to use SSL, develop a message flow that contains a CICSRequest node by following the steps in Developing a message flow with a CICSRequest node.

Related concepts:

“Security overview” on page 203

When you are designing an IBM Integration Bus application it is important to consider the security measures that are needed to protect the information in the system.



CICS Transaction Server for z/OS connectivity

Use the CICSRequest node to connect IBM Integration Bus with CICS Transaction Server for z/OS applications.



Using the IBM Integration Explorer to work with configurable services

Configurable services are used to define properties that are related to external services on which the integration node (broker) relies. Use the IBM Integration Explorer to view, add, modify and delete configurable services.

Related tasks:

“Implementing SSL authentication” on page 415

Use SSL authentication to enhance security in your broker environment.



Changing connection information for the CICSRequest node

You can create a configurable service that the CICSRequest node or message flow refers to at run time for connection information, instead of defining the connection properties on the node or the message flow. The advantage being that you can change the host name and performance values without needing to redeploy your message flow.



Creating a message flow

Create a message flow to specify how to process messages in the broker. You can create one or more message flows and deploy them to one or more brokers.

Related reference:



Configurable services properties

The supplied configurable services, and the configurable services that you create, are defined by their names and properties. You can use the supplied services.



mqsicreateconfigurable-service command

Use the **mqsicreateconfigurable-service** command to create an object name for a broker external resource.



mqsichangeproperties command

Use the **mqsichangeproperties** command to modify broker properties and properties of broker resources.

mqsisetdbparms command

Use the **mqsisetdbparms** command to associate a specific user ID and password (or SSH identity file) with one or more resources that are accessed by the broker.

CICSRequest node

Use the CICSRequest node to call CICS Transaction Server for z/OS programs over TCP/IP-based IP InterCommunications (IPIC) protocol.

Securing the connection to IMS by using SSL:

Configure the IMSRequest node to communicate with IMS over the Secure Sockets Layer (SSL) protocol by creating a keystore file, and configuring the broker to use SSL.

Before you begin

Before you start:

Set up a public key infrastructure (PKI) at broker level by following the instructions in “Setting up a public key infrastructure” on page 415.

About this task

To configure the IMSRequest node to use SSL, complete the following steps.

Procedure

1. Turn on SSL support in the broker by setting the `UseSSL` and `SSLEncryptionType` properties on the `IMSConnect` configurable service, as shown in the following example.

This example changes the IMSRequest node that is configured to use the `myIMSConnectService` configurable service. After you run this command, the IMSRequest node connects to IMS over SSL.

```
mqsichangeproperties IB9NODE -c IMSConnect -o myIMSConnectService -n UseSSL,SSLEncryptionType -v True,Weak
```

2. Optional: Develop a message flow that contains a IMSRequest node.
3. Test your configuration.


Related concepts:

“Security overview” on page 203

When you are designing an IBM Integration Bus application it is important to consider the security measures that are needed to protect the information in the system.

IMS connections

Open Transaction Manager Access (OTMA) is used to provide access to IMS from IBM Integration Bus.

 Using the IBM Integration Explorer to work with configurable services
Configurable services are used to define properties that are related to external services on which the integration node (broker) relies. Use the IBM Integration Explorer to view, add, modify and delete configurable services.

Related tasks:

“Implementing SSL authentication” on page 415

Use SSL authentication to enhance security in your broker environment.



Changing connection information for the IMSRequest node

You can create a configurable service that the IMSRequest node or message flow refers to at run time for connection information, instead of defining the connection properties on the node or the message flow. The advantage being that you can change the host name, performance, and security values without needing to redeploy your message flow.



Creating a message flow

Create a message flow to specify how to process messages in the broker. You can create one or more message flows and deploy them to one or more brokers.

Related reference:



Configurable services properties

The supplied configurable services, and the configurable services that you create, are defined by their names and properties. You can use the supplied services.



mqsicreateconfigurable-service command

Use the **mqsicreateconfigurable-service** command to create an object name for a broker external resource.



mqsichangeproperties command

Use the **mqsichangeproperties** command to modify broker properties and properties of broker resources.



IMSRequest node

Use the IMSRequest node to send a request to run a transaction on a local or remote IMS system, and wait for a response. IMS Connect must be configured and running on the IMS system.

SSL and the TCP/IP nodes:

Configure a TCP/IP configuration to use SSL to secure connectivity to and from the TCPIP nodes.

Related tasks:

“Configuring TCP/IP client nodes to use SSL”

Configure a TCP/IP configuration to use SSL to secure connectivity to and from the TCPIP client nodes.

“Configuring TCP/IP server nodes to use SSL” on page 384

Configure a TCP/IP configuration to use SSL to secure connectivity to and from the TCPIP server nodes.

“Implementing SSL authentication” on page 415

Use SSL authentication to enhance security in your broker environment.

Configuring TCP/IP client nodes to use SSL:

Configure a TCP/IP configuration to use SSL to secure connectivity to and from the TCPIP client nodes.

You can create or modify TCP/IP client connections that use SSL, by creating or modifying a configurable service. You can specify the type of protocol, and the allowed cipher suites. By default, SSL is not enabled for any configurable services. The nodes use the standard broker keystore and truststore configuration.

Before you begin

Before you start: Set up a public key infrastructure (PKI) at broker or integration server level by following the instructions in “Setting up a public key infrastructure” on page 415.

About this task

Follow these steps to configure the TCPIP nodes to use SSL:

1. Optional: “Changing a TCP/IP client configuration to use SSL”
2. Optional: “Creating a TCP/IP client configuration that uses SSL”

Changing a TCP/IP client configuration to use SSL:

About this task

Use the **mqsichangeproperties** command to change an existing TCPIPClient configurable service.

Procedure

1. The following command specifies that the myTCPIPClientService configurable service must use SSLv3 as the protocol, with any available cipher suite.

```
mqsichangeproperties MYBROKER
  -c TCPIPClient
  -o myTCPIPClientService
  -n SSLProtocol
  -v SSLv3
```

2. Restart the integration server that contains the message flows.

Creating a TCP/IP client configuration that uses SSL:

About this task

Use the **mqsicreateconfigurable-service** command to create a TCPIPClient configurable service.

Procedure

1. The following command creates a TCPIPClient configurable service for making connections on port 1455 on the local machine. It uses the SSL protocol SSLv3 with a specific list of allowed cipher suites.

```
mqsicreateconfigurable-service MYBROKER
  -c TCPIPClient
  -o myTCPIPClientService
  -n Port,Hostname,SSLProtocol,SSLCiphers
  -v 1455,localhost,SSLv3,SSL_RSA_WITH_RC4_128_MD5;
  SSL_RSA_WITH_3DES_EDE_CBC_SHA
```

2. Restart the integration server that contains the message flows.

Testing your configuration:

About this task

Use either a TCPIPClientInput node, or a TCPIPClientOutput node to open a connection to a remote SSL server application that is listening on a TCP/IP port.

Related concepts:

“SSL and the TCP/IP nodes” on page 382

Configure a TCP/IP configuration to use SSL to secure connectivity to and from

the TCPIP nodes.

“Security overview” on page 203

When you are designing an IBM Integration Bus application it is important to consider the security measures that are needed to protect the information in the system.

Related tasks:

“Implementing SSL authentication” on page 415

Use SSL authentication to enhance security in your broker environment.

“Configuring TCP/IP server nodes to use SSL”

Configure a TCP/IP configuration to use SSL to secure connectivity to and from the TCPIP server nodes.

Related reference:



mqsichangeproperties command

Use the **mqsichangeproperties** command to modify broker properties and properties of broker resources.



mqsireportproperties command

Use the **mqsireportproperties** command to display properties that relate to a broker, an integration server, or a configurable service.



Configurable services properties

The supplied configurable services, and the configurable services that you create, are defined by their names and properties. You can use the supplied services.

Configuring TCP/IP server nodes to use SSL:

Configure a TCP/IP configuration to use SSL to secure connectivity to and from the TCPIP server nodes.

You can create or modify TCP/IP server connections that use SSL, by creating or modifying a configurable service. You can specify:

- The type of protocol.

- The allowed cipher suites.

- A key alias.

- Whether a connecting client should provide authentication information.

By default, SSL is not enabled for any configurable services. The nodes use the standard broker keystore and truststore configuration.

Before you begin

Before you start: Set up a public key infrastructure (PKI) at broker level by following the instructions in “Setting up a public key infrastructure” on page 415.

About this task

Follow these steps to configure the TCPIP nodes to use SSL:

1. Optional: “Changing a TCP/IP server configuration to use SSL”
2. Optional: “Creating a TCP/IP server configuration that uses SSL” on page 385

Changing a TCP/IP server configuration to use SSL:

About this task

Use the **mqsichangeproperties** command to change an existing TCPIPService configurable service.

Procedure

1. The following command changes a TCPIPService configurable service to use SSLv3 with any available cipher suite. Connecting clients are not asked to authenticate.

```
mqsichangeproperties MYBROKER
-c TCPIPClient
-o myTCPIPService
-n SSLProtocol
-v SSLv3
```

2. Restart the integration server that contains the message flows.

Creating a TCP/IP server configuration that uses SSL:

About this task

Use the **mqsicreateconfigurable-service** command to create a TCPIPService configurable service.

Procedure

1. The following command creates a TCPIPService configurable service for making connections on port 1455. It uses the SSL protocol SSLv3 with a specific list of allowed cipher suites. Connecting clients are required to authenticate.

```
mqsicreateconfigurable-service MYBROKER
-c TCPIPService
-o myTCPIPService
-n Port,SSLProtocol,SSLCiphers,SSLClientAuth
-v 1455,SSLv3,SSL_RSA_WITH_RC4_128_MD5;
  SSL_RSA_WITH_3DES_EDE_CBC_SHA,require
```

2. Restart the integration server that contains the message flows.

Using an SSL key alias:

About this task

A key alias identifies the key that is to be used for the SSL connection, if the keystore for your broker or integration server contains more than one key. Use the **mqsichangeproperties** or **mqsicreateconfigurable-service** as appropriate, with the `SSLKeyAlias` property. The default value "" or none, means that an SSL key alias is not used. Any other string identifies the alias.

Note: If the keystore contains more than one key, and no key alias is defined, the Java virtual machine arbitrarily chooses a key at run time.

The following command creates a TCPIPService configurable service for making connections on port 1455. It uses the SSL protocol SSLv3 with the cipher suites `SSL_RSA_WITH_RC4_128_MD5` and `SSL_RSA_WITH_3DES_EDE_CBC_SHA`. It requires the client to authenticate itself, and uses the key alias `MyKey` to identify the key to be used.

```
mqsicreateconfigurableservice MYBROKER
-c TCPIPServer
-o myTCPIPService
-n Port,SSLProtocol,SSLCiphers,SSLClientAuth,SSLKeyAlias
-v 1455,SSLv3,SSL_RSA_WITH_RC4_128_MD5;SSL_RSA_WITH_3DES_EDE_CBC_SHA
,require,MyKey
```

The following command changes a TCPIPService configurable service to use the first key retrieved from the keystore, with SSL protocol SSLv3. SSLClientAuth is disabled.

```
mqsichangeproperties MYBROKER
-c TCPIPClient
-o myTCPIPService
-n SSLProtocol
-v SSLv3
```

Testing your configuration:

About this task

To test your configuration, connect an SSL-enabled client, such as another program, or a web browser, to the server port. Connection error messages, such as handshake failures, or untrusted keys, indicate that you must change the configuration.

Client identity:

About this task

If SSL client authentication is requested or required, and the client successfully authenticates, the distinguished name is present as an identity source token in the properties parser, in the tree propagated from the Open terminal at connection time. This applies only to the TCPIPServiceInput node.

The IdentitySourceToken field is set to the distinguished name from the client certificate.

The IdentitySourceType field is set to the string username.

The IdentitySourceIssuedBy field is set to the issuer of the certificate presented by the client.

If SSL client authentication is requested, and the client does not provide the required credentials, the fields are set to blank.

Related concepts:

“Security overview” on page 203

When you are designing an IBM Integration Bus application it is important to consider the security measures that are needed to protect the information in the system.

Related tasks:

“Implementing SSL authentication” on page 415

Use SSL authentication to enhance security in your broker environment.

Related reference:



mqsichangeproperties command

Use the **mqsichangeproperties** command to modify broker properties and properties of broker resources.



mqsireportproperties command

Use the **mqsireportproperties** command to display properties that relate to a

broker, an integration server, or a configurable service.



Configurable services properties

The supplied configurable services, and the configurable services that you create, are defined by their names and properties. You can use the supplied services.

Securing integration services by using SSL configuration

You can secure integration services by configuring the SOAP/HTTP binding or JavaScript client API to use SSL and certificates.

Before you begin

Define a public key infrastructure (PKI) for IBM Integration Bus; see “Setting up a public key infrastructure” on page 415.

About this task

After you establish a public key infrastructure configuration for your whole broker or for some of its integration servers, you can use the configuration to secure your integration services by completing the following steps:

1. Optional: If you are using the broker listener: Configure the broker to use SSL.
2. Optional: If you are using the embedded (integration server) listener: Configure the integration server to use SSL.
3. Configure the integration service bindings to use SSL

Configuring the broker to use SSL:

About this task

Complete the following steps:

Procedure

1. Turn on SSL support in the broker, by setting a value for **enableSSLConnector**

```
mqschangeproperties broker name  
-b httplistener -o HTTPListener  
-n enableSSLConnector -v true
```

2. Optional: If you do not want to use the default port 7083 for HTTPS messages, specify the **port** on which the broker listens:

```
mqschangeproperties broker name  
-b httplistener -o HTTPSConnector  
-n port -v Port to listen on for https
```

On UNIX systems, only processes that run under a privileged user account (in most cases, root) can bind to ports lower than 1024. For the broker to listen on these ports, the user ID under which the broker is started must be root.

3. Optional: Enable Client Authentication (mutual authentication):

```
mqschangeproperties broker_name -b httplistener -o HTTPSConnector  
-n clientAuth -v true
```

4. Restart the broker after changing one or more of the HTTP listener properties.
5. Optional: Use the following commands to display HTTP listener properties:

```
mqsireportproperties broker_name -b httplistener -o AllReportableEntityNames -a  
mqsireportproperties broker_name -b httplistener -o HTTPListener -a  
mqsireportproperties broker_name -b httplistener -o HTTPSConnector -a
```

Configuring an integration server to use SSL:

About this task

Complete the following steps:

Procedure

1. Optional: Specify a specific port on which the integration server listens for HTTPS requests, or leave the value unset to use the next available port number.

```
mqsichangeproperties broker_name  
-e integration_server_name -o HTTPSConnector  
-n explicitlySetPortNumber -v port_number
```

On UNIX systems, only processes that run under a privileged user account (in most cases, root) can bind to ports lower than 1024. For the integration server to listen on these ports, the user ID under which the broker is started must be root.

If you do not complete this step, the first available port in the default range (7843 - 7884) is used.

2. Optional: Enable Client Authentication (mutual authentication):

```
mqsichangeproperties broker_name  
-e integration_server_name -o HTTPSConnector  
-n clientAuth -v true
```

3. Optional: Change the SSL protocol. The default protocol for the HTTPInput node is TLS. Run the following command to change it to SSL:

```
mqsichangeproperties broker_name  
-e integration_server_name -o HTTPSConnector  
-n sslProtocol -v SSL
```

4. Restart the broker after changing one or more of the listener properties.
5. Optional: Use the following command to display HTTPS properties:

```
mqsireportproperties broker_name  
-e integration_server_name -o HTTPSConnector -r
```

Configuring the integration service bindings to use SSL:

About this task

Configure the integration service bindings to use SSL by completing the following steps:

Procedure

1. In the IBM Integration Toolkit, open your integration service in the integration service editor by double-clicking **Integration Service Description** in the Application Development view.
2. Click the **Service** tab. The integration service description is displayed, which includes the integration service bindings.
3. If you are using the SOAP/HTTP binding, then click **SOAP/HTTP Binding** and select **Use HTTPS** from the HTTP Transport properties panel.
4. If you are using the JavaScript client API, then click **JavaScript client API** and then select **Use HTTPS** from the Basic properties panel.

Note: If you are using a web browser-based JavaScript application to call the integration service, then you must select **Use HTTPS** on both the SOAP/HTTP binding and the JavaScript client API. The HTTP proxy servlet routes requests only to endpoints that use the same protocol as the web browser. The HTTP

- proxy servlet routes requests to both the SOAP and JavaScript client API endpoints, and so both endpoints must match the web browser protocol.
5. Save and redeploy the integration service.

Results

You have configured the integration service to use SSL.

Related concepts:



Integration service JavaScript client API

An integration service developer can generate a JavaScript client API from an existing integration service. The JavaScript client API provides operation functions that a JavaScript developer can call from an application that is running in a JavaScript environment.

Related tasks:

“Implementing SSL authentication” on page 415

Use SSL authentication to enhance security in your broker environment.

Configuring for identity propagation

To enable a message flow to perform identity propagation, the input nodes must extract the identity and the output node must propagate it.

Before you begin

Before you start:

Before you can configure a message flow to perform identity propagation, you must check that an appropriate security profile exists, or create a new security profile. See “Creating a security profile” on page 294.

About this task

An input node extracts security tokens if it is configured with a security profile at deployment time. An output node propagates an identity if it is configured with a security profile that enables propagation at deployment time.

To enable a message flow to perform identity propagation, complete the following steps.

Procedure

By using the Broker Archive editor, select a security profile that has identity propagation enabled. You can use the Default Propagation profile, which is a predefined profile that requests only identity propagation. You can set a security profile on a message flow or on individual input and output nodes. If no security profile is set for the input and output nodes, the setting is inherited from the setting on the message flow.

1. In the Application Development view, right-click the BAR file, then click **Open with > Broker Archive Editor**.
2. Click the **Manage and Configure** tab.
3. Click the flow or node on which you want to set the security profile. The properties that you can configure for the message flow or for the node are displayed in the **Properties** view.

4. In the **Security Profile Name** field, select a security profile that has identity propagation enabled.
5. Save the BAR file.

What to do next

For a SOAPRequest or SOAPAsyncRequest node, you can define an appropriate policy set and bindings to specify how the propagated identity is placed in the WS-Security header (rather than the underlying transport headers). For more information, see Policy sets.

On SOAPRequest and SOAPAsyncRequest nodes, only Username and SAML tokens can be propagated. However, on the SOAPRequest and SOAPAsyncRequest nodes with a Kerberos policy set and bindings, a Username and password token can be propagated into the node to provide the Kerberos client credentials.

For the SAPRequest node, you can propagate only the user name and password. For the CICSRequest and IMSRequest nodes, you can propagate the user name, or the user name and password.

If the message identity does not contain enough information for identity propagation, you can use any of the following methods to acquire the necessary information:

- Take the information from the message body. For example, if the message comes from WebSphere MQ with only a username token, and the output is an HTTP request node requiring a Username + Password token, the password might be present in the body of the incoming message. For more information, see “Configuring the extraction of an identity or security token” on page 308.
- Configure an identity mapper using TFIM. For more information, see the IBM Tivoli Federated Identity Manager product documentation.
- Use ESQL or Java to set the Mapped Identity fields in the Properties tree.

Related concepts:

“Identity” on page 251

In IBM Integration Bus, an identity is a security token that uniquely identifies an individual, or that provides a set of assertions that can be validated.

“Authentication and validation” on page 257

Authentication is the process of establishing the identity of a user or system and verifying that the identity is valid. Applying authentication to a SAML security token involves validating the assertions that it carries and confirming that it is being processed within its validity period.

“Identity mapping” on page 263

Identity mapping is the transformation of a security token from one format to another format, or the federation of an identity from one realm to an equivalent identity in another realm.

“Authorization” on page 261

Authorization is the process of verifying that an identity token has permission to access a message flow.

“Identity and security token propagation” on page 286

Identity and security token propagation enables the identity and security tokens (associated with each message) to be propagated throughout a message flow, and on to target applications through output or request nodes.

“Security profiles” on page 247

A security profile defines the security operations that are to be performed in a

message flow at SecurityPEP nodes and security enabled input and output nodes. “Security exception processing” on page 290
A security exception is raised when a message flow security failure occurs during security processing in an input node or SecurityPEP node.

Related tasks:

“Configuring the extraction of an identity or security token” on page 308

You can configure the SecurityPEP node or security enabled input nodes to extract the identity or security token from a message and store it in the properties tree identity fields, enabling it to be processed throughout the message flow and propagated at output or request nodes.

“Configuring identity authentication and security token validation” on page 312

You can configure a message flow to perform identity authentication or security token validation using Integrated Windows Authentication (IWA), Lightweight Directory Access Protocol (LDAP), a WS-Trust V1.3 compliant security token sever (STS) such as Tivoli Federated Identity Manager (TFIM) Version 6.2, or HTTP requests. Support for TFIM V6.1 is also provided, for compatibility with previous versions of IBM Integration Bus.

“Configuring identity mapping” on page 333

Configure a security token server (STS), such as Tivoli Federated Identity Manager (TFIM) V6.2, to map the incoming security token and, if required, to authenticate and authorize it.

“Creating a security profile” on page 294

You can create a security profile for use with Lightweight Directory Access Protocol (LDAP) or a WS-Trust V1.3 compliant security token server (STS), such as Tivoli Federated Identity Manager (TFIM) V6.2. Support is also provided for TFIM V6.1, for compatibility with previous versions of IBM Integration Bus. You can create the security profile by using either the **mqscreateconfigurable** command or an editor in the IBM Integration Explorer.

“Configuring authorization” on page 342

Configure the broker to use an external security provider to authorize an identity in a message flow. You can use either Lightweight Directory Access Protocol (LDAP) or a WS-Trust V1.3 compliant security token server (STS) such as Tivoli Federated Identity Manager (TFIM) V6.2. Support for TFIM V6.1 is also provided for compatibility with previous versions.

“Setting up message flow security” on page 292

Set up security on a message flow to control access based on the identity of a message passing through the message flow.

Related reference:

 **mqscreateconfigurable** command

Use the **mqscreateconfigurable** command to create an object name for a broker external resource.

 **mqsdeleteconfigurable** command

Use the **mqsdeleteconfigurable** command to delete a configurable service, such as a JMS provider, JDBC provider, or FTP server, that you have created by using the **mqscreateconfigurable** command.

 **mqschangeproperties** command

Use the **mqschangeproperties** command to modify broker properties and properties of broker resources.

 **mqsreportproperties** command

Use the **mqsreportproperties** command to display properties that relate to a

broker, an integration server, or a configurable service.



MQInput node

Use the MQInput node to receive messages from clients that connect to the broker by using the WebSphere MQ Enterprise Transport, and that use the MQI and AMI application programming interfaces.



HTTPInput node

Use the HTTPInput node to receive an HTTP message from an HTTP client for processing by a message flow.



SOAPInput node

Use the SOAPInput node to process client SOAP messages, so that the broker operates as a SOAP Web Services provider.



SOAPAsyncRequest node

Use the SOAPAsyncRequest node with the SOAPAsyncResponse node to construct a pair of message flows that call a Web service asynchronously.



HTTPRequest node

Use the HTTPRequest node to interact with a web service.



MQOutput node

Use the MQOutput node to send messages to clients that connect to the broker using the WebSphere MQ Enterprise Transport and that use the MQI and AMI application programming interfaces.

Checking the security credentials that are used by an integration node

To check what security credentials are set on an integration node that is connected to a remote system or database, use the `mqsireportdbparms` command.

About this task

If you have an integration node that accesses external resources, you can check what security credentials are set if a security identity was created for the integration node. For more information, see “Security identities for integration nodes connecting to external systems” on page 289. If security credentials are not set, then no results are returned.

You can also check whether a password is valid for a set of security credentials. You might want to check a password if you recently updated a password while the integration node was running, to see if the new password is effective.

To review the security credentials that are set for an integration node, complete the following steps.

Procedure

1. Run the `mqsireportdbparms` command. A list of integration nodes that have parameters that are set is returned. You can then see whether user IDs are assigned to the associated security profile for each of the results.

In the previous step, you can identify which integration nodes have security profiles. If you want to check that you have a valid password for a specific user ID:

2. Run the `mqsireportdbparms` command with the name of the integration node that you want to check in the following format:

```
mqsireportdbparms -n dsn::xxxx -p yyyy
```

Where xxxx represents the user ID, and yyyy represents the password that you want to check is valid for the integration node.

Results

A result is returned that confirms whether the password that was provided is correct, or if it was incorrect.

Additionally, the command returns if the integration node was restarted after a password change, so that you know what password is valid for the integration node while it is running.

Related concepts:

“Identity” on page 251

In IBM Integration Bus, an identity is a security token that uniquely identifies an individual, or that provides a set of assertions that can be validated.

“Identity mapping” on page 263

Identity mapping is the transformation of a security token from one format to another format, or the federation of an identity from one realm to an equivalent identity in another realm.

“Authorization” on page 261

Authorization is the process of verifying that an identity token has permission to access a message flow.

“Identity and security token propagation” on page 286

Identity and security token propagation enables the identity and security tokens (associated with each message) to be propagated throughout a message flow, and on to target applications through output or request nodes.

“Security profiles” on page 247

A security profile defines the security operations that are to be performed in a message flow at SecurityPEP nodes and security enabled input and output nodes.

Related tasks:

“Configuring the extraction of an identity or security token” on page 308

You can configure the SecurityPEP node or security enabled input nodes to extract the identity or security token from a message and store it in the properties tree identity fields, enabling it to be processed throughout the message flow and propagated at output or request nodes.

“Configuring identity authentication and security token validation” on page 312

You can configure a message flow to perform identity authentication or security token validation using Integrated Windows Authentication (IWA), Lightweight Directory Access Protocol (LDAP), a WS-Trust V1.3 compliant security token server (STS) such as Tivoli Federated Identity Manager (TFIM) Version 6.2, or HTTP requests. Support for TFIM V6.1 is also provided, for compatibility with previous versions of IBM Integration Bus.

“Configuring identity mapping” on page 333

Configure a security token server (STS), such as Tivoli Federated Identity Manager (TFIM) V6.2, to map the incoming security token and, if required, to authenticate and authorize it.

“Creating a security profile” on page 294

You can create a security profile for use with Lightweight Directory Access Protocol (LDAP) or a WS-Trust V1.3 compliant security token server (STS), such as Tivoli Federated Identity Manager (TFIM) V6.2. Support is also provided for TFIM V6.1, for compatibility with previous versions of IBM Integration Bus. You can create the security profile by using either the **mqsicreateconfigurable** service

command or an editor in the IBM Integration Explorer.

“Configuring authorization” on page 342

Configure the broker to use an external security provider to authorize an identity in a message flow. You can use either Lightweight Directory Access Protocol (LDAP) or a WS-Trust V1.3 compliant security token server (STS) such as Tivoli Federated Identity Manager (TFIM) V6.2. Support for TFIM V6.1 is also provided for compatibility with previous versions.

Related reference:



mqsireportdbparms command

Use the **mqsireportdbparms** command to list all parameters that are set for a specific broker.

Diagnosing security problems

This topic explains how to find out why access to a secured flow has been denied.

About this task

By default, security exceptions occurring in an input node are not processed in the same way as other errors (see “Security exception processing” on page 290). Security exceptions are not logged to the system event log, to prevent a security denial of service attack filling the logs and destabilizing the system.

This means that, by default, you cannot diagnose input node security exceptions in the same way as other errors. However, in a SecurityPEP node, a failing security operation causes a security exception to be raised, wrapped in a normal recoverable exception, which invokes the error handling that is provided by the message flow.

To see what might be causing the security exceptions, you can do either of the following things:

- Select the **Treat Security Exceptions as normal exceptions** property on the input nodes.
- Use the user trace.

The following steps show you how to use the user trace to find out why access to a secured message flow has been denied:

Procedure

1. Use the **mqsireloadsecurity** command to clear the security cache, so that the traced request goes to the security provider rather than using a result held in the cache. This ensures that the reason codes returned from the security provider are displayed in the traced exception.
2. Enable user trace for the message flow, using either the workbench or the **mqsichangeTrace** command (see Starting user trace for more information).
3. Resend the request that has been rejected by the security provider.
4. Stop the user trace, using either the workbench or the **mqsichangeTrace** command.
5. Use the **mqsireadlog** command to examine the trace information that was recorded by the user trace. This trace information contains the error codes provided by the broker and the security provider.

Related concepts:

“Security exception processing” on page 290

A security exception is raised when a message flow security failure occurs during security processing in an input node or SecurityPEP node.

“Message flow security overview” on page 243

IBM Integration Bus provides a security manager, which enables you to control access to individual messages in a message flow, using the identity of the message.

Related tasks:

“Setting up message flow security” on page 292

Set up security on a message flow to control access based on the identity of a message passing through the message flow.



Starting user trace

Use user trace for debugging your applications; you can trace brokers, integration servers, and deployed message flows. Start user trace facilities using the **mqsichangetrace** command or the IBM Integration Explorer.



Stopping user trace

Use user trace for debugging your applications; you can trace brokers, integration servers, and deployed message flows. Stop user trace facilities by using the **mqsichangetrace** command or the IBM Integration Explorer.



Retrieving user trace

Use the **mqsireadlog** command to access the trace information that is recorded by the user trace facilities.

“Creating a security profile” on page 294

You can create a security profile for use with Lightweight Directory Access Protocol (LDAP) or a WS-Trust V1.3 compliant security token server (STS), such as Tivoli Federated Identity Manager (TFIM) V6.2. Support is also provided for TFIM V6.1, for compatibility with previous versions of IBM Integration Bus. You can create the security profile by using either the **mqsicreateconfigurable-service** command or an editor in the IBM Integration Explorer.

Related reference:



mqsireloadsecurity command

Use the **mqsireloadsecurity** command to force the immediate expiry of some or all the entries in the security cache.



mqsichangetrace command

Use the **mqsichangetrace** command to set the tracing characteristics for a broker.



mqsireadlog command

Use the **mqsireadlog** command to retrieve trace records for the specified component.



Parameter values for the securitycache component

Select the objects and properties associated with the securitycache component that you want to change.



SecurityPEP node

Use the SecurityPEP node in a message flow to invoke the message flow security manager at any point in the message flow.

Integration Bus server security

You must consider several security aspects when you are setting up brokers running on Windows, Linux, z/OS, or UNIX platforms.

Before you begin

- Read “Security overview” on page 203 for an introduction to various aspects of security.
- Refer to Planning for security, which contains links to security information that you need before, during, and after installation of IBM Integration Bus.

About this task

Use the following list of tasks as a security checklist:

- “Creating user IDs” on page 397
- “Security for the IBM Integration Toolkit and IBM Integration Explorer” on page 398
- “Considering security for a broker” on page 407
- “Implementing SSL authentication” on page 415
- “Using security exits” on page 399
- “Configuring broker user IDs on z/OS” on page 409
- “Specifying an alternative user ID to run an integration server on z/OS” on page 413

Related concepts:

“Security for the IBM Integration Toolkit and IBM Integration Explorer” on page 398

What to consider when you set up security for the IBM Integration Toolkit and IBM Integration Explorer.

“Integration server user IDs on z/OS” on page 411

On z/OS, you can specify an alternative user ID to run an integration server so that it accesses resources according to the permissions assigned to it, rather than the permissions assigned to the main broker user ID.

Related tasks:



Planning for security

It is important to be aware of security issues when you are planning to install and use IBM Integration Bus.

“Considering security for a broker” on page 407

Consider several factors when you are deciding which users can execute broker commands, and which users can control security for other broker resources.

“Configuring broker user IDs on z/OS” on page 409

When you create your brokers on z/OS, you must set up security by configuring broker user IDs with the appropriate permissions.

Related reference:



Security requirements for Windows systems

Security requirements depend on the administrative task that you want to perform.



Security requirements for Linux and UNIX platforms

View a summary of the authorizations in a Linux or UNIX environment.



Summary of required access (z/OS)

The professionals in your organization require access to components and resources

on z/OS.

Creating user IDs

When you plan the administration of your broker configuration, you might have to define one or more user IDs for the tasks associated with particular roles.

About this task

Some operating systems, and other products, impose restrictions on user IDs:

- On Windows systems, user IDs can be up to 12 characters long, but on Linux, UNIX, and z/OS systems, they are restricted to eight characters.
- Database products might also restrict user IDs to eight characters; for example DB2[®] has this restriction. If you have a mixed environment, ensure that the user IDs that you use are limited to a maximum of eight characters.
- Ensure that the case (upper, lower, or mixed) of user IDs is consistent. In some environments, uppercase and lowercase user IDs are considered the same, but in other environments, user IDs of different case are considered unique.

For example, on Windows systems, the user IDs 'tester' and 'TESTER' are identical; on Linux and UNIX systems, they are recognized as different user IDs.

- Check the validity of spaces and special characters in user IDs to ensure that, if used, these characters are accepted by all relevant systems and products that you install.

Consider the following roles:

Procedure

- Administrator user IDs that can issue **mqs**i* commands. Ensure that all these user IDs are members of the mqbrkrs group.

If you enable broker administration security, you must set up additional authorization for user IDs for the commands listed in Commands and authorizations for broker administration security. For more information about broker administration security, see “Administration security overview” on page 214.

- On Windows only, service user IDs under which brokers run. For further information, see “Deciding which user account to use for the broker service ID” on page 407.
- IBM Integration Toolkit users. See “Security for the IBM Integration Toolkit and IBM Integration Explorer” on page 398 for information about checking and securing connections from the IBM Integration Toolkit.

Related concepts:

“Security overview” on page 203

When you are designing an IBM Integration Bus application it is important to consider the security measures that are needed to protect the information in the system.

“Administration security overview” on page 214

Broker administration security controls the rights of users to complete administrative tasks for a broker and its resources.

Related tasks:

“Considering security for a broker” on page 407

Consider several factors when you are deciding which users can execute broker commands, and which users can control security for other broker resources.

“Configuring broker user IDs on z/OS” on page 409

When you create your brokers on z/OS, you must set up security by configuring broker user IDs with the appropriate permissions.

Related reference:



Security requirements for Windows systems

Security requirements depend on the administrative task that you want to perform.



Security requirements for Linux and UNIX platforms

View a summary of the authorizations in a Linux or UNIX environment.



Installation Guide

Installation information for IBM Integration Bus is provided in the online information center.

Security for the IBM Integration Toolkit and IBM Integration Explorer

What to consider when you set up security for the IBM Integration Toolkit and IBM Integration Explorer.

When you create a broker, a default SVRCONN channel, `SYSTEM.BKR.CONFIG`, is created. This channel supports connections from one or more remote clients to the broker. Clients that are running on the same computer as the broker connect directly to the queue manager; they do not require a connection through a channel.

If you enable security, then, by default, the user ID that is used to run the IBM Integration Toolkit and the IBM Integration Explorer is passed to the broker (in the WebSphere MQ header) and used for authorization. For information about authorizations for administration applications, see [Tasks and authorizations for administration security](#).

You can override the default behavior by specifying a user ID to be used for authorization in the `MCAUSER` attribute on the SVRCONN channel. If you set the `MCAUSER` attribute, then any incoming request from the IBM Integration Toolkit or the IBM Integration Explorer is processed by the broker as if it was made by the user that is specified in the `MCAUSER` attribute, instead of the user that is running the IBM Integration Toolkit or the IBM Integration Explorer.

Ensure that user IDs are not more than eight characters long.

If you want to secure the connection between your IBM Integration Toolkit session, or IBM Integration Explorer session, and the broker, you can configure the SVRCONN channel to specify security options.

You can use a single channel for all client connections, create a channel for each client connection, or share connections between two or more clients that have the same security requirements. You can use the default channel, and create additional channels if required. If you do not use the default channel, you must set the alternative name in the connection properties.

You can secure the connection between the IBM Integration Explorer, the IBM Integration Toolkit, a command that uses the CMP interface (`mqsichangeresourcestats`, `mqsicreateexecutiongroup`, `mqsidedeleteexecutiongroup`,

`mqsdeploy`, `mqsilist`, `mqsimode`, `mqsireloadsecurity`, `mqsireportresourcestats`, `mqsistartmsgflow`, `mqsistopmsgflow`), or a CMP application, by using one or both of the following options:

- Create and enable a pair of WebSphere MQ channel security exits to run at the client and broker ends of the SVRCONN channel that connects the two components. For information about defining security exits, see “Security exits” on page 205.
- Implement Secure Socket Layer (SSL) security on the SVRCONN channel. For information about enabling SSL, see “Enabling SSL on the WebSphere MQ Java Client” on page 400.

Using security exits

Define a security exit on the WebSphere MQ channel when you create an integration node (broker) connection.

About this task

To create a security exit on the WebSphere MQ channel that you define for communications between the IBM Integration Toolkit, or IBM Integration Explorer and the integration node, you must define a security exit when you create the connection.

Procedure

1. In the IBM Integration Toolkit or the IBM Integration Explorer, right-click the Integration Nodes folder and click **Connect to a Remote Integration Node**. The Connect to a Remote Integration Node wizard opens.
2. Enter the values for **Queue Manager Name**, **Host**, and **Port** that you want to use.
3. Enter the Security Exit **Class** name. The name must be a valid Java class name.
4. Enter the **JAR File location** for the Security Exit that is required on this connection. In IBM Integration Explorer you can click **Browse** to select the file.

Results

The security exit is started every time a message passes across the connection.

What to do next

Alternatively, use SSL to communicate between the IBM Integration API (also known as the CMP) and the broker; see “Implementing SSL authentication” on page 415.

Related concepts:



IBM Integration Explorer

The IBM Integration Explorer is a graphical user interface based on the Eclipse platform for administering your integration nodes (brokers).

“Security overview” on page 203

When you are designing an IBM Integration Bus application it is important to consider the security measures that are needed to protect the information in the system.

“Security exits” on page 205

Use security exit programs to verify that the partner at the other end of a connection is genuine.

Related tasks:

“Integration Bus server security” on page 396

You must consider several security aspects when you are setting up brokers running on Windows, Linux, z/OS, or UNIX platforms.

“Implementing SSL authentication” on page 415

Use SSL authentication to enhance security in your broker environment.

Related reference:



Security requirements for administrative tasks

You can configure access permissions to govern which users and groups can manipulate objects in the broker network. Security requirements for administrative tasks depend on the platform that you use.

Enabling SSL on the WebSphere MQ Java Client

The WebSphere MQ Java Client supports SSL-encrypted connections over the server-connection (SVRCONN) channel between an application and the queue manager. Configure SSL support for connections between applications that use the CMP (including the IBM Integration Toolkit and the IBM Integration Explorer) and a broker.

About this task

Use the following instructions to implement SSL security on the SVRCONN channel. You must have the appropriate software to manage SSL certificate stores; for example, you can install either the WebSphere MQ Client or the Server, and use the IBM Key Management tools for the client. You can use either JKS or PKCS12 stores.

Procedure

1. Use WebSphere MQ facilities to update the SVRCONN definition to specify the required value in the SSLCIPH attribute.
2. In the IBM Integration Toolkit or IBM Integration Explorer, define the connection to the broker. You can set the SSL fields only when you define the connection; you cannot change them later. If you have already defined your connection, delete it, and define it again.
3. Select the cipher suite that matches the value you set for the SSLCIPH property of the SVRCONN channel.
4. Enter the full path and name for the keystore and truststore, or click **Browse** to search for them.
5. Add the queue manager certificate to the client truststore.
6. For one-way authentication, when the client CMP application authenticates the broker, complete the following steps:
 - a. Generate or obtain all the appropriate keys and certificates. You must include a signed pkcs12 certificate for the server and the appropriate public key for the certificate authority that signed the pkcs12 certificate. See “Creating SSL certificates for the WebSphere MQ Java Client” on page 402, for some example steps for creating keys and certificates.
 - b. Add the pkcs12 certificate to the queue manager certificate store and assign it to the queue manager. Use the standard WebSphere MQ facilities; for example, WebSphere MQ Explorer.
 - c. Add the certificate of the certificate authority to the Java Secure Socket Extension (JSSE) truststore of the Java Virtual Machine (JVM) at the CMP application end by using a tool such as Keytool.

- d. Decide which cipher suite to use and change the properties on the server-connection channel by using WebSphere MQ Explorer to specify the cipher suite to be used. This channel has a default name of `SYSTEM.BKR.CONFIG`; this name is used unless you specified a different name on the Connect to Remote Integration Node wizard; see “Connecting to a remote broker” on page 4 and “Connecting to a remote integration node on z/OS” on page 6.
- e. Add the required parameters (cipher suite, for example) to the CMP application. If a truststore other than the default is used, its full path must be passed in by the truststore parameter.

After you complete these steps, the CMP application connects to the broker if it has a valid key that is signed by a trusted certificate authority.

7. For two-way authentication, when the broker also authenticates the CMP application, complete the following additional steps:
 - a. Generate or obtain all the appropriate keys and certificates. You must include a signed pkcs12 certificate for the client and the appropriate public key for the certificate authority that signed the pkcs12 certificate. See “Creating SSL certificates for the WebSphere MQ Java Client” on page 402, for some example steps for creating keys and certificates.
 - b. Add the certificate of the certificate authority to the queue manager certificate store by using the standard WebSphere MQ facilities.
 - c. Set the server-connection channel to always authenticate. Specify `SSLCAUTH(REQUIRED)` in `runmqsc`, or in WebSphere MQ Explorer.
 - d. Add the pkcs12 certificate to the JSSSE keystore of the JVM at the CMP application end by using a tool such as Keytool.
 - e. If you are not using the default keystore, its full path must be passed into the CMP through the keystore parameter.

After you complete these steps, the broker allows the CMP application to connect only if that application has a certificate signed by one of the certificate authorities in its keystore.

What to do next

You can make further restrictions by using the `sslPeerName` field; for example, you can allow connections only from certificate holders with a specific company or department name in their certificates. In addition, you can invoke a security exit for communications between the CMP applications and the broker; see “Using security exits” on page 399.

For more information about configuring connections to be secured with SSL, see the WebSphere MQ Java Client developerWorks® article.

Related concepts:

“Security overview” on page 203

When you are designing an IBM Integration Bus application it is important to consider the security measures that are needed to protect the information in the system.

“Security for the IBM Integration Toolkit and IBM Integration Explorer” on page 398

What to consider when you set up security for the IBM Integration Toolkit and IBM Integration Explorer.

Related tasks:

“Using security exits” on page 399

Define a security exit on the WebSphere MQ channel when you create an integration node (broker) connection.

“Implementing SSL authentication” on page 415

Use SSL authentication to enhance security in your broker environment.

Related reference:



Security requirements for administrative tasks

You can configure access permissions to govern which users and groups can manipulate objects in the broker network. Security requirements for administrative tasks depend on the platform that you use.



mqsicreatebroker command

Use the **mqsicreatebroker** command to create a broker and its associated resources.



mqsichangebroker command

Use the **mqsichangebroker** command to change one or more of the configuration parameters of the broker.

Related information:



WebSphere MQ Version 7 product documentation

Creating SSL certificates for the WebSphere MQ Java Client:

The WebSphere MQ Java Client supports SSL-encrypted connections over the server-connection (SVRCONN) channel between an application and the queue manager. To configure SSL-encrypted connections you must first create key stores and certificates.

Before you begin

Before you start:

- Create a broker
- Start the broker
- Set the environment variable `JAVA_HOME` to the location of the IBM Key Management tools in the WebSphere MQ install, for example `C:\Program Files\IBM\WebSphere MQ\gskit\jre\` or `/opt/mqm/ssl/jre`.

About this task

Each WebSphere MQ queue manager has a key repository for certificates. When an application attempts to connect to a secure queue manager, the application's certificate must be validated against the contents of the queue manager's key repository. One option for configuring SSL for the queue manager is to use a self-signed certificate.

Two certificates must be signed and created. One must be created for the server queue manager, and a second created for the client, for example the IBM Integration Explorer.

The instructions in this topic use the **gsk7cmd** command to create and sign the certificates. Run the **gsk7cmd** for a full list of the parameters you can use on the command. To run the **gsk7cmd** command:

- On Windows, enter the following command on a command line:

C:\Program Files\IBM\gsk7\bin\gsk7cmd

- On Linux, enter the following command on a command line:
/opt/mqm/ssl/jre/bin/gsk7cmd

Refer to the WebSphere MQ security documentation for more information about SSL, and creating certificates.

Creating a server certificate for the queue manager:

About this task

Use the IBM Key Management tools on the command line to create a certificate for the queue manager. In the following example you must replace the following parameters with your own values:

password

A password for the certificate repository.

qmname

The name of the queue manager for which you want to create a certificate in lower case.

QMNAME

The name of the queue manager for which you want to create a certificate in upper case.

Procedure

1. Run the following command to create a key repository of type cms:

```
gsk7cmd
-keydb-create
-dbkey.kdb
-pwPASSWORD
-typecms -stash
```

The key.crl, key.kdb, key.rdb, and key.sth key files are created.

2. Run the following command to create a self-signed certificate, where the -dn flag contains details of your organization:

```
gsk7cmd
-cert-create
-dbkey.kdb
-pwPASSWORD
-label"qmname"
-dn"CN=My Queue Manager,O=My Company,C=UK"
-expire1000
```

3. Run the following command to create a request for a personal certificate:

```
gsk7cmd
-certreq-create
-dbkey.kdb
-pwPASSWORD
-label"ibmwebspheremqmname"
-dn"CN=My Queue Manager,O=My Company,C=UK"
-fileQMNAME_request.arm
```

4. Sign the certificate using a certificate authority.

- To obtain a certificate from a certificate authority, you must send the file containing a certificate signing request to your chosen certificate authority.
- Alternatively you can use the IBM Key Management tools on the command line to sign the certificate.

- ```

gsk7cmd
 -cert-sign
 -dbkey.kdb
 -pwPASSWORD
 -label"qmname"
 -fileQMNAME_request.arm
 -targetQMNAME_signed.arm
 -expire364

```
5. Run the following command to add the signed certificate to the repository:

```

gsk7cmd
 -cert-receive
 -dbkey.kdb
 -pwPASSWORD
 -fileQMNAME_signed.arm

```
  6. Run the following command to export the signed client userid certificate in a transferable format (in this case PKCS12), with the associated private key and public CA certificate:

```

gsk7cmd
 -cert-export
 -dbkey.kdb
 -pwPASSWORD
 -label"ibmwebspheremqqmname"
 -targetQMNAME_personal.p12
 -target_pwPASSWORD
 -target_typepkcs12

```
  7. Delete the certificate from the repository:

```

gsk7cmd
 -cert-delete
 -dbkey.kdb
 -pwPASSWORD
 -label"ibmwebspheremqqmname"

```
  8. Create a subdirectory called *QMNAME\_CMS*, and navigate to this directory on the command line.
  9. Run the following command to create a certificate repository in the *QMNAME\_CMS* directory:

```

gsk7cmd
 -keydb-create
 -dbkey.kdb
 -pwPASSWORD
 -typecms -stash

```
  10. Run the following command to import the PKCS12 file into the repository:

```

gsk7cmd
 -cert-import
 -file"QMNAME_personal.p12"
 -pwPASSWORD
 -typepkcs12
 -targetkey.kdb
 -target_pwPASSWORD
 -target_typecms

```
  11. Return to the original directory in which you created the key repository in step 1.

## What to do next

Follow the steps in the next section to create a client certificate for the IBM Integration Explorer.

*Creating a client certificate for the IBM Integration Explorer:*

### About this task

Use the IBM Key Management tools on the command line to create a certificate for the IBM Integration Explorer. In the following example you must replace the following parameters with your own values:

*password*

A password for the certificate repository.

*qmname*

The name of the queue manager for which you want to create a certificate in lower case. This is the same value used in the steps to create a client certificate for the queue manager.

*USERID*

The user id for which you want to create a certificate.

### Procedure

1. In the directory where you created a key repository for the server queue manager in step 1 above, run the following command to create a request (private key plus certificate details) for a certificate to be signed for the server queue manager:

```
gsk7cmd
-certreq-create
-dbkey.kdb
-pwPASSWORD
-label"ibmwebspheremqmname"
-dn"CN=userid@mycompany.com,O=My Company,C=UK"
-fileUSERID_request.arm
```

2. Run the following command to sign the certificate:

```
gsk7cmd
-cert-sign
-dbkey.kdb
-pwPASSWORD
-label"qmname"
-fileUSERID_request.arm
-targetUSERID_signed.arm
-expire364
```

3. Run the following command to add the signed certificate to the repository:

```
gsk7cmd
-cert-receive
-dbkey.kdb
-pwPASSWORD
-fileUSERID_signed.arm
```

4. Run the following command to export the signed client userid certificate in a transferable format (in this case pkcs12), with the associated private key and public CA certificate:

```
gsk7cmd
-cert-export
-dbkey.kdb
-pwPASSWORD
```



- ```

-label"ibmwebspheremqmnname"
-targetUSERID_personal.p12
-target_pwPASSWORD
-target_typepkcs12

```
5. Delete the certificate from the repository:

```

gsk7cmd
-cert-delete
-dbkey.kdb
-pwPASSWORD
-label"ibmwebspheremqmnname"

```
 6. Create a subdirectory called *USERID_JKS*, and navigate to this directory on the command line.
 7. Run the following command to create a certificate repository in the *USERID_JKS* directory:

```

gsk7cmd
-keydb-create
-dbkeyStore.jks
-pwPASSWORD
-typejks

```
 8. Run the following command to import the pkcs12 file into the repository:

```

gsk7cmd
-cert-import
-file"USERID_personal.p12"
-pwPASSWORD
-typepkcs12
-targetkeyStore.jks
-target_pwPASSWORD
-target_typejks"

```
 9. Return to the original directory in which you created the key repository in step 1.

What to do next

You must now copy the files from the *Label_CMS* directory to your queue manager's SSL directory. For example, */var/mqm/qmgrs/QM1/ssl* or *C:\Program Files\IBM\WebSphere MQ\Qmgrs\QM1\ssl*. The *keystore.jks* file in the *LABEL_JKS* directory must be on the same machine as the IBM Integration Explorer. You might also require the *AMQCLCHL.TAB* file to be copied to the same system as the IBM Integration Explorer. This file can be found in the queue manager's *@ipcc* directory, for example, */var/mqm/qmgrs/QM1/@ipcc* or *C:\Program Files\IBM\WebSphere MQ\qmgrs\QM1\@ipcc*.

When you configure the SSL settings in the IBM Integration Explorer you must specify the full path to the *keystore.jks* file.

Related concepts:

"Security overview" on page 203

When you are designing an IBM Integration Bus application it is important to consider the security measures that are needed to protect the information in the system.

Related tasks:

"Using security exits" on page 399

Define a security exit on the WebSphere MQ channel when you create an integration node (broker) connection.

“Implementing SSL authentication” on page 415

Use SSL authentication to enhance security in your broker environment.

“Enabling SSL on the WebSphere MQ Java Client” on page 400

The WebSphere MQ Java Client supports SSL-encrypted connections over the server-connection (SVRCONN) channel between an application and the queue manager. Configure SSL support for connections between applications that use the CMP (including the IBM Integration Toolkit and the IBM Integration Explorer) and a broker.

Related reference:



Security requirements for administrative tasks

You can configure access permissions to govern which users and groups can manipulate objects in the broker network. Security requirements for administrative tasks depend on the platform that you use.

Related information:



WebSphere MQ Version 7 product documentation

Considering security for a broker

Consider several factors when you are deciding which users can execute broker commands, and which users can control security for other broker resources.

About this task

Although most security for the broker and broker resources is optional, you might find it appropriate to restrict the tasks that some user IDs can perform. You can then apply greater control to monitor changes.

You can control all broker administration tasks by enabling broker administration security when you create a broker. You can also change existing brokers to enable administration security. This option is described in “Setting up administration security” on page 221, and is independent of the options described in this section.

When you are deciding which users are to perform the different tasks, consider the following steps:

Procedure

1. “Deciding which user account to use for the broker service ID”
2. “Setting security on the broker queues” on page 408
3. “Securing the broker registry” on page 409

Deciding which user account to use for the broker service ID

About this task

On a Linux or UNIX operating system, when you run the **mqsistart** command with a user ID that is a member of the **mqm** and **mqbrkrs** groups, the user ID under which you run the **mqsistart** command becomes the user ID under which the broker component process runs.

On the Windows platform the broker runs under a service user account. To decide which user ID to use for the broker service ID answer the following questions:

Procedure

1. Do you want your broker to run under a Windows local account?
 - a. No: Go to the next question.
 - b. Yes: Ensure that your user ID has the following characteristics:
 - It is defined in your local domain.
 - It is a member of the mqbrkrs group.Go to “Setting security on the broker queues.”
2. Do you want your broker to run under a Windows domain account?
 - a. No: Go to the next question.
 - b. Yes: Assume that your computer named, for example, WKSTN1, is a member of a domain named DOMAIN1. When you run a broker using, for example, DOMAIN1\user1, ensure that:
 - Your user ID has been granted the Logon as a service privilege (from the Local Security Policy).
 - DOMAIN1\user1 is a member of DOMAIN1\MyDomainGroup group, where MyDomainGroup is a domain group which you have defined on your domain controller.
 - DOMAIN1\MyDomainGroup is a member of WKSTN1\mqbrkrs.Go to “Setting security on the broker queues.”
3. Do you want your broker to run under the Windows built in LocalSystem account?
 - a. Yes: Specify LocalSystem for the *-i* parameter on the **mqsicreatebroker** or **mqsichangebroker** command.

In either case you must enter the *-a* (password) parameter on the command line, but the value entered is ignored.

Go to “Setting security on the broker queues.”

Results

Note that for cases one and two above, the user ID chosen must be granted the Logon as a service privilege.

This is normally done automatically by the **mqsichangebroker** command or the **mqsichangeproperties** command when a service user ID is specified that does not have this privilege.

However, if you want to do this manually before running these commands, you can do this by using the Local Security Policy tool in Windows, which you can access by selecting **Control Panel > Performance and maintenance > Administrative Tools > Local Security Policy**.

Setting security on the broker queues

About this task

When you run the **mqsicreatebroker** command, the local mqbrkrs group is granted access to internal queues whose names begin with the characters SYSTEM.BROKER.

Securing the broker registry

About this task

Broker operation depends on the information in the broker registry, which you must secure to guard against accidental corruption. The broker registry is stored on the file system. Set your operating system security options so that only user IDs that are members of the group `mqbrkrs` can read from or write to `brokername/CurrentVersion` and all subkeys.

Related concepts:

“Security overview” on page 203

When you are designing an IBM Integration Bus application it is important to consider the security measures that are needed to protect the information in the system.

“Authorization for configuration tasks” on page 205

Authorization is the process of granting or denying access to a system resource.

Related reference:



Security requirements for Windows systems

Security requirements depend on the administrative task that you want to perform.



`mqsi createbroker` command

Use the `mqsi createbroker` command to create a broker and its associated resources.



`mqsi changebroker` command

Use the `mqsi changebroker` command to change one or more of the configuration parameters of the broker.

Configuring broker user IDs on z/OS

When you create your brokers on z/OS, you must set up security by configuring broker user IDs with the appropriate permissions.

Before you begin

Before you start

Before starting this task, read “Integration Bus server security” on page 396 and Creating a broker on z/OS.

About this task

The following steps guide you through configuring a broker user ID on z/OS:

Procedure

1. Decide on the started task name of the broker. This name is used to set up started task authorizations, and to manage your system performance.
2. Decide on a dataset naming convention for your IBM Integration Bus PDSE. A typical name might be `WMQ1.MQP1BRK.CNTL`, where `MQP1` is the queue manager name. You must give the IBM Integration Bus, WebSphere MQ, and z/OS administrators access to these data sets. You can control access in several ways, for example:
 - Give each user individual access to the specific data set

- Define a generic dataset profile, defining a group that contains the user IDs of the administrators. Grant the group control access to the generic data set profile
3. Configure access to components and resources on z/OS. For more information, see Summary of required access (z/OS).
 4. Define an OMVS group segment for this group so that information can be extracted from the External Security Manager (ESM) database to enable you to use Publish/Subscribe security.
 5. Define an OMVS segment for the started task user ID and give its home directory sufficient space for any IBM Integration Bus memory dumps. Consider using the started task procedure name as the started task user ID.
 6. Check that your OMVS segment is defined by using the following TSO command:

```
LU userid OMVS
```

The command output includes the OMVS segment, for example:

```
USER=MQP1BRK NAME=SMITH, JANE OWNER=TSOUSER
CREATED=99.342 DEFAULT-GROUP=TSOUSER PASSDATE=01.198
PASS-INTERVAL=30
.....
OMVS INFORMATION
-----
UID=0000070594
HOME=/u/MQP1BRK
PROGRAM=/bin/sh
CPUTIMEMAX=NONE
ASSIZEMAX=NONE
FILEPROCMAx=NONE
PROCUSERMAX=NONE
THREADSMAX=NONE
MMAPAREAMAX=NONE
```

The command:

```
df -P /u/MQP1BRK
```

displays the amount of space used and available, where /u/MQP1BRK is the value from HOME (on a previous line). This command shows you how much space is currently available in the file system. Check with your data administrator that this space is sufficient. You require a minimum of 400 000 blocks available if a memory dump is taken.

7. Associate the started task procedure with the user ID to be used. For example, you can use the STARTED class in RACF. The IBM Integration Bus and z/OS administrators must agree on the name of the started task.
8. IBM Integration Bus administrators require an OMVS segment and a home directory. Check the setup previously described.
9. The started task user IDs and the IBM Integration Bus administrators require access to the install processing files, the component-specific files, and the home directory of the started task. During customization, the file ownership can be changed to alter group access. This change might require superuser authority.

Results

On z/OS, you can specify an alternative user ID to run an integration server so that it accesses resources according to the permissions assigned to it, rather than

the permissions assigned to the main broker user ID; for more information, see “Integration server user IDs on z/OS.” For information about how to specify an integration server user ID, see “Specifying an alternative user ID to run an integration server on z/OS” on page 413.

What to do next

When the service user ID is *root*, all libraries loaded by the broker, including all user-written plug-in libraries, and all shared libraries that they might access, also have root access to all system resources (for example, file sets). Review and assess the risk involved in granting this level of authorization.

Related concepts:



The IBM Integration Bus environment

An integration node (broker) is a set of execution processes that hosts one or more message flows to route, transform, and enrich in flight messages.

“Security overview” on page 203

When you are designing an IBM Integration Bus application it is important to consider the security measures that are needed to protect the information in the system.

“Integration server user IDs on z/OS”

On z/OS, you can specify an alternative user ID to run an integration server so that it accesses resources according to the permissions assigned to it, rather than the permissions assigned to the main broker user ID.

Related tasks:



Creating a broker on z/OS

Create the broker component and the other resources on which it depends.



Customizing the broker JCL

This subtask is part of the larger task of creating a broker on z/OS.



Creating the environment file

This is part of the larger task of creating a broker on z/OS.

“Specifying an alternative user ID to run an integration server on z/OS” on page 413

You can change the user ID under which an integration server runs so that it can access resources according to the permissions assigned to it, rather than the permissions assigned to the main broker user ID.

Integration server user IDs on z/OS

On z/OS, you can specify an alternative user ID to run an integration server so that it accesses resources according to the permissions assigned to it, rather than the permissions assigned to the main broker user ID.

You can specify an alternative user ID to run an integration server, which means that you can run one or more message flows under a different user ID from the main broker ID. When external resources are accessed by a message flow, access is granted according to the permissions assigned to the user ID that is running the integration server. By having different user IDs for different integration servers, you can control the access to resources at the level of the integration server rather than at the level of the main broker user ID. The user IDs for the integration servers must be in the same primary group as the main broker user ID, so that shared resources can be read and updated.

On z/OS, the user ID assigned to the broker is the started task (STC) user ID that is assigned to the started task JCL. By default, each broker on z/OS has a single started task JCL, which is used to start the main broker address space and all associated integration server address spaces. However, you can specify a different started task JCL, and therefore a different user ID, for one or more integration servers. As a result, integration servers can be started using a different started task JCL and run under different user IDs with different permissions to access resources. For example, an integration server can access messages from WebSphere MQ through the integration server's task ID (rather than the main broker ID) by default. Integration servers can also access files according to the permissions that are assigned to the integration server's user ID.

You can specify the required environment variable in the main broker profile, BIPBPROF. You can use the `MQSI_STARTEDTASK_FIXED_integrationServerName`, `MQSI_STARTEDTASK_MULTI_integrationServerName`, or `MQSI_STARTEDTASK_DEFAULT` environment variables to specify a different started task and user ID, for one or more integration servers (where *integrationServerName* is the name of your integration server). These environment variables override the started task and user ID that are associated with the broker, and replace them with the started task and user ID associated with a specific integration server:

- Use the `MQSI_STARTEDTASK_FIXED_integrationServerName=STC` environment variable to specify the name of one or more integration servers (where *integrationServerName* is the last 8 characters of the integration server name, and *STC* is the name of the integration server started task JCL). For example, specify `1DEFAULT` in place of *integrationServerName* to override an integration server called `TEST1DEFAULT`. If multiple integration servers end with the same 8 characters, all will be overridden; for example, `TEST11DEFAULT` would be overridden, but `TEST12DEFAULT` would not.
- Use the `MQSI_STARTEDTASK_MULTI_integrationServerName=STC` environment variable to override the user ID for multiple integration servers (where *integrationServerName* functions as a wildcard and *STC* is the name of the started task JCL that is used to start each of the integration servers). For example, specify `MQ05` in place of *integrationServerName* to override the user ID for any integration servers in which the last 8 characters start with `MQ05`.
- Use the `MQSI_STARTEDTASK_DEFAULT=STC` environment variable to override the started task JCL (*STC*) for all integration servers, unless it is overridden by the `MQSI_STARTEDTASK_FIXED_integrationServerName` or `MQSI_STARTEDTASK_MULTI_integrationServerName` environment variable.

For information about how to define a user ID on an integration server, see “Specifying an alternative user ID to run an integration server on z/OS” on page 413.

Related concepts:



The IBM Integration Bus environment

An integration node (broker) is a set of execution processes that hosts one or more message flows to route, transform, and enrich in flight messages.


Related tasks:


“Specifying an alternative user ID to run an integration server on z/OS” on page 413


You can change the user ID under which an integration server runs so that it can access resources according to the permissions assigned to it, rather than the permissions assigned to the main broker user ID.


“Configuring broker user IDs on z/OS” on page 409

When you create your brokers on z/OS, you must set up security by configuring broker user IDs with the appropriate permissions.

 Creating a broker on z/OS
Create the broker component and the other resources on which it depends.

 Customizing the broker JCL
This subtask is part of the larger task of creating a broker on z/OS.

 Creating the environment file
This is part of the larger task of creating a broker on z/OS.

 Copying the broker started task to the procedures library
This is part of the larger task of creating a broker on z/OS.

Specifying an alternative user ID to run an integration server on z/OS

You can change the user ID under which an integration server runs so that it can access resources according to the permissions assigned to it, rather than the permissions assigned to the main broker user ID.

Before you begin

Before you start

Before starting this task, read the following topics:

- Creating a broker on z/OS
- “Configuring broker user IDs on z/OS” on page 409
- “Integration server user IDs on z/OS” on page 411

About this task

Complete the following steps to specify an alternative user ID for the integration server, to be used instead of the broker's user ID:

Procedure

1. Create the new RACF started task profile with a new user ID, which will be used to run the integration server. Consider the following points when you are creating the new started task:
 - The new started task must be created with an OMVS segment including a unique UID, home directory, and the ability to create data sets under the broker's HLQ and alias.
 - The started task procedure name to be used for the integration server address space must start with the same four characters as the main broker started task. For example, if the main broker started task is MQ01BRK, the started task name for the integration server could be MQ01EG1 but not MQ02EG2. As a result, consistency is maintained between the main broker started task, the integration server, and the queue manager, which helps to identify the relationship between them. If the first four characters are not the same, the integration server is started using the main broker started task JCL.
2. Ensure that the new user ID associated with the new started task JCL has the same RACF primary group as the existing broker user ID, so that they can access shared resources. Also ensure that the new user ID has the required privileges to the existing broker filesystem and dataset (which it should have through the primary group access).

3. Ensure that the MQ and SMF authorizations are updated for the new user ID; for more information, see Summary of required access (z/OS).
4. Copy the existing broker started task JCL to the new started task JCL in the PROCLIB.
5. Ensure that the main broker user ID has been granted permission to the SUPERUSER.PROCESS.KILL RACF profile. This permission is required so that the main control address space can recover any existing integration server address spaces in the event of a failure.
6. Refresh the started RACF classes to implement the updates.
7. Change the user ID by adding the appropriate environment variable to the broker's profile.
 - The integration server name specified in the environment variable is the last 8 characters of the integration server, after any overrides have been applied. This is the same 8-character name that is displayed as the STEPNAME against the integration server address space in SDSF.
 - Ensure that the integration server name contains only characters that are valid in the environment variable. If invalid characters are used, the user ID cannot be overridden.
 - If you specify more than one environment variable, they are read in the following order (with MQSI_STARTEDTASK_FIXED_ *integrationServerName* taking precedence):
 - a. MQSI_STARTEDTASK_FIXED_ *integrationServerName*
 - b. MQSI_STARTEDTASK_MULTI_ *integrationServerName*
 - c. MQSI_STARTEDTASK_DEFAULT
 where *integrationServerName* is the name of your integration server. For example:
 - export MQSI_STARTEDTASK_FIXED_DEFAULT=MQ01EG1 changes any integration server which has the last 8 characters equal to DEFAULT to started task MQ01EG1
 - export MQSI_STARTEDTASK_MULTI_TEST=MQ01EG2 changes any integration server which has the last 8 characters starting with TEST to started task MQ01EG2
 - export MQSI_STARTEDTASK_DEFAULT=MQ01EG3 changes all integration servers which are not overridden by MQSI_STARTEDTASK_FIXED_ *integrationServerName* or MQSI_STARTEDTASK_MULTI_ *integrationServerName* to started task MQ01EG3.
8. Submit BIPGEN to the broker's ENVFILE.
9. Restart the broker.

Related concepts:

“Integration server user IDs on z/OS” on page 411

On z/OS, you can specify an alternative user ID to run an integration server so that it accesses resources according to the permissions assigned to it, rather than the permissions assigned to the main broker user ID.




The IBM Integration Bus environment


An integration node (broker) is a set of execution processes that hosts one or more message flows to route, transform, and enrich in flight messages.


Related tasks:


“Configuring broker user IDs on z/OS” on page 409

When you create your brokers on z/OS, you must set up security by configuring broker user IDs with the appropriate permissions.

 Creating a broker on z/OS
Create the broker component and the other resources on which it depends.

 Customizing the broker JCL
This subtask is part of the larger task of creating a broker on z/OS.

 Creating the environment file
This is part of the larger task of creating a broker on z/OS.

 Copying the broker started task to the procedures library
This is part of the larger task of creating a broker on z/OS.

Implementing SSL authentication

Use SSL authentication to enhance security in your broker environment.

About this task

The following topics contain instructions for implementing SSL authentication:


- “Setting up a public key infrastructure”
- “Authorization by using SSL Client Certificates” on page 424
- “Listing SSL cipher suites” on page 428
- “Implementing SSL authentication on z/OS” on page 429
- “Securing message flows by using SSL configuration” on page 364
- “Securing integration services by using SSL configuration” on page 387
- Configuring IBM Integration Bus and Business Process Manager to work together with SSL

Related concepts:

“Security overview” on page 203

When you are designing an IBM Integration Bus application it is important to consider the security measures that are needed to protect the information in the system.

Related reference:

 Security requirements for administrative tasks
You can configure access permissions to govern which users and groups can manipulate objects in the broker network. Security requirements for administrative tasks depend on the platform that you use.

Setting up a public key infrastructure

Configure keystores, truststores, passwords, and certificates to enable SSL communication, and Web Services Security.

Before you begin

Decide how you will use the public key infrastructure (PKI). You can configure keystores and truststores at either broker level (one keystore, one truststore, and one personal certificate for each broker) or at integration server level (one keystore, one truststore, and one personal certificate for each integration server). Integration servers that do not have PKI configured use the broker-level PKI configuration.

Encryption strength

The IBM Integration Bus Java runtime environment (JRE) is provided with strong but limited strength encryption. If you cannot import keys into keystores, limited strength encryption might be the cause. Either start ikeyman by using the **strmqikm** command, or download unrestricted jurisdiction policy files from IBM developer kits: Security information.

Important: Your country of origin might have restrictions on the import, possession, use, or re-export to another country, of encryption software. Before you download or use the unrestricted policy files, you must check the laws of your country. Check its regulations and its policies on the import, possession, use, and re-export of encryption software, to determine whether it is permitted. Note that when you apply a fix pack to an existing IBM Integration Bus installation, the JVM is overwritten, including any updated policy set files. These policy set files must be restored before you restart the broker.

About this task

This topic uses the command line tool, **gsk7cmd**, to create and populate keystores and truststores. The **gsk7cmd** tool is a part of the Global Secure Toolkit, supplied with WebSphere MQ. Other options, which are supplied with the IBM Integration Bus JVM, include the following:

- A command-line tool, **keytool**.
- A graphical tool, **iKeyman**.

To create the infrastructure, complete the following tasks:

1. "Creating a keystore file or a truststore"
2. "Creating a self-signed certificate for test use" on page 417
3. "Importing a certificate for production use" on page 417
4. "Viewing details of a certificate" on page 417
5. "Extracting a certificate" on page 418
6. "Adding a signer certificate to the truststore" on page 419
7. "Listing all certificates in a keystore" on page 420
8. "Configuring PKI at broker level" on page 421
9. "Configuring PKI for the broker-wide HTTP listener" on page 421
10. "Configuring PKI for an embedded HTTP listener in an integration server" on page 422

Creating a keystore file or a truststore:

The keystore file contains the personal certificate for the broker or for the integration server. You can have only one personal certificate in the keystore. You can store signer certificates in the same file, or create a separate file, which is known as a truststore.

Procedure

1. Set the JAVA_HOME environment variable. For example,

```
set JAVA_HOME=install_dir\MQSI\9.0\jre17
```
2. Issue the following command:

```
gsk7cmd -keydb -create  
-db keystore_name  
[-pw password]  
-type jks
```

The *password* argument is optional. If you omit it, you are prompted to enter a password. For example:

```
gsk7cmd -keydb -create
        -db myBrokerKeystore.jks
        -type jks
```

A password is required to access this key database.
Please enter a password:

Creating a self-signed certificate for test use:

Use self-signed certificates only for testing SSL, not in production.

Procedure

Enter the following command:

```
gsk7cmd -cert -create
        -db keystore_name
        [-pw password]
        -label cert_label
        -dn "distinguished_name"
```

For example:

```
gsk7cmd -cert -create
        -db -myBrokerKeystore.jks
        -label MyCert
        -dn "CN=MyBroker.Server,O=IBM,OU=ISSW,L=Hursley,C=GB"
```

A password is required to access this key database.
Please enter a password:

Importing a certificate for production use:

Import a personal certificate from a certificate authority for production use.

Procedure

Issue the following command:

```
gsk7cmd -cert -import
        -db pkcs12_file_name
        [-pw pkcs12_password]
        -label label
        -type pkcs12
        -target keystore_name
        [-target_pw keystore_password]
```

For example:

```
gsk7cmd -cert -import
        -db SOAPListenerCertificate.p12
        -label soaplistener
        -type pkcs12
        -target myBrokerKeystore.jks
        -target_pw myBrokerKpass
```

A password is required to access this key database.
Please enter a password:

Viewing details of a certificate:

Procedure

Issue the following command:

```
gsk7cmd -cert -details
  -db keystore_name
  [-pw password]
  -label label
```

For example:

```
gsk7cmd -cert -details
  -db myKeyStore.jks
  -label MyCert
```

A password is required to access this key database.
Please enter a password:

```
Label: MyCert
Key Size: 1024
Version: X509 V3
Serial Number: 4A D7 39 1F
Issued By: MyBroker.Server
ISSW
IBM
Hursley, GB
Subject: MyBroker.Server
ISSW
IBM
Hursley, GB
Valid From: 15 October 2009 16:00:47 o'clock BST To: 15 October 2010 16:00:47 o'clock BST
Fingerprint: 98:5D:C4:70:A0:28:84:72:FB:F6:3A:D2:D2:F5:EE:8D:30:33:87:82
Signature Algorithm: 1.2.840.113549.1.1.4
Trust Status: enabled
```

Extracting a certificate:

Generate a copy of a self-signed certificate that you can import as a trusted (or signer certificate) into a truststore file. Use this procedure only for testing, not production.

About this task

Certificates can be extracted in two formats:

- Base64-encoded ASCII data (.arm). This format is convenient for inclusion in XML messages, and transmission over the Internet.
- Binary DER data (.der).

Procedure

Issue the following command:

```
gsk7cmd -cert -extract
  -db keystore_name
  -pw keystore_passwd
  -label label
  -target file_name
  [-format ascii | binary]
```

For example:

```

gsk7cmd -cert -extract
        -db myBrokerKeystore.jks
        -pw myKeyPass
        -label MyCert
        -target MyCert.arm
        -format ascii

```

You can then view the certificate in a text editor, such as Notepad:

```

notepad MyCert.arm
-----BEGIN CERTIFICATE-----
MIICIZCCAAYygAwIBAgIEStc5HzANBgkqhkiG9w0BAQQFADBWMQswCQYDVQQGEwJHQjEQMA4GA1UE
BxMHSVyc2xleTEMMAoGA1UEChMDSUJNMQ0wCwYDVQQLEwRlRlU1NXMRgwFgYDVQQDEw9NeUJyb2t1
ci5TZXJ2ZXIwHhcNMDkxMDE1MTUwMDQ3WhcNMTAxMDE1MTUwMDQ3WjBWMQswCQYDVQQGEwJHQjEQ
MA4GA1UEBxMHSVyc2xleTEMMAoGA1UEChMDSUJNMQ0wCwYDVQQLEwRlRlU1NXMRgwFgYDVQQDEw9N
eUJyb2t1ci5TZXJ2ZXIwZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMwkK5kFLwC29YsHLX1f
hd0CgqFeytH1I0sZesdi8hEPXKs0zs30Qta2b0GZyUbBkh4tNeUHNWE9o7Hx2/SfziPQRKUw908R
F/6FPaHGezRkkaLJGX3uEhjt/2+n5t0JGytnKwaWJTpzdmZ79c0XjFv083q3yXPYjKzq8rS1iVBf
AgMBAAEwDQYJKoZIhvcNAQEEBQADgYEAQejpvZkjRcg3AHqY4RwBSMtXVWFFyoHSbjymR8IdURoQ
DCGZ2jjsv3kxQLADaCX0BYgohGJAHS7PzkQoHUCiHR0kusyuAt1MNYbhEcs+BYAzvsSz1ay4oiqUCi
Qs3aeNLV0b9c1RyzbuKYZ10uX59GAfGVLvyk6vQ/g7wPVL4TVgc=
-----END CERTIFICATE-----

```

Adding a signer certificate to the truststore:

Add a signer certificate to the truststore of a broker or integration server.

About this task

The following steps show how to add an extracted certificate as signer certificate to the truststore file. Adding the broker self-signed certificate to a broker or integration server truststore enables request nodes (HTTP or SOAP) to send test messages to input nodes (HTTP or SOAP) when the flows are running on the broker or integration server.

Procedure

Issue the following command:

```

gsk7cmd -cert -add
        -db truststore_name
        [-pw password]
        -label label
        -file file_name
        -format [ascii | binary]

```

For example:

```

gsk7cmd -cert -add
        -db myBrokerTruststore.jks
        -label CACert
        -file TRUSTEDPublicCertificate.arm
        -format ascii

```

You can view details of the certificate:

```

gsk7cmd -cert -details -db myBrokerTruststore.jks -label CACert
A password is required to access this key database.
Please enter a password:

```

```

Label: CACert
Key Size: 1024

```


Version: X509 V3
Serial Number: 49 49 23 1B
Issued By: VSR1BK
ISSW
IBM
GB
Subject: VSR1BK
ISSW
IBM
GB
Valid From: 17 December 2008 16:04:43 o'clock GMT To: 17 December 2009 16:04:43
o'clock GMT
Fingerprint: CB:39:E7:D8:1D:C0:00:A1:3D:B1:97:69:7A:A7:77:19:6D:09:C2:A7
Signature Algorithm: 1.2.840.113549.1.1.4
Trust Status: enabled

Listing all certificates in a keystore: Procedure

Issue the following command:

```
gsk7cmd -cert -list  
-db keystore_name
```

For example:

```
gsk7cmd -cert -list  
-db myBrokerKeystore.jks
```

A password is required to access this key database.
Please enter a password:

```
Certificates in database: myBrokerKeystore.jks  
verisign class 1 public primary certification authority - g3  
verisign class 4 public primary certification authority - g3  
verisign class 1 public primary certification authority - g2  
verisign class 4 public primary certification authority - g2  
verisign class 2 public primary certification authority  
entrust.net global client certification authority  
rsa secure server certification authority  
verisign class 2 public primary certification authority - g3  
verisign class 2 public primary certification authority - g2  
verisign class 3 secure server ca  
verisign class 3 public primary certification authority  
verisign class 3 public primary certification authority - g3  
verisign class 3 public primary certification authority - g2  
thawte premium server ca  
verisign class 1 public primary certification authority  
entrust.net global secure server certification authority  
thawte personal basic ca  
thawte personal premium ca  
thawte personal freemail ca  
verisign international server ca - class 3  
thawte server ca  
entrust.net certification authority (2048)  
cacert  
entrust.net client certification authority  
entrust.net secure server certification authority  
soaplistener  
mycert
```

Configuring PKI at broker level:

Define the broker registry properties that identify the location, name, and password of the keystore and truststore files.

About this task

These settings are used as the default settings for the broker-wide HTTP listener and all embedded HTTP listeners in integration servers on the broker. These settings can be overridden for the broker-wide HTTP listener (see “Configuring PKI for the broker-wide HTTP listener”) and for individual embedded HTTP listeners (see “Configuring PKI for an embedded HTTP listener in an integration server” on page 422).

1. Start the broker:

```
mqsistart broker_name
```

2. Display the current settings of the broker registry properties:

```
mqsireportproperties broker_name  
-o BrokerRegistry  
-r
```

3. Set the keystore property:

```
mqsichangeproperties broker_name  
-o BrokerRegistry  
-n brokerKeystoreFile  
-v install_dir\MQSI\9.0\MyBrokerKeystore.jks
```

4. Set the truststore property:

```
mqsichangeproperties broker_name  
-o BrokerRegistry  
-n brokerTruststoreFile  
-v install_dir\MQSI\9.0\MyBrokerTruststore.jks
```

5. Stop the broker:

```
mqsistop broker_name
```

6. Set the password for the keystore:

```
mqsisetdbparms broker_name  
-n brokerKeystore::password  
-u ignore  
-p keystore_pass
```

7. Set the password for the truststore:

```
mqsisetdbparms broker_name  
-n brokerTruststore::password  
-u ignore  
-p truststore_pass
```

8. Start the broker:

```
mqsistart broker_name
```

9. Display and verify the broker registry properties:

```
mqsireportproperties broker_name  
-o BrokerRegistry  
-r
```

Configuring PKI for the broker-wide HTTP listener:

Define the properties for the broker-wide HTTP listener to identify the location, name, and password of the keystore and truststore files.

About this task

These settings override any PKI configuration that is set at the broker level. If you enable SSL on the broker-wide HTTP listener but do not set the following properties, then the broker-level settings are applied, see “Configuring PKI at broker level” on page 421.

1. Start the broker.

```
mqsistart broker_name
```

2. Display the current settings of the broker-wide listener properties.

```
mqsireportproperties broker_name  
-b httplistener  
-o HTTPSConnector  
-a
```

3. Set the keystore property.

```
mqsichangeproperties broker_name  
-b httplistener  
-o HTTPSConnector  
-n keystoreFile  
-v install_dir\MQSI\9.0\MyBrokerKeystore.jks
```

4. Set the truststore property.

```
mqsichangeproperties broker_name  
-b httplistener  
-o HTTPSConnector  
-n truststoreFile  
-v install_dir\MQSI\9.0\MyBrokerTruststore.jks
```

5. Stop the broker:

```
mqsistop broker_name
```

6. Set the password for the keystore.

```
mqsichangeproperties broker_name  
-b httplistener  
-o HTTPSConnector  
-n keystorePass  
-v keystore_pass
```

7. Set the password for the truststore.

```
mqsichangeproperties broker_name  
-b httplistener  
-o HTTPSConnector  
-n truststorePass  
-v truststore_pass
```

8. Start the broker:

```
mqsistart broker_name
```

9. Display and verify the broker-wide listener properties.

```
mqsireportproperties broker_name  
-b httplistener  
-o HTTPSConnector  
-a
```

Configuring PKI for an embedded HTTP listener in an integration server:

Define the ComIbmJVMManager properties for the required integration server to identify the location, name, and password of the keystore and truststore files.

About this task

These settings override any PKI configuration that is set at the broker level. If you enable SSL on an embedded HTTP listener but do not set the following properties, then the broker-level settings are applied, see “Configuring PKI at broker level” on page 421.

1. Start the broker.

```
mqsistart broker_name
```

2. Display the current settings of the ComIbmJVMMManager properties.

```
mqsireportproperties broker_name  
-e exec_grp_name  
-o ComIbmJVMMManager  
-r
```

3. Set the keystore property.

```
mqsichangeproperties broker_name  
-e exec_grp_name  
-o ComIbmJVMMManager  
-n keystoreFile  
-v install_dir\MQSI\9.0\MyExecGrpKeystore.jks
```

4. Set the keystore password key property. The value for this property is in the format *any_prefix_name*::password. This value is used to correlate the password that is defined in the **mqsisetdbparms** command.

```
mqsichangeproperties broker_name  
-e exec_grp_name  
-o ComIbmJVMMManager  
-n keystorePass  
-v exec_grp_nameKeystore::password
```

5. Set the truststore property.

```
mqsichangeproperties broker_name  
-e exec_grp_name  
-o ComIbmJVMMManager  
-n truststoreFile  
-v install_dir\MQSI\9.0\MyExecGrpTruststore.jks
```

6. Set the truststore password key property. The value for this property is in the format *any_prefix_name*::password. This value is used to correlate the password that is defined in the **mqsisetdbparms** command.

```
mqsichangeproperties broker_name  
-e exec_grp_name  
-o ComIbmJVMMManager  
-n truststorePass  
-v exec_grp_nameTruststore::password
```

7. Stop the broker.

```
mqsistop broker_name
```

8. Set the password for the keystore.

```
mqsisetdbparms broker_name  
-n exec_grp_nameKeystore::password  
-u ignore  
-p keystore_pass
```

9. Set the password for the truststore.

```
mqsisetdbparms broker_name  
-n exec_grp_nameTruststore::password  
-u ignore  
-p truststore_pass
```

10. Start the broker.
`mqsisstart broker_name`
11. Display and verify the ComIbmJVMMManager properties.
`mqsireportproperties broker_name`
`-e exec_grp_name`
`-o ComIbmJVMMManager`
`-r`

For information about cipher-suite requirements (such as the cryptographic algorithm and corresponding key lengths), see the Java Secure Socket Extension (JSSE) IBMJSSE2 Provider reference guide.

Related tasks:

“Implementing SSL authentication” on page 415

Use SSL authentication to enhance security in your broker environment.

“Configuring the broker to use SSL with JMS nodes” on page 365

Configure your broker to work with a JMS provider that supports JMS clients that can connect by using the Secure Sockets Layer (SSL) protocol.

“Configuring HTTPInput and HTTPReply nodes to use SSL (HTTPS)” on page 372

Configure the HTTPInput and HTTPReply nodes to communicate with other applications that use HTTPS by creating a keystore file, configuring the broker or integration server to use SSL, and creating a message flow to process HTTPS requests.

“Configuring the HTTPRequest and HTTPAsyncRequest nodes to use SSL (HTTPS)” on page 375

Configure the HTTPRequest or HTTPAsyncRequest nodes to communicate with other applications that use HTTP over SSL.

Related reference:

 Broker-wide HTTP listener parameters

Select the resources and properties that are associated with the broker-wide HTTP listener that you want to change.

Authorization by using SSL Client Certificates:

Client authentication data for SSL X509 certificates can be propagated into the local environment and used for authorization.

When you implement a message flow to use SSL authentication, you can check authenticated client certificates for authorization. When a security profile is configured for authorization and set on the input node, the data is passed to a security manager. A broker security manager receives relevant parts of the certificate for authorization and sends it to the properties parser. During authentication, data from the identity or security token that is provided replaces the values in the properties tree identity fields. This data can be from a Basic-Auth transport header or a WS-Security token, for example. Parameter data in the certificate replaces fields in the properties tree. The following fields are reset with new data:

- The **IdentitySourceType** is set to the user name.
- The **IdentitySourceToken** is set to the subject name of the client certificate.
- The **IdentitySourceIssuedBy** is set to the issuer name of the client certificate.

When you use the SOAPInput, HTTPInput, SCAInput, or TCPIPServerInput nodes, properties tree fields contain the information from the client certificate. Propagation

is not automatically enabled but when it is enabled, a certificate is processed throughout the message flow and propagated for output or request nodes. By populating the local environment, the certificate data becomes available to the rest of the message flow.

A higher level of authentication (such as Basic-Auth or WS-Security) can overwrite the properties tree. Because of missing properties tree data, you are unable to authorize the client at the input node. However, you can use a SecurityPEP node to locate authentication (or other certificate) fields in the local environment to do the authorization. You can locate client certificates by using the local variable `LocalEnvironment.input_node_name.Input.TransportSecurity.ClientAuth.Certificate`, where `input_node_name` is one of SOAP, HTTP, SCA, or TCPIP.

Different nodes access the client certificate in different ways:

- Use a SecurityPEP node for authorization of the **subject** field in a client certificate that is received by a SOAPInput node. Specify the following XPath in the identity token location property on a SecurityPEP node:

```
$LocalEnvironment/SOAP/Input/TransportSecurity/ClientAuth/Certificate/Subject
```

For more information about the SecurityPEP node, see SecurityPEP node.

- Use a JavaCompute node to retrieve the **issuer** field in a client certificate that is received by an HTTPInput node.

```
String clientCertSubject = localEnv.getFirstElementByPath("HTTP/Input/TransportSecurity/ClientAuth/Certificate/Issuer").getValueAsString();
```

For more information about the JavaCompute node, see JavaCompute node.

- Use a Compute node to set the **subject** field of a certificate that is received by a TCPIPServerInput node.

```
SET LocalEnvironment.TCPIP.Input.TransportSecurity.ClientAuth.Certificate.Subject = 'BROKERUSER';
```

For more information about the Compute node, see Compute node.

Related concepts:

“Digital certificates” on page 208

Certificates provide a way of authenticating users. Instead of requiring each participant in an application to authenticate every user, third-party authentication relies on the use of digital certificates.



Message tree structure

The message tree is a part of the logical message tree in which the broker stores its internal representation of the message body.



Local environment tree structure

The local environment tree is a part of the logical message tree in which you can store information while the message flow processes the message.

Related tasks:



Accessing the Properties tree

The Properties tree has its own correlation name, Properties, and you must use this in all ESQL statements that refer to or set the content of this tree.

“Setting up a public key infrastructure” on page 415

Configure keystores, truststores, passwords, and certificates to enable SSL communication, and Web Services Security.

“Configuring authorization” on page 342

Configure the broker to use an external security provider to authorize an identity in a message flow. You can use either Lightweight Directory Access Protocol (LDAP) or a WS-Trust V1.3 compliant security token server (STS) such as Tivoli Federated Identity Manager (TFIM) V6.2. Support for TFIM V6.1 is also provided for compatibility with previous versions.

Related reference:



SOAPInput node

Use the SOAPInput node to process client SOAP messages, so that the broker operates as a SOAP Web Services provider.



HTTPInput node

Use the HTTPInput node to receive an HTTP message from an HTTP client for processing by a message flow.



SCAInput node

Use the SCAInput node with the SCAReply node to process messages from WebSphere Process Server.



TCPIPServerInput node

Use the TCPIPServerInput node to create a server connection to a raw TCPIP socket, and to receive data over that connection.

Working with certificate revocation lists:

Certificate revocation lists (CRLs) provide a means for an SSL endpoint to verify that a certificate that is received from a remote host, and that is signed by a trusted certificate authority (CA), is still valid and trustworthy.

By using CRLs, CAs can manage the validity of the certificates they previously issued, when these certificates are published and in use. A CRL is a downloadable file that is maintained by the CA and contains a list of certificates that the CA issued but which the CA no longer considers trustworthy. Each entry in the CRL includes a certificate that has been revoked, a time limit (if the revocation applies for only a period of time), and a reason for the revocation. By checking that a presented certificate is not included in a recent CRL from the CA, the endpoint ensures that the CA still considers the certificate trustworthy.

You can configure HTTP-based output nodes (HTTPRequest, HTTPAsyncRequest, SOAPRequest, SOAPAsyncRequest, SCARRequest, and SCAAsyncRequest) to specify whether CRL checking should be enabled for SSL connections that are made on behalf of the nodes. You configure CRL checking by setting the Enable certificate revocation list checking property in the node. The default value is false.

For more information about the properties of these nodes, see the following documents:

- HTTPRequest node
- HTTPAsyncRequest node
- SOAPRequest node
- SOAPAsyncRequest node
- SCARRequest node
- SCAAsyncRequest node

You must enable CRL checking per node, and not per Java Virtual Machine (JVM). That is, the `com.ibm.jsse2.checkRevocation` JVM system property is ignored and the value of the node's `Enable certificate revocation list checking` property is used instead.

Configure the path to the CRL files that the JVM uses to check certificates by using the `broker.crlFileList` property. You can set this property at the level of the integration server or at the level of the integration node (broker). The value of the property is a list of paths to CRL files separated by the path separator character for the host operating system. You can set the value of this property by using the **mqsichangeproperties** command. For example:

```
mqsichangeproperties IB9NODE -o BrokerRegistry -n crlFileList -v file_path
```

where `IB9NODE` is the name of the node and *file_path* is the path to the CRL file.

You can configure the JVM to automatically download any CRL files (from CRL distribution points that are specified in the CA's digital signature) by setting the `com.ibm.security.enableCRLDP` property to `true`. The default value is `false`. You can set the value of this property by using the **mqsichangeproperties** command. For example:

```
mqsichangeproperties IB9NODE -e exgroup1 -o ComIbmJVManager -n enableCRLDP -v true
```

where `IB9NODE` is the name of the node and `exgroup1` is the name of the integration server.

You can use the `com.ibm.security.enableCRLDP` and `broker.crlFileList` properties together to enable automatic loading of CRLs.

You cannot check the revocation status of self-signed certificates because those certificates are by definition untrusted and they have no CA against which to check validity. If CRL checking is enabled on a node, and that node is presented with a self-signed certificate, the connection fails. By enabling CRL checking, you can prevent IBM Integration Bus from connecting to remote hosts that use untrusted certificates when trusted certificates are expected.

When you use IBM Integration Bus to receive HTTP requests, you can configure the HTTP listener to use a CRL file. The validity of client certificates is then checked against the CRL file before connections are accepted. You can configure the broker-wide listener to use a CRL file by using the following command:

```
mqsichangeproperties IB9NODE -b httpListener -o HTTPSConnector  
-n crlFile -v file_path
```

where `IB9NODE` is the name of the node and *file_path* is the path to the CRL file.

You can configure an HTTP listener that is embedded in an integration server to use a CRL file by using the following command:

```
mqsichangeproperties IB9NODE -e exgroup1 -n crlFile -v file_path
```

where `IB9NODE` is the name of the node, `exgroup1` is the name of the integration server, and *file_path* is the path to the CRL file.

Related tasks:

“Implementing SSL authentication” on page 415

Use SSL authentication to enhance security in your broker environment.

“Configuring SOAPRequest and SOAPAsyncRequest nodes to use SSL (HTTPS)” on page 370

Configure the SOAPRequest and SOAPAsyncRequest nodes to communicate with other applications that use HTTP over SSL.

“Configuring the HTTPRequest and HTTPAsyncRequest nodes to use SSL (HTTPS)” on page 375

Configure the HTTPRequest or HTTPAsyncRequest nodes to communicate with other applications that use HTTP over SSL.

Related reference:



mqschangeproperties command

Use the **mqschangeproperties** command to modify broker properties and properties of broker resources.

Listing SSL cipher suites:

You can view the available cipher suites in the IBM Integration Toolkit when you connect to a remote integration node (broker). You can also view a list of the cipher suites that are supported by IBM Integration Bus.

About this task

A cipher suite is a collection of algorithms that are used to encrypt data. During SSL authentication, the client and server compare cipher suites and select the first one that they have in common. If no suitable cipher suites exist, the server returns a handshake failure alert and closes the connection.

To use SSL encryption, you need a Java Secure Socket Extension (JSSE) provider. IBM Integration Bus supports the ciphers that are provided by the JSSE provider. For more information about Java security, see IBM Java security web page.

You can view the available cipher suites in the IBM Integration Toolkit when you connect to a remote integration node:

1. In the Integration Nodes view, right-click **Integration Nodes > Connect to a Remote Integration Node**.
2. Provide a queue manager name, host name, and port number, then click **Next**.
3. Click the arrow to the right of **Cipher suite**.

A list of the cipher suites that are available in the Toolkit is displayed.

For a list of all the cipher suites that are supported by IBM Integration Bus, see Appendix A of the IBM JSSE2 Guide.

Related tasks:

“Implementing SSL authentication” on page 415

Use SSL authentication to enhance security in your broker environment.

“Connecting to a remote broker” on page 4

To administer a remote broker by using the IBM Integration Toolkit or IBM Integration Explorer, you must connect to the broker.

“Connecting to a remote integration node on z/OS” on page 6

To administer a remote integration node (broker) that is deployed on z/OS by using the IBM Integration Toolkit or IBM Integration Explorer, you must connect to the integration node.

“Configuring TCP/IP client nodes to use SSL” on page 382

Configure a TCP/IP configuration to use SSL to secure connectivity to and from

the TCPIP client nodes.

“Configuring TCP/IP server nodes to use SSL” on page 384

Configure a TCP/IP configuration to use SSL to secure connectivity to and from the TCPIP server nodes.

Related reference:

 SOAPRequest node

Use the SOAPRequest node to send a SOAP request to the remote Web service.

 SOAPAsyncRequest node

Use the SOAPAsyncRequest node with the SOAPAsyncResponse node to construct a pair of message flows that call a Web service asynchronously.

 HTTPRequest node

Use the HTTPRequest node to interact with a web service.

IBM Java security web page

IBM JSSE2 Guide

Implementing SSL authentication on z/OS:

Use SSL authentication to enhance security in your broker environment.

About this task

Complete the following tasks to implement SSL authentication for brokers running on z/OS.

1. “Generate a broker certificate using RACF as the Certification Authority (z/OS)”
2. “Create and initialize the broker keystore and truststore (z/OS)” on page 431
3. “Configure IBM Integration Bus on z/OS for SSL” on page 432

Alternatively, you can enable SSL for IBM Integration Bus on z/OS by following the instructions in “Enabling SSL on z/OS IBM Integration Bus by using AT-TLS” on page 434.

Generate a broker certificate using RACF as the Certification Authority (z/OS):

You can use RACF as the Certification Authority (CA) for internal certificates in your enterprise.

About this task

To generate broker certificates, take the following steps:

1. Create the RACF CA signer certificate. This self-signed certificate is used to sign any other personal certificates created or requested in RACF. This step is required once.
2. Export the RACF CA signer certificate in CERTDER format. This certificate must be extracted without private keys; CERTDER is a binary format that guarantees that no private keys are exported.
3. Create the broker personal certificate. A copy of the certificate and of the private keys is maintained in RACF for future reissue or validation. This certificate must be associated with the broker user ID. Create a personal certificate for each broker or integration server for which you want to enable SSL.

- Export the broker personal certificate in PKCS12DER format. PKCS12DER is a password-protected, binary format that contains the broker certificate and its private keys. You will later import it into the broker keystore; see “Create and initialize the broker keystore and truststore (z/OS)” on page 431.

Example commands for each step are as follows:

Procedure

- Create the RACF CA signer certificate. For example:

```
RACDCERT CERTAUTH GENCERT +
  SUBJECTSDN(CN('RACF Cert Authority') T('PROD') +
  OU('RACF Group') +
  O('IBM') +
  L('HURSLEY') SP('WINCHESTER') C('GB')) +
  KEYUSAGE(CERTSIGN) +
  WITHLABEL('RACFCA') +
  NOTAFTER(DATE(2020/01/30)) +
  SIZE(1024)
```

- Export the RACFCA certificate in CERTDER format. For example:

```
RACDCERT CERTAUTH EXPORT(LABEL('RACFCA')) +
  DSN('CSQP.CSQPBRK.CACERT.DER') FORMAT(CERTDER)

OPUT 'CSQP.CSQPBRK.CACERT.DER' +
  '/u/CSQPBRK/ssl/csqpbrk.ca.der' +
  BINARY CONVERT(NO)
```

The OPUT command is optional. It is used to copy the certificate into a HFS file before FTP to another server.

- Create the broker personal certificate. For example:

```
RACDCERT ID(CSQPBRK) +
  GENCERT SUBJECTSDN(CN('BROKER.HTTP.CSQPBRK') T('PROD') +
  OU('ISSW') O('IBM') +
  L('HURSLEY') SP('WINCHESTER') C('GB')) +
  WITHLABEL('CSQPBRKCERT') SIZE(1024) +
  SIGNWITH(CERTAUTH LABEL('RACFCA')) +
  KEYUSAGE(HANDSHAKE DATAENCRYPT DOCSIGN) +
  NOTAFTER(DATE(2020/01/30))
```

- Export the broker certificate in PKCS12 format. For example:

```
RACDCERT ID(CSQPBRK) EXPORT(LABEL('CSQPBRKCERT')) +
  DSN('CSQP.CSQPBRK.PERSCERT.P12') +
  FORMAT(PKCS12DER) PASSWORD('changeit')

OPUT 'CSQP.CSQPBRK.PERSCERT.P12' +
  '/u/CSQPBRK/ssl/csqpbrk.pers.p12' +
  BINARY CONVERT(NO)
```

What to do next

What to do next: Create the broker keystore and import the personal certificate and RACF CA signer certificates.

Related tasks:

“Implementing SSL authentication on z/OS” on page 429

Use SSL authentication to enhance security in your broker environment.

“Create and initialize the broker keystore and truststore (z/OS)” on page 431

Create a keystore and import your personal certificate and signer certificates.

“Configure IBM Integration Bus on z/OS for SSL” on page 432
Define the location of the keystore and truststore, set passwords, and enable SSL.

Create and initialize the broker keystore and truststore (z/OS):

Create a keystore and import your personal certificate and signer certificates.

Before you begin

Before you start:

- Create the necessary certificates.

Note: Due to export restrictions, the IBM JDKs ship with a set of restricted policy files that limit the size of the cryptographic keys that are supported. To overcome these restrictions, use the unrestricted policy files in the `$JAVA_HOME/lib/security` directory:

- `local_policy.jar`
- `US_export_policy.jar`

The unrestricted policy files are the same for the IBM JDK 1.4.2, IBM JDK 5, and IBM JDK 6. These files are in the `JAVA_HOME/demo/jce/policy-files/unrestricted` directory.

About this task

This topic describes how to use the same file as keystore and truststore. To specify different files, complete the process twice:

- Do not import signer certificates into the keystore.
- Do not import personal certificates into the truststore.

The tasks use **keytool** to create the keystore. An alternative is the **ikeman** graphical tool, which requires an X Window System.

The following are the steps required to create and initialize the broker keystore:

1. Create the keystore. **keytool** requires a dummy key to be created to force the creation of the keystore file. The dummy key is deleted after the keystore is created.
2. Import the CA signer certificate or certificates. These are certificates that have signed certificates of client applications that connect to the IBM Integration Bus and that are accepted as trusted applications.

Example commands for each step are as follows:

Procedure

1. Create the JKS keystore. For example:

```
/usr/lpp/java/J6.0/bin/keytool -genkey
    -alias DUMMY
    -keystore /u/CSQPBRK/ssl/csqpbrkKeystore.jks
    -storetype jks
    -dname "CN=DUMMY,OU=BROKER,O=IBM,L=Hursley,C=GB"
    -storepass changeit
    -keypass changeit
```

2. Delete the dummy key. For example:

```
/usr/lpp/java/J6.0/bin/keytool -delete
  -alias DUMMY
  -keystore /u/CSQPBRK/ssl/csqpbrkKeystore.jks
  -storepass changeit
```

3. Optional: Import the CA signer certificates. Omit this step if you require separate files for a keystore and truststore, and are creating a keystore. For example:

```
/usr/lpp/java/J6.0/bin/keytool -import
  -keystore /u/CSQPBRK/ssl/csqpbrkKeystore.jks
  -storepass changeit
  -alias RACFCA
  -file /u/CSQPBRK/ssl/csqpbrk.ca.der -v
```

4. Optional: Import the broker personal certificate. Omit this step if you require separate files for a keystore and truststore, and are creating a truststore. For example:

```
/usr/lpp/java/J6.0/bin/keytool -import
  -keystore /u/CSQPBRK/ssl/csqpbrkKeystore.jks
  -storepass changeit
  -alias CSQPBRK
  -file /u/CSQPBRK/ssl/csqpbrk.pers.p12
  -v
  -pkcs12
  -keypass changeit
  -noprompt
```

5. List the contents of the broker keystore. For example:

```
/usr/lpp/java/J6.0/bin/keytool -list
  -keystore /u/CSQPBRK/ssl/csqpbrkKeystore.jks
  -storepass changeit
```

What to do next

What to do next: “Configure IBM Integration Bus on z/OS for SSL.”

Related tasks:

“Implementing SSL authentication on z/OS” on page 429

Use SSL authentication to enhance security in your broker environment.

“Generate a broker certificate using RACF as the Certification Authority (z/OS)” on page 429

You can use RACF as the Certification Authority (CA) for internal certificates in your enterprise.

“Configure IBM Integration Bus on z/OS for SSL”

Define the location of the keystore and truststore, set passwords, and enable SSL.

Configure IBM Integration Bus on z/OS for SSL:

Define the location of the keystore and truststore, set passwords, and enable SSL.

Before you begin

Before you start: complete the following tasks:

- “Generate a broker certificate using RACF as the Certification Authority (z/OS)” on page 429
- “Create and initialize the broker keystore and truststore (z/OS)” on page 431

About this task

The process is essentially the same as on Windows and UNIX. This topic describes how to enable SSL at broker level; it can also be done at integration server level for the SOAP nodes. See “Configuring SOAPInput and SOAPReply nodes to use SSL (HTTPS)” on page 366 and “Configuring SOAPRequest and SOAPAsyncRequest nodes to use SSL (HTTPS)” on page 370 for a description of the process on distributed platforms.

To execute the following commands, you can run the BIPCHPR job in the broker component library.

Procedure

1. Define the location of the keystore. This example shows how to define a keystore at broker level. For example:

```
BPXBATSL PGM -  
  /usr/lpp/mqsi/V9R0M0/bin/-  
mqsichangeproperties -  
  CSQPBRK -  
  -o BrokerRegistry -  
  -n brokerKeystoreFile -  
  -v /u/CSQPBRK/ssl/csqrkKeystore.jks
```

2. Define the location of the truststore. For example:

```
BPXBATSL PGM -  
  /usr/lpp/mqsi/V9R0M0/bin/-  
mqsichangeproperties -  
  CSQPBRK -  
  -o BrokerRegistry -  
  -n brokerTruststoreFile -  
  -v /u/CSQPBRK/ssl/csqrkKeystore.jks
```

3. Enable the HTTPS Connector. For example:

```
BPXBATSL PGM -  
  /usr/lpp/mqsi/V9R0M0/bin/-  
mqsichangeproperties -  
  CSQPBRK -  
  -b httpListener -  
  -o HTTPListener -  
  -n enableSSLConnector -  
  -v true
```

4. Optional: Enable client authentication. For example:

```
BPXBATSL PGM -  
  /usr/lpp/mqsi/V9R0M0/bin/-  
mqsichangeproperties -  
  CSQPBRK -  
  -b httpListener -  
  -o HTTPSConnector -  
  -n clientAuth -  
  -v true
```

5. Stop the broker. You must stop the broker before you can define passwords.

6. Define the keystore password. For example:

```
BPXBATSL PGM -  
  /usr/lpp/mqsi/V9R0M0/bin/-  
mqsisetdbparms -
```


- ```

CSQPBRK -
-n brokerKeystore::password -
-u ignore -
-p changeit

```
7. Define the truststore password. For example:
- ```

BPXBATSL PGM -
 /usr/lpp/mqsi/V9R0M0/bin/-
mqsisetdbparms -
 CSQPBRK -
-n brokerTruststore::password -
-u ignore -
-p changeit

```
8. Start the broker.
9. Verify and test your configuration.

Related tasks:

“Implementing SSL authentication on z/OS” on page 429

Use SSL authentication to enhance security in your broker environment.

“Generate a broker certificate using RACF as the Certification Authority (z/OS)” on page 429

You can use RACF as the Certification Authority (CA) for internal certificates in your enterprise.

“Create and initialize the broker keystore and truststore (z/OS)” on page 431

Create a keystore and import your personal certificate and signer certificates.

“Starting and stopping a broker on z/OS” on page 23

Run the appropriate command from SDSF to start or stop a broker.

Enabling SSL on z/OS IBM Integration Bus by using AT-TLS:

You can use Application Transparent Transport Layer Security (AT-TLS) to provide Secure Sockets Layer (SSL) services on behalf of IBM Integration Bus on z/OS. AT-TLS is part of z/OS Communication Server.

About this task

You can enable SSL by following the instructions in “Implementing SSL authentication on z/OS” on page 429. This topic describes an alternative method that uses AT-TLS to enable SSL without the need to complete configuration steps in IBM Integration Bus. AT-TLS provides the following benefits when using SSL/TLS protocols with IBM Integration Bus on z/OS:

- AT-TLS uses RACF key rings and certificates.
- The Policy Agent (PAGENT) manages the rules and policies that define how SSL is used to connect to IBM Integration Bus.
- PAGENT can distribute the rules and policies in a z/OS SYSPLEX environment.
- The IBM Integration Bus
- HTTP or SOAP nodes in message flows can have standard HTTP settings (no SSL/HTTPS).

To configure AT-TLS in your z/OS environment for IBM Integration Bus , complete the following steps:

Procedure

1. Create a RACF key ring by following the instructions in “Creating a RACF key ring” on page 437.

2. Configure and activate PAGENT by following the instructions in “Configuring and activating the policy agent (PAGENT)” on page 439.
3. Define and install AT-TLS policies for IBM Integration Bus by following the instructions in “Defining and installing AT-TLS policies” on page 441.
4. Test and verify AT-TLS by using IBM Integration Bus, as described in “Testing and verifying AT-TLS” on page 444.

Related concepts:

“Application Transparent Transport Layer Security”

AT-TLS is a service provided by the z/OS Communication Server Policy Agent (PAGENT) and the TCP/IP stack. The AT-TLS service manages SSL connections on behalf of applications that are running on z/OS. The z/OS applications are unaware that SSL is used in the connection with partner applications.

Related tasks:

“Creating a RACF key ring” on page 437

To create a RACF key ring, you must first generate a RACF CA certificate and a personal certificate for IBM Integration Bus, then connect the certificates to the key ring.

“Configuring and activating the policy agent (PAGENT)” on page 439

Configure PAGENT by updating the TCP/IP profile, granting RACF permission to TCP/IP resources, preparing the PAGENT startup JCL, and activating syslogd.

“Defining and installing AT-TLS policies” on page 441

Define and install AT-TLS policies by using the IBM Configuration Assistant for z/OS Communications Server.

“Testing and verifying AT-TLS” on page 444

Test and verify AT-TLS by using the SOAP Nodes sample.

“Diagnosing problems with PAGENT and AT-TLS” on page 446

To help with problem determination with PAGENT and AT-TLS, activate tracing and logging.

Application Transparent Transport Layer Security:

AT-TLS is a service provided by the z/OS Communication Server Policy Agent (PAGENT) and the TCP/IP stack. The AT-TLS service manages SSL connections on behalf of applications that are running on z/OS. The z/OS applications are unaware that SSL is used in the connection with partner applications.

The following diagram shows how AT-TLS works. The numbers in the diagram represent the steps that follow the diagram.

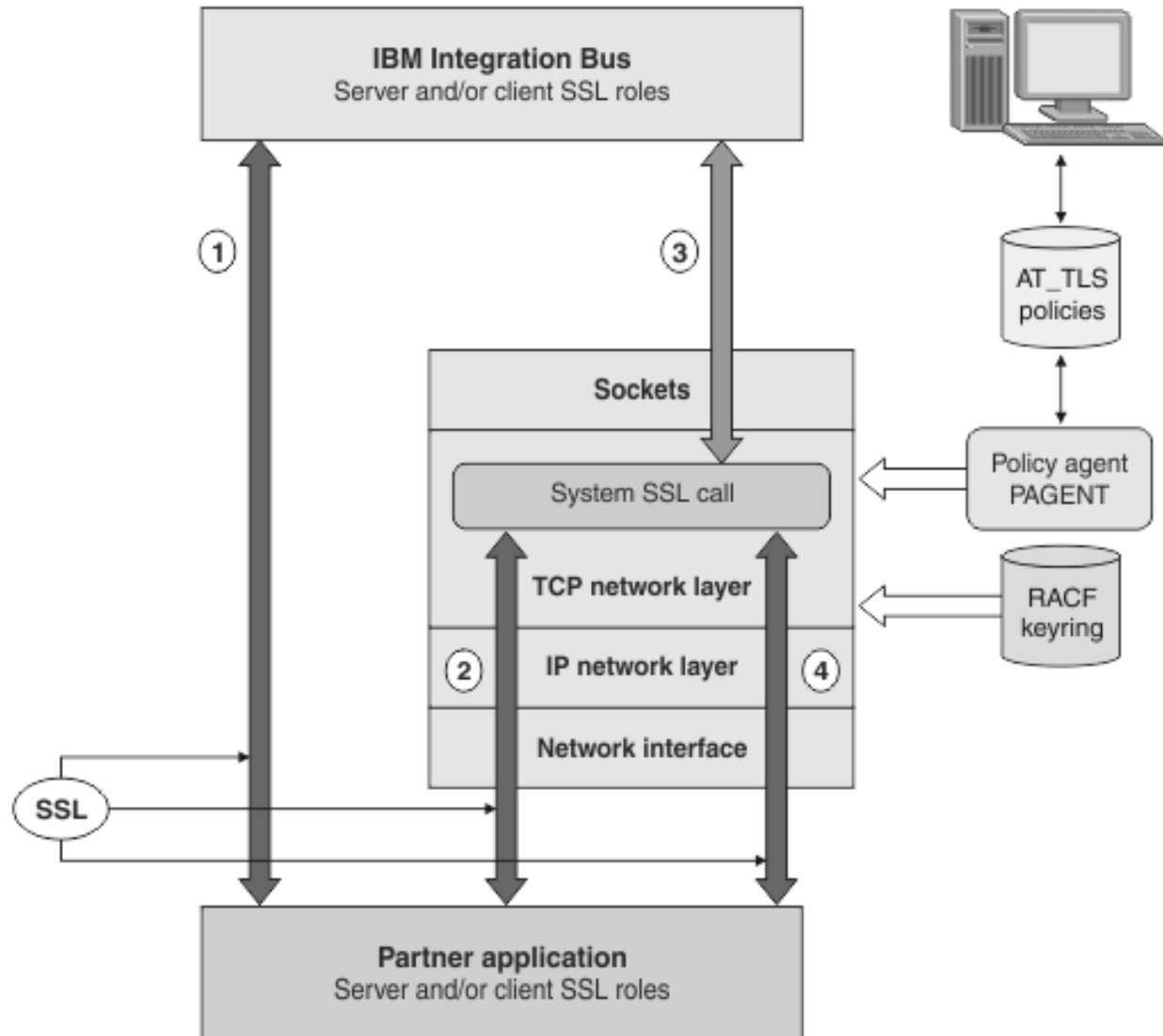


Figure 24. The diagram is described in the surrounding text.

1. Step 1 represents an SSL connection when AT-TLS is not used, which requires that IBM Integration Bus and the partner application are both enabled for SSL.
2. The SSL handshake is managed by AT-TLS in the TCP layer.
3. Inbound or outbound application data is received or sent in the clear by IBM Integration Bus. The TCP layer validates and decrypts inbound data from partner applications, or encrypts outbound data to partner applications.
4. Inbound or outbound application data is protected by SSL.

AT-TLS components

RACF key ring

The RACF key ring contains the IBM Integration Bus personal certificate and the partner application signer certificate.

AT-TLS policies

This file contains the rules and policies that control the SSL connections

that are managed by AT-TLS. These policies are created by the network administrator, and are checked and enforced by the TCP network layer of the TCP/IP stack.

Policy Agent

This component manages and distributes network policies, including AT-TLS policies, to the TCP/IP stack or stacks. The policy agent is also called PAGENT. For AT-TLS to function successfully, PAGENT must be configured correctly and operational.

TCP/IP stack

The TCP/IP stack is the component that implements the AT-TLS services. The TCP network layer is where SSL connections are intercepted, the SSL handshake is performed, and data is decrypted and encrypted. The TCP/IP stack uses RACF services to validate and accept certificates that are presented by the partner application during the handshake. RACF retrieves the IBM Integration Bus personal certificate from the key ring.

Related tasks:

“Enabling SSL on z/OS IBM Integration Bus by using AT-TLS” on page 434

You can use Application Transparent Transport Layer Security (AT-TLS) to provide Secure Sockets Layer (SSL) services on behalf of IBM Integration Bus on z/OS. AT-TLS is part of z/OS Communication Server.

“Creating a RACF key ring”

To create a RACF key ring, you must first generate a RACF CA certificate and a personal certificate for IBM Integration Bus, then connect the certificates to the key ring.

“Configuring and activating the policy agent (PAGENT)” on page 439

Configure PAGENT by updating the TCP/IP profile, granting RACF permission to TCP/IP resources, preparing the PAGENT startup JCL, and activating syslogd.

“Defining and installing AT-TLS policies” on page 441

Define and install AT-TLS policies by using the IBM Configuration Assistant for z/OS Communications Server.

“Testing and verifying AT-TLS” on page 444

Test and verify AT-TLS by using the SOAP Nodes sample.

“Diagnosing problems with PAGENT and AT-TLS” on page 446

To help with problem determination with PAGENT and AT-TLS, activate tracing and logging.

Creating a RACF key ring:

To create a RACF key ring, you must first generate a RACF CA certificate and a personal certificate for IBM Integration Bus, then connect the certificates to the key ring.

About this task

Each RACF key ring has its own name up to 237 characters long and is associated with a user ID. A RACF key ring is connected to a set of personal certificates and trusted certificates that are stored in the RACF database. The RACF command **RACDCERT** is used to create and delete key rings and to connect or disconnect certificates to the key rings. RACF key rings are also called System Authorization Facility (SAF) key rings. SAF is an open standard to access security services.

To create a RACF key ring to be used by AT-TLS on behalf of IBM Integration Bus, complete the following steps.

Procedure

1. Generate a RACF certificate authority (CA) certificate.

You can use RACF as a CA to generate and sign personal certificates for their internal systems or applications. This certificate must be created once, and it is used to sign every personal certificate that is generated by RACF. The following example shows how to use a RACF command to generate a RACF CA certificate.

```
RACDCERT CERTAUTH GENCERT +
  SUBJECTSDN(CN('MQRootCA') +
  OU('ISSW') +
  O('IBM') +
  L('HURSLEY') SP('WINCHESTER') C('GB')) +
  KEYUSAGE(HANDSHAKE DATAENCRYPT DOCSIGN CERTSIGN) +
  WITHLABEL('MQRootCA') +
  NOTAFTER(DATE(2020/01/30)) +
  SIZE(1024)
```

2. Generate a personal certificate for IBM Integration Bus.

This certificate identifies a specific instance of IBM Integration Bus. This certificate is presented to the partner application during the SSL handshake. This certificate must be associated with the user ID under which IBM Integration Bus is running. The following example shows how to use a RACF command to generate the personal certificate for a broker called WI02BRK that is running under user ID WI02USR.

```
RACDCERT ID(WI02USR) +
  GENCERT SUBJECTSDN(CN('WI02BRK') +
  OU('ISSW') O('IBM') +
  L('HURSLEY') SP('WINCHESTER') C('GB')) +
  WITHLABEL('WI02BRK') SIZE(1024) +
  SIGNWITH(CERTAUTH LABEL('MQRootCA')) +
  KEYUSAGE(HANDSHAKE DATAENCRYPT DOCSIGN) +
  NOTAFTER(DATE(2012/01/30))
```

3. Create a RACF key ring and connect the certificates to the key ring.

The RACF key ring must be associated with a user ID (in this case, the IBM Integration Bus user ID). The key ring must have a name (in this case, the same name as the broker), and the IBM Integration Bus personal certificate must be connected to the key ring. The following example shows how to use a RACF command to create a key ring and connect the IBM Integration Bus personal certificate.

```
RACDCERT ID(WI02USR) ADDRING(WI02BRK)
RACDCERT ID(WI02USR) +
  CONNECT(ID(WI02USR) LABEL('WI02BRK') RING(WI02BRK))
RACDCERT ID(WI02USR) LISTRING(WI02BRK)
```

For RACF to validate a partner application certificate, you must import the signer certificate of the CA that generated and signed the personal certificate of the partner application. Typically, this certificate is extracted from the partner application keystore, transferred to z/OS as a data set (WI02USR.VSR1BK.DER), imported to RACF, and connected to the RACF key ring as signer (trusted) certificate. The following example shows how to use a RACF command to add a signer certificate to RACF and connect it to the RACF key ring.

```
RACDCERT CERTAUTH ADD('WI02USR.VSR1BK.DER') +
  WITHLABEL('VSR1BK') TRUST
RACDCERT CERTAUTH LIST(LABEL('VSR1BK'))
```

```
RACDCERT ID(WI02USR) +  
CONNECT(CERTAUTH LABEL('VSR1BK') RING(WI02BRK))  
RACDCERT ID(WI02USR) LISTRING(WI02BRK)
```

What to do next

Next: Configure and activate the policy agent by following the instructions in “Configuring and activating the policy agent (PAGENT).”

Related concepts:

“Application Transparent Transport Layer Security” on page 435

AT-TLS is a service provided by the z/OS Communication Server Policy Agent (PAGENT) and the TCP/IP stack. The AT-TLS service manages SSL connections on behalf of applications that are running on z/OS. The z/OS applications are unaware that SSL is used in the connection with partner applications.

Related tasks:

“Enabling SSL on z/OS IBM Integration Bus by using AT-TLS” on page 434

You can use Application Transparent Transport Layer Security (AT-TLS) to provide Secure Sockets Layer (SSL) services on behalf of IBM Integration Bus on z/OS. AT-TLS is part of z/OS Communication Server.

“Configuring and activating the policy agent (PAGENT)”

Configure PAGENT by updating the TCP/IP profile, granting RACF permission to TCP/IP resources, preparing the PAGENT startup JCL, and activating syslogd.

“Defining and installing AT-TLS policies” on page 441

Define and install AT-TLS policies by using the IBM Configuration Assistant for z/OS Communications Server.

“Testing and verifying AT-TLS” on page 444

Test and verify AT-TLS by using the SOAP Nodes sample.

“Diagnosing problems with PAGENT and AT-TLS” on page 446

To help with problem determination with PAGENT and AT-TLS, activate tracing and logging.

Configuring and activating the policy agent (PAGENT):

Configure PAGENT by updating the TCP/IP profile, granting RACF permission to TCP/IP resources, preparing the PAGENT startup JCL, and activating syslogd.

Before you begin

Before you start:

- Create a RACF key ring by following the instructions in “Creating a RACF key ring” on page 437.

About this task

To enable PAGENT for AT-TLS, complete the following steps. For a more detailed description of how to install and configure PAGENT, see the *Policy-based networking* chapter of the z/OS Communications Server IP Configuration Guide on the z/OS library web page.

Procedure

1. Update the TCP/IP profile.

You must make two changes to the TCP/IP profile to enable AT-TLS:

- Add the statement `TCPCONFIG TTLS` to activate the functionality of AT-TLS inside the TCP/IP stack.
 - Add `PAGENT` to the `AUTOLOG` list.
2. Grant RACF permissions to TCP/IP resources.
Users require permissions to the following resources as part of activating `PAGENT`:
 - a. Define `PAGENT` as a started task with its own user ID.
 - b. The `EZB.INITSTACK.sysname.tcpprocname` resource profile controls which users can have access to the TCP/IP stack before `PAGENT` is active. Give `READ` access to all users who do not require `PAGENT` policies to access the TCP/IP stack; for example, `PAGENT`, `NETVIEW`, `DB2`, and so on.
 - c. The `EZB.PAGENT.sysname.tcpprocname.*` resource controls which users can start, stop, and refresh `PAGENT`. Give `READ` access to the users who are allowed to run the TSO/Unix commands **Pagent** or **pasearch**.
 - d. The user ID of `PAGENT` must have `READ` access to the `BPX.DAEMON` facility.

For more detailed information about the RACF permissions, check the sample `EZARACF` in the `TCPIP.SEZAINST` library.

3. Prepare the `PAGENT` startup JCL.
 - a. Copy the sample JCL `PAGENT` in the `TCPIP.SEZAINST` library to the system procedure library (for example, `SYS1.PROCLIB`).
 - b. Edit the JCL according to your installation standards. Specify the location of the `PAGENT` configuration file (for example, `/etc/pagent/pagent.config`). You can specify the location and name of the configuration file by setting the environment variable `PAGENT_CONFIG_FILE=/etc/pagent/pagent.config`. The environment variables for the TCP/IP stack are usually specified in a member (for example, `ENVVARS`) of the TCP/IP parameters library (for example, `TCPIP.PARMS`). The `PAGENT` JCL has `ddname STDENV` that points to the member with the environment variables definitions.

The `PAGENT` configuration file (`/etc/pagent/pagent.config`) specifies the location and name of the `PAGENT` stack-specific configuration file by using the statement `TcpImage: TcpImage TCPIP /etc/pagent/TCPIP.image FLUSH NOPURGE 1800`.

The stack-specific configuration file (`/etc/pagent/TCPIP.image`) specifies the location and name of the AT-TLS policies file by using the statement `TTLSConfig: TTLSConfig /etc/pagent/TCPIP_TTLS.policy`.

4. Activate the system log daemon (`syslogd`).
`syslogd` acts as the central message logging facility for `PAGENT` and AT-TLS. `syslogd` is not specific to the policy infrastructure, but the policy infrastructure depends on `syslogd` to provide a central logging facility to maintain an audit trail. If you do not start `syslogd`, messages are lost. Start one `syslog` daemon per LPAR.

What to do next

Define and install AT-TLS policies for IBM Integration Bus by following the instructions in “Defining and installing AT-TLS policies” on page 441.

Related concepts:

“Application Transparent Transport Layer Security” on page 435

AT-TLS is a service provided by the z/OS Communication Server Policy Agent (PAGENT) and the TCP/IP stack. The AT-TLS service manages SSL connections on behalf of applications that are running on z/OS. The z/OS applications are unaware that SSL is used in the connection with partner applications.

Related tasks:

“Enabling SSL on z/OS IBM Integration Bus by using AT-TLS” on page 434

You can use Application Transparent Transport Layer Security (AT-TLS) to provide Secure Sockets Layer (SSL) services on behalf of IBM Integration Bus on z/OS. AT-TLS is part of z/OS Communication Server.

“Creating a RACF key ring” on page 437

To create a RACF key ring, you must first generate a RACF CA certificate and a personal certificate for IBM Integration Bus, then connect the certificates to the key ring.

“Defining and installing AT-TLS policies”

Define and install AT-TLS policies by using the IBM Configuration Assistant for z/OS Communications Server.

“Testing and verifying AT-TLS” on page 444

Test and verify AT-TLS by using the SOAP Nodes sample.

“Diagnosing problems with PAGENT and AT-TLS” on page 446

To help with problem determination with PAGENT and AT-TLS, activate tracing and logging.

Defining and installing AT-TLS policies:

Define and install AT-TLS policies by using the IBM Configuration Assistant for z/OS Communications Server.

Before you begin

Before you start:

- Configure and activate PAGENT by following the instructions in “Configuring and activating the policy agent (PAGENT)” on page 439.

About this task

You can create the AT-TLS policies by using the IBM Configuration Assistant for z/OS Communications Server, a Java application that you can download from IBM. Complete the following steps to define the policies required to enable SSL support on behalf of IBM Integration Bus for z/OS running SOAPInput and SOAPInput nodes:

Procedure

1. Start the configuration assistant by clicking **Start > All Programs > IBM Programs > IBM Configuration Assistant for z/OS > Configuration Assistant V1R10**.
2. Click **Add a New z/OS Image**, enter the name of your z/OS image (LPAR) and a description, then click **OK**.
3. In the Configuration Assistant Navigation pane, select the image that you added in step 2, click **Add New TCP/IP Stack**, enter the stack name and description, then click **OK**.
4. In the Configuration Assistant Navigation pane, select the stack that you added in step 3, select **AT-TLS** from the list of technologies, then click **Enable**.

5. Click **Configure**.
6. Click **Add**. The Connectivity Rule wizard opens. Click **Next**
7. Identify the data endpoints by completing the following fields. A generic rule facilitates testing, but can be made more specific later.
 - a. In the Local data endpoint field, select **ALL_IP_Addresses**.
 - b. In the Remote data endpoint field, select **ALL_IP_Addresses**.
 - c. In the Connectivity Rule Name field, enter a suffix for the name of the rules, then click **Next**.
8. Select a requirement map by clicking **Add**. The map is used to match the type of IP traffic with the security level to be implemented by AT-TLS.
9. Enter a name and description for the requirement map, then click **Work with Traffic Descriptors**. Two traffic descriptors are required: one for the inbound SOAP requests (IBM Integration Bus is the server), and another for the outbound SOAP requests (IBM Integration Bus is the client).
10. Create an inbound traffic descriptor by clicking **Add** , enter a name and description, then click **OK**.
11. Enter details about the inbound traffic descriptor:
 - a. For the local port, select **Single port** and set the port number to **7800** (the port on which the SOAPInput node normally listens).
 - b. For the remote port, select **All ports**.
 - c. Set the Indicate the TCP connect direction field to **Inbound only**.
 - d. In the Jobname field, enter an asterisk (*).
 - e. In the User ID field, enter an asterisk (*).
 - f. Select **Use the following key ring database**.
 - g. Select **Key ring is in SAF produce (such as RACF)**, then enter the name of the key ring.
 - h. Set the AT-TLS handshake role to **Server**, then click **AT-TLS Advanced**.
 - i. Enter the label of the IBM Integration Bus personal certificate, then click **OK**.
12. Click **OK** to save the traffic details for inbound SOAP traffic, then click **OK** to create the traffic descriptor for inbound SOAP.
13. Create an outbound traffic descriptor by clicking **Add**, add a name and description, then click **OK** .
14. Enter details about the outbound traffic descriptor:
 - a. For the local port, select **All ports**.
 - b. For the remote port, select **Single port** and set the port number to **7843**.
 - c. Set the Indicate the TCP connect direction to **Outbound only**.
 - d. In the Jobname field, enter an asterisk (*).
 - e. In the User ID field, enter an asterisk (*).
 - f. Select **Use the following key ring database**.
 - g. Select **Key ring is in SAF produce (such as RACF)**, then enter the name of the key ring.
 - h. Set the AT-TLS handshake role to **Client**, then click **AT-TLS Advanced**.
 - i. Enter the label of the IBM Integration Bus personal certificate, then click **OK**.
15. Click **OK** to save the traffic details for outbound SOAP traffic, then click **OK** to create the traffic descriptor for outbound SOAP.
16. Click **Close**.

17. To create a security level for IBM Integration Bus, click **Work with Security Levels**, then click **Add**.
 - a. On the Name and Type tab, enter a name and description.
 - b. On the Ciphers tab, select **Use TLS V1**, **Use SSL V3**, and **Use System SSL defaults**, then click **OK**.
18. To add traffic descriptors to the requirement map, select **SOAP_Server** and **SOAP_Client** from the Objects list, then click **Add**.
19. For each traffic descriptor, select the AT-TLS security level that you created in step 17, then click **OK**.
20. Click **Next** and set the appropriate Optional Connectivity Rule Settings, which are used to set tracing levels, tuning parameters, and timings when the rule is in effect..
21. Click **Finish**.
22. To save changes to the AT-TLS rules, click **Apply changes**, then click **Main perspective**.
23. To install the AT-TLS policy, select **AT-TLS technology**, click **Install**, then click **FTP** to send the policy rules to the LPAR.
24. Specify the FTP parameters:
 - a. Enter the LPAR host name and set the port number to 21.
 - b. Enter your user ID and password.
 - c. Enter the AT-TLS policy file location and name (for example, `/etc/pagent/TCP_IP_TLS.policy`).
 - d. Select **Default transfer mode**.
 - e. Click **Send**, wait for file transfer to complete, then check that the transfer was successful.
 - f. Click **Close**.
 - g. After the file transfer, refresh or restart PAGENT.

The AT-TLS policies have been created and deployed.

What to do next

Next: Test and verify AT-TLS for IBM Integration Bus by following the instructions in “Testing and verifying AT-TLS” on page 444.

Related concepts:

“Application Transparent Transport Layer Security” on page 435

AT-TLS is a service provided by the z/OS Communication Server Policy Agent (PAGENT) and the TCP/IP stack. The AT-TLS service manages SSL connections on behalf of applications that are running on z/OS. The z/OS applications are unaware that SSL is used in the connection with partner applications.

Related tasks:

“Enabling SSL on z/OS IBM Integration Bus by using AT-TLS” on page 434

You can use Application Transparent Transport Layer Security (AT-TLS) to provide Secure Sockets Layer (SSL) services on behalf of IBM Integration Bus on z/OS. AT-TLS is part of z/OS Communication Server.

“Creating a RACF key ring” on page 437

To create a RACF key ring, you must first generate a RACF CA certificate and a personal certificate for IBM Integration Bus, then connect the certificates to the key ring.

“Configuring and activating the policy agent (PAGENT)” on page 439

Configure PAGENT by updating the TCP/IP profile, granting RACF permission to

TCP/IP resources, preparing the PAGENT startup JCL, and activating syslogd.

“Testing and verifying AT-TLS”

Test and verify AT-TLS by using the SOAP Nodes sample.

“Diagnosing problems with PAGENT and AT-TLS” on page 446

To help with problem determination with PAGENT and AT-TLS, activate tracing and logging.

Testing and verifying AT-TLS:

Test and verify AT-TLS by using the SOAP Nodes sample.

Before you begin

Before you start:

Complete the following tasks:

- “Creating a RACF key ring” on page 437
- “Configuring and activating the policy agent (PAGENT)” on page 439
- “Defining and installing AT-TLS policies” on page 441

About this task

Use the following scenario to test and verify AT-TLS with IBM Integration Bus:

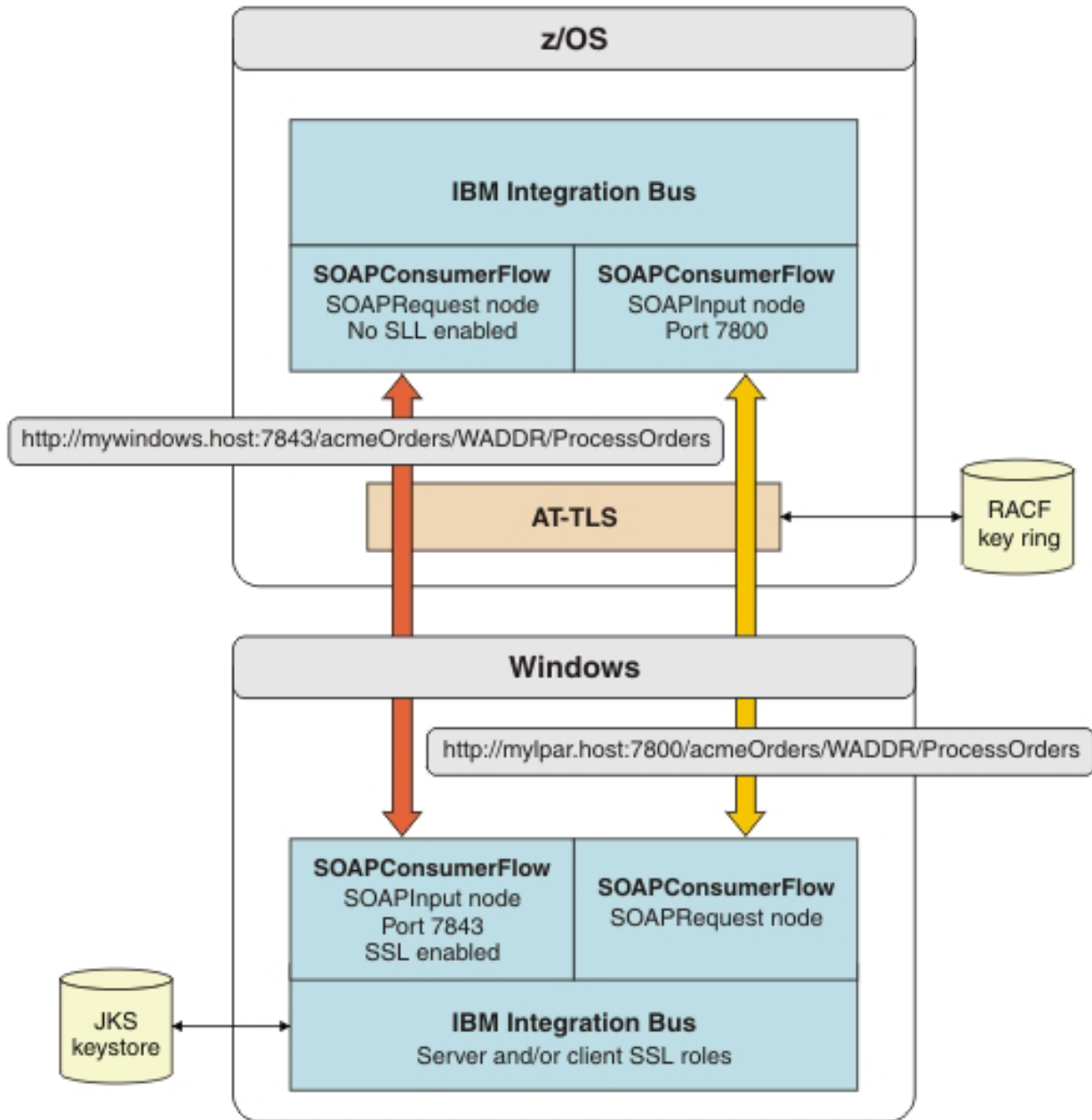


Figure 25. The diagram is described in the surrounding text.

This scenario has the following components:

- One instance of IBM Integration Bus is running on z/OS.
- Another instance of IBM Integration Bus is running on Windows.
- SOAP node sample flows from the IBM Integration Toolkit samples gallery are deployed to the two IBM Integration Bus instances.
- The broker instance running on z/OS is not enabled to use SSL. The SSL services are provided by AT-TLS.
- AT-TLS policies are active and PAGENT is running.
- The broker instance running on Windows is enabled to use SSL, and it has keystores defined with personal and signer certificates loaded.

The flows receive or send SOAP inbound and outbound requests between the two instances of IBM Integration Bus. The SOAP consumer flow in one broker sends requests to the SOAP provider flow in the other broker. The SOAP nodes on the Windows broker are SSL enabled. The scenario shows the URLs that are configured on the SOAPRequest node on each broker.

For more information about how to configure, deploy, and run the SOAP Nodes sample, see Samples.

Related concepts:

“Application Transparent Transport Layer Security” on page 435

AT-TLS is a service provided by the z/OS Communication Server Policy Agent (PAGENT) and the TCP/IP stack. The AT-TLS service manages SSL connections on behalf of applications that are running on z/OS. The z/OS applications are unaware that SSL is used in the connection with partner applications.

Related tasks:

“Enabling SSL on z/OS IBM Integration Bus by using AT-TLS” on page 434

You can use Application Transparent Transport Layer Security (AT-TLS) to provide Secure Sockets Layer (SSL) services on behalf of IBM Integration Bus on z/OS. AT-TLS is part of z/OS Communication Server.

“Creating a RACF key ring” on page 437

To create a RACF key ring, you must first generate a RACF CA certificate and a personal certificate for IBM Integration Bus, then connect the certificates to the key ring.

“Configuring and activating the policy agent (PAGENT)” on page 439

Configure PAGENT by updating the TCP/IP profile, granting RACF permission to TCP/IP resources, preparing the PAGENT startup JCL, and activating syslogd.

“Defining and installing AT-TLS policies” on page 441

Define and install AT-TLS policies by using the IBM Configuration Assistant for z/OS Communications Server.

“Diagnosing problems with PAGENT and AT-TLS”

To help with problem determination with PAGENT and AT-TLS, activate tracing and logging.

Diagnosing problems with PAGENT and AT-TLS:

To help with problem determination with PAGENT and AT-TLS, activate tracing and logging.

About this task

PAGENT has its own log file with the default name /tmp/pagent.log, which contains messages about loading AT-TLS rules. Invalid rules are rejected, and errors are written to the PAGENT log file. You can specify different levels of logging in the PAGENT configuration file by using the statement LogLevel. LogLevel 511 gives the most verbose logging.

The TCP/IP stack and the AT-TLS service log messages use SYSLOGD. The AT-TLS level of tracing is specified by using the advanced options in the connectivity rules. The highest (more verbose) level of tracing is 255.

The name and location of the log files are specified in the configuration file of the SYSLOGD (/etc/syslog.conf). The following example shows a configuration that can be used during testing:

```
*.* /var/log/%Y/%m/%d/errors
```

This syslogd configuration creates log files with names like /var/log/2010/03/20/errors. Every time syslogd is restarted, it creates a directory based on the current date. A good procedure is to restart syslogd once a day at midnight.

For more information, see the *Diagnosing Policy Agent problems* chapter of the z/OS Communications Server IP Diagnosis Guide on the z/OS library web page.

Related concepts:

“Application Transparent Transport Layer Security” on page 435

AT-TLS is a service provided by the z/OS Communication Server Policy Agent (PAGENT) and the TCP/IP stack. The AT-TLS service manages SSL connections on behalf of applications that are running on z/OS. The z/OS applications are unaware that SSL is used in the connection with partner applications.

Related tasks:

“Enabling SSL on z/OS IBM Integration Bus by using AT-TLS” on page 434

You can use Application Transparent Transport Layer Security (AT-TLS) to provide Secure Sockets Layer (SSL) services on behalf of IBM Integration Bus on z/OS. AT-TLS is part of z/OS Communication Server.

“Creating a RACF key ring” on page 437

To create a RACF key ring, you must first generate a RACF CA certificate and a personal certificate for IBM Integration Bus, then connect the certificates to the key ring.

“Configuring and activating the policy agent (PAGENT)” on page 439

Configure PAGENT by updating the TCP/IP profile, granting RACF permission to TCP/IP resources, preparing the PAGENT startup JCL, and activating syslogd.

“Defining and installing AT-TLS policies” on page 441

Define and install AT-TLS policies by using the IBM Configuration Assistant for z/OS Communications Server.

“Testing and verifying AT-TLS” on page 444

Test and verify AT-TLS by using the SOAP Nodes sample.

Chapter 3. Performance, monitoring, and workload management

You can change various aspects of your broker configuration to tune brokers and message flows, and monitor message flows and publish/subscribe applications.

About this task

Performance

The way in which you configure your broker environment can affect the performance of your applications. For more information about these options, see “Performance” on page 450.

Monitoring

You can configure your broker or message flows to generate data and statistics that you can use to assess behavior and performance. For more information about these options, see “Business-level monitoring” on page 565.

Statistics and accounting

You can collect message flow statistics and accounting data to record performance and operating details of one or more message flows. For more information, see “Message flow statistics and accounting data” on page 471.

Recording and viewing messages

You can configure a message flow to capture some of the data that it processes. You can then view this data from the web user interface or programmatically. For more information, see “Recording, viewing, and replaying data” on page 595.

Activity logging

Activity logs help users to understand what their message flows are doing by providing a high-level overview of how IBM Integration Bus interacts with external resources. For more information about activity logs, see “Using Activity logs” on page 623.

Workload management

System administrators can monitor and control the speed that messages are processed within message flows. For more information, see “Workload management” on page 633.

Related tasks:



Configuring brokers for development environments

Set up application development environments on Linux on x86 or Windows to create, test, and deploy message flows and associated resources.



Configuring brokers for test and production environments

Create one or more brokers on one or more computers, and configure them on your test and production systems to process messages that contain your business data.

Chapter 1, “Administering brokers and broker resources,” on page 1
Administering brokers and associated integration node resources includes the tasks that you perform frequently to activate and manage those resources. Choose the method you prefer to administer your brokers and associated resources.

Performance

You can use message flow design and coding techniques to optimize the performance of your message flows. You can then analyze statistical information for your message flows and modify aspects of your broker configuration to tune the performance of your brokers and message flows.

About this task

The way in which you configure your broker environment can affect the performance of your applications. Review the information in “Performance planning” on page 451, and decide which actions you can take to improve performance.

The following topics describe some design tips and techniques for optimizing performance, and explain how you can collect message flow and resource statistics and then use the data to help you tune performance:

- “Message flow design and performance” on page 453
- “Code design and performance” on page 455
- “Analyzing message flow performance” on page 469
- “Tuning message flow performance” on page 510
- “Analyzing resource performance” on page 529
- “Subscribing to statistics reports” on page 542
- “Tuning the broker” on page 543
- “Tuning the SAP adapter for scalability and performance” on page 558

The following topic contains links to scenarios that you can use to help you solve performance-related problems:

- “Troubleshooting performance problems” on page 560

The following topics provide reference information that you might find helpful when analyzing and tuning the performance of your broker:

- Message flow statistics and accounting data
- Resource statistics data
- Activity logs

Related tasks:



Configuring brokers for development environments

Set up application development environments on Linux on x86 or Windows to create, test, and deploy message flows and associated resources.



Configuring brokers for test and production environments

Create one or more brokers on one or more computers, and configure them on your test and production systems to process messages that contain your business data.

Chapter 1, “Administering brokers and broker resources,” on page 1
Administering brokers and associated integration node resources includes the tasks that you perform frequently to activate and manage those resources. Choose the method you prefer to administer your brokers and associated resources.

Related reference:



Performance and monitoring

Use the reference information in this section to accomplish the performance and monitoring tasks that address your business needs.

Performance planning

When you design your IBM Integration Bus environment and the associated resources, the decisions that you make can affect the performance of your applications.

About this task

If you are planning to create and maintain a small, stand-alone system with limited requirements for availability and performance, it is possible to achieve this relatively quickly. However, if you are planning a large or complex system, or if you have specific requirements for high availability or performance, it is worth taking time to understand the factors that can influence your design and the techniques that you can use to optimize it. As a starting point, consider the following aspects of your system and understand how these factors can influence performance:

Message flows

A message flow includes an input node that receives a message from an application over a particular protocol; for example, WebSphere MQ. The message must be parsed by the input node, and the performance impact of this parsing varies according to the parser that is used and the number of parses required. You can reduce the impact of parsing by using some optimization techniques such as parsing avoidance or partial parsing. For more information about parsing, see “Parsing and message flow performance” on page 466.

The number and length of message flows have implications for performance, and it is important to keep the number of nodes to a minimum. See “Message flow design and performance” on page 453 for more information about optimizing message flow performance. Other aspects of processing in a message flow that might affect performance are the amount, efficiency, and complexity of ESQL, access to databases, and how many message tree copies are made. For more guidance about these factors, see “Code design and performance” on page 455.

It is also important to consider how you split your business logic; how much work should the application do, and how much should the message flow do? Every interaction between an application and a message flow involves I/O and message parsing, and therefore adds to processing time. Design your message flows, and design or restructure your applications, to minimize these interactions.

For more information about these factors, see “Optimizing message flow response times” on page 554 and “Optimizing message flow throughput” on page 551.

Messages and message models

The type, format, and size of the messages that are processed can have a

significant effect on the performance of a message flow. For example, if you process persistent messages, they have to be stored for safekeeping. You might need to process messages with a well-defined structure; if so, you can create DFDL models for your messages. If you do not plan to interrogate the structure, you can work with undefined messages, such as BLOB messages.

If you are working in XML, be aware that it can be verbose, and therefore produce large messages, but XML message content is easier to understand than other formats, such as CWF. Field size and order might be important; these factors can be included in your DFDL model.

For more information about these factors, see “Optimizing message flow response times” on page 554 and Performance considerations for regular expressions in TDS messages.

Broker configuration

You can create and configure one or more brokers, on one or more computers, and for each broker you can create multiple integration servers, and multiple message flows. Your configuration decisions can influence message flow performance, and how efficiently messages can be processed.

For more information about these factors, see “Tuning the broker” on page 543, and “Optimizing message flow throughput” on page 551.

All these factors are examined in more detail in the Designing for Performance SupportPac (IP04).

For a description of common performance scenarios, review “Troubleshooting performance problems” on page 560.

For further articles about IBM Integration Bus and performance, review these sources:

- The Business Integration Zone on developerWorks. This site has a search facility; enter "performance" and review the links that are returned.
- The developerWorks article on message flow performance.
- The developerWorks article on WebSphere Message Broker Explorer accounting and statistics.
- The developerWorks article on monitoring resource use.

Related tasks:

“Message flow design and performance” on page 453

Consider the performance implications arising from the design of your message flow.

“Optimizing message flow throughput” on page 551

Each message flow that you design must provide a complete set of processing for messages received from a particular source. This design might result in very complex message flows that include large numbers of nodes that can cause a performance overhead, and might create potential bottlenecks. You can increase the number of message flows that process your messages to provide the opportunity for parallel processing and therefore improved throughput.

“Optimizing message flow response times” on page 554

You can use different solutions to improve message flow response times.

“Setting configuration timeout values” on page 548

Change timeout values that affect configuration tasks in the broker.

“Tuning the broker” on page 543

You can complete several tasks that enable you to tune different aspects of the broker performance.

“Resolving problems with performance” on page 823

Use the advice given here to help you to resolve common problems with performance.

“Analyzing message flow performance” on page 469

You can configure your broker to collect statistics about the operation and behavior of your message flows. Use this information to assess the performance of your message flows.

“Tuning message flow performance” on page 510

You can configure your message flows to enhance their performance in terms of resource usage, message flow throughput, and response times.

“Analyzing resource performance” on page 529

You can collect statistics to assess the performance of resources used by integration servers.

“Troubleshooting performance problems” on page 560

Follow this guidance to resolve common problems with IBM Integration Bus performance.

Related reference:



Message Sets: Performance considerations when using regular expressions
Take care when specifying regular expressions: some forms of regular expression can involve a large amount of work to find the best match, which might degrade performance.

Message flow design and performance

Consider the performance implications arising from the design of your message flow.

Before you begin

Before you start:

Read the following topics:

- “Performance” on page 450
- “Performance planning” on page 451
- “Tuning message flow performance” on page 510

About this task

To achieve the best performance from your message flows, consider the following points:

- Ensure that message flows are an appropriate length. Long message flows take longer to execute than short ones, but the use of many short message flows reduces performance more than the use of a few longer message flows.
- Use the minimum number of nodes required, and ensure that any subflows or loops are used appropriately.
- Avoid having consecutive ESQL Compute nodes in a message flow with no other nodes between them. Combine the logic into a single Compute node instead. Also avoid having consecutive JavaCompute nodes in the message flow.

- Minimize the volume of message parsing and adopt the most efficient parsing strategy; for more information, see “Parsing and message flow performance” on page 466.
- Custom canonical data formats provide a point of standardization, but they also involve additional CPU processing. The use of such formats typically results in two additional conversions per invocation of a message flow: from the external format into the canonical format and then from the canonical format to the final external format.
- If a message flow makes a synchronous call to an external application or service that can be slow or unpredictable in its response, it is more efficient to write the message flow using an asynchronous model.
- Ensure that your processing logic implements the coding tips provided in “Code design and performance” on page 455.

Related tasks:

“Performance” on page 450

You can use message flow design and coding techniques to optimize the performance of your message flows. You can then analyze statistical information for your message flows and modify aspects of your broker configuration to tune the performance of your brokers and message flows.

“Tuning message flow performance” on page 510

You can configure your message flows to enhance their performance in terms of resource usage, message flow throughput, and response times.



Developing message flows

Develop message flows to process your business messages and data. A message flow is a sequence of processing steps that run in an integration node when an input message is received.

“Analyzing message flow performance” on page 469

You can configure your broker to collect statistics about the operation and behavior of your message flows. Use this information to assess the performance of your message flows.

“Optimizing message flow throughput” on page 551

Each message flow that you design must provide a complete set of processing for messages received from a particular source. This design might result in very complex message flows that include large numbers of nodes that can cause a performance overhead, and might create potential bottlenecks. You can increase the number of message flows that process your messages to provide the opportunity for parallel processing and therefore improved throughput.

“Optimizing message flow response times” on page 554

You can use different solutions to improve message flow response times.

“Parsing and message flow performance” on page 466

Understand the impact of parsing on message flow performance, and use techniques to limit the effect and improve performance.

“Message tree navigation and copying” on page 467

Message tree navigation and message tree copying can reduce message flow performance, so it is important to use them appropriately and limit their usage where possible.

“ESQL code tips” on page 456

You can improve message flow performance with ESQL by using some optimization techniques.

“Java code tips” on page 462

You can improve message flow performance with Java by using some optimization techniques.

“XPath and XSLT code tips” on page 464

You can improve message flow performance with XPath and XSLT by using some optimization techniques.

“Identifying the cause of a slow message flow” on page 512

You can use the message flow statistics data to help you identify aspects of a message flow that might be reducing the performance of the flow, and to help you understand how you can optimize it.

“Comparing the performance of message flows in an application” on page 515

You can compare the performance of message flows in an application to identify the flows that might benefit from tuning, and to find out where any specific performance problems might exist.

“Code design and performance”

Consider the performance implications arising from the design of your code.

Code design and performance

Consider the performance implications arising from the design of your code.

Before you begin

Before you start:

Read the following topics:

- “Performance” on page 450
- “Performance planning” on page 451
- “Tuning message flow performance” on page 510

About this task

The following topics describe some of the design decisions that can influence the performance of your code:

- “ESQL code tips” on page 456
- “Java code tips” on page 462
- “XPath and XSLT code tips” on page 464
- “Parsing and message flow performance” on page 466
- “Message tree navigation and copying” on page 467

Related tasks:

“Performance” on page 450

You can use message flow design and coding techniques to optimize the performance of your message flows. You can then analyze statistical information for your message flows and modify aspects of your broker configuration to tune the performance of your brokers and message flows.

“Tuning message flow performance” on page 510

You can configure your message flows to enhance their performance in terms of resource usage, message flow throughput, and response times.



Developing message flows

Develop message flows to process your business messages and data. A message flow is a sequence of processing steps that run in an integration node when an input message is received.

“Optimizing message flow throughput” on page 551

Each message flow that you design must provide a complete set of processing for messages received from a particular source. This design might result in very complex message flows that include large numbers of nodes that can cause a performance overhead, and might create potential bottlenecks. You can increase the number of message flows that process your messages to provide the opportunity for parallel processing and therefore improved throughput.

“Optimizing message flow response times” on page 554

You can use different solutions to improve message flow response times.

“Parsing and message flow performance” on page 466

Understand the impact of parsing on message flow performance, and use techniques to limit the effect and improve performance.

“Message tree navigation and copying” on page 467

Message tree navigation and message tree copying can reduce message flow performance, so it is important to use them appropriately and limit their usage where possible.

“ESQL code tips”

You can improve message flow performance with ESQL by using some optimization techniques.

“Java code tips” on page 462

You can improve message flow performance with Java by using some optimization techniques.

“XPath and XSLT code tips” on page 464

You can improve message flow performance with XPath and XSLT by using some optimization techniques.

“Identifying the cause of a slow message flow” on page 512

You can use the message flow statistics data to help you identify aspects of a message flow that might be reducing the performance of the flow, and to help you understand how you can optimize it.

“Comparing the performance of message flows in an application” on page 515

You can compare the performance of message flows in an application to identify the flows that might benefit from tuning, and to find out where any specific performance problems might exist.

“Message flow design and performance” on page 453

Consider the performance implications arising from the design of your message flow.

ESQL code tips

You can improve message flow performance with ESQL by using some optimization techniques.

Before you begin

Before you start:

Read the following topics:

- “Performance” on page 450
- “Performance planning” on page 451
- “Tuning message flow performance” on page 510

About this task

When you write your ESQL code, you can use several techniques to improve the performance of your message flows. The following sections contain guidance about how to improve the performance of your ESQL code:

- “ESQL array processing”
- “ESQL CARDINALITY function” on page 458
- “ESQL DECLARE and EVAL statements” on page 458
- “ESQL PASSTHRU statement” on page 458
- “ESQL reference variables” on page 459
- “ESQL string functions” on page 459
- “Message trees with repeating records” on page 459

ESQL array processing:

About this task

Array subscripts [] are expensive in terms of performance because of the way in which subscript is evaluated dynamically at run time. By avoiding the use of array subscripts wherever possible, you can improve the performance of your ESQL code. You can use reference variables instead, which maintain a pointer into the array and which can then be reused; for example:

```
DECLARE myref REFERENCE TO InputRoot.XML.Invoice.Purchases.Item[1];
-- Continue processing for each item in the array
WHILE LASTMOVE(myref)=TRUE DO
  -- Add 1 to each item in the array
  SET myref = myref + 1;
  -- Do some processing
  -- Move the dynamic reference to the next item in the array
  MOVE myref NEXTSIBLING;
END WHILE;
```

ESQL array processing example:

The following example shows ESQL being used to process records read from a database. The repeated use of array subscripts such as `Environment.Variables.DBData[A]` increases the processing time significantly:

```
SET Environment.Variables.DBDATA[] =
(
SELECT T.*
FROM Database.{ 'ABC' }. { 'XYZ' } as T
);

DECLARE A INTEGER 1;
DECLARE B INTEGER CARDINALITY(Environment.Variables.*[]);
SET JPcntFODS = B;
WHILE A <= B DO
  CALL CopyMessageHeaders();
  CREATE FIELD OutputRoot.XML.FODS;
  DECLARE outRootRef REFERENCE TO OutputRoot.XML.Data;

  SET outRootRef.Field1 = Trim(Environment.Variables.DBDATA[A].Field1);
  SET outRootRef.Field2 = Trim(Environment.Variables.DBDATA[A].Field2);
  SET outRootRef.Field3 = Trim(Environment.Variables.DBDATA[A].Field3);
  SET outRootRef.Field4 = Trim(Environment.Variables.DBDATA[A].Field4);
  SET outRootRef.Field5 = Trim(Environment.Variables.DBDATA[A].Field5);
  . . .
  . . .
```

```

        SET outRootRef.Field37 = CAST(Environment.Variables.DBDATA[A].Field37)

        SET A = A + 1;
        PROPAGATE;
END WHILE;

```

You can reduce the processing time significantly by using reference variables

ESQL CARDINALITY function:

About this task

Avoid the use of CARDINALITY in a loop; for example:

```
WHILE ( I < CARDINALITY (InputRoot.MRM.A.B.C[]
```

The CARDINALITY function must be evaluated each time the loop is traversed, which is costly in performance terms. This is particularly true with large arrays because the loop is repeated more frequently. It is more efficient to determine the size of the array before the WHILE loop (unless it changes in the loop) so that it is evaluated only once; for example:

```
SET ARRAY_SIZE = CARDINALITY (InputRoot.MRM.A.B.C[])
WHILE ( I < ARRAY_SIZE )
```

ESQL DECLARE and EVAL statements:

About this task

Reduce the number of DECLARE statements (and therefore the performance cost) by declaring a variable and setting its initial value within a single statement. Alternatively, you can declare multiple variables of the same data type within a single ESQL statement rather than in multiple statements. This technique also helps to reduce memory usage.

The EVAL statement is sometimes used when there is a requirement to dynamically determine correlation names. However, it is expensive in terms of CPU use, because it involves the statement being run twice. The first time it runs, the component parts are determined, in order to construct the statement that will be run; then the statement that has been constructed is run.

ESQL PASSTHRU statement:

About this task

The following techniques can significantly improve performance when you are using PASSTHRU statements:

- Avoid the use of the PASSTHRU statement with a CALL statement to invoke a stored procedure. As an alternative, you can use the CREATE PROCEDURE ... EXTERNAL ... and CALL ... commands.
- When you are working with SQL statements that require literal or data values, use host variables, which map a column value to a variable. This enables dynamic SQL statements to be reused within the database. An SQL PREPARE on a dynamic statement is an expensive operation in performance terms, so it is more efficient to run this only once and then EXECUTE the statement repeatedly, rather than to PREPARE and EXECUTE every time.

For example, the following statement has two data and literal values, 100 and IBM:
 PASSTHRU('UPDATE SHAREPRICES AS SP SET Price = 100 WHERE SP.COMPANY = 'IBM');

This statement is effective when the price is 100 and the company is IBM. When either the *Price* or *Company* changes, another statement is required, with another SQL PREPARE statement, which impacts performance.

However, by using the following statement, *Price* and *Company* can change without requiring another statement or another PREPARE:

```
PASSTHRU('UPDATE SHAREPRICES AS SP SET Price = ? WHERE SP.COMPANY = ?',  
InputRoot.XML.Message.Price,InputRoot.XML.Message.Company);
```

You can check that dynamic SQL is achieving maximum statement reuse, by using the following commands to display the contents of the SQL statement cache in DB2:

```
db2 connect to <database name>  
db2 get snapshot for database on <database name>
```

Use the following commands to see the contents of the dynamic statement cache:

```
db2 connect to <database name>  
db2 get snapshot for dynamic SQL on <database name>
```

ESQL reference variables:

About this task

You can use reference variables to refer to long correlation names such as InputRoot.XMLNSC.A.B.C.D.E. Declare a reference pointer as shown in the following example:

```
DECLARE refPtr REFERENCE to InputRoot.XMLNSC.A.B.C.D;
```

To access element E of the message tree, use the correlation name refPtr.E.

You can use REFERENCE and MOVE statements to help reduce the amount of navigation within the message tree, which improves performance. This technique can be useful when you are constructing a large number of SET or CREATE statements; rather than navigating to the same branch in the tree, you can use a REFERENCE variable to establish a pointer to the branch and then use the MOVE statement to process one field at a time.

ESQL string functions:

About this task

String manipulation functions used within ESQL can be CPU intensive; functions such as LENGTH, SUBSTRING, and RTRIM must access individual bytes in the message tree. These functions are expensive in performance terms, so minimizing their use can help to improve performance. Where possible, also avoid executing the same concatenations repeatedly, by storing intermediate results in variables.

Message trees with repeating records:

About this task

Performance can be reduced under the following conditions:

- You are using ESQL processing to manipulate a large message tree
- The message tree consists of repeating records or many fields
- You have used explicit SET statements with field reference paths to access or create the fields

- You have observed a gradual slowing of message flow processing as the ESQL processes more fields or repetitions

This problem occurs when you use field references, rather than reference variables, to access or create consecutive fields or records.

The following example shows independent SET statements using field reference paths to manipulate the message tree. The SET statement takes a source and target parameter, where either or both parameters are field references:

```
SET OutputRoot.XMLNS.TestCase.StructureA.ParentA.field = '1';
```

Performance is affected by the SET statement being used to create many more fields, as shown in the following example:

```
SET OutputRoot.XMLNS.TestCase.StructureA.ParentA.field1 = '1';
SET OutputRoot.XMLNS.TestCase.StructureA.ParentA.field2 = '2';
SET OutputRoot.XMLNS.TestCase.StructureA.ParentA.field3 = '3';
SET OutputRoot.XMLNS.TestCase.StructureA.ParentA.field4 = '4';
SET OutputRoot.XMLNS.TestCase.StructureA.ParentA.field5 = '5';
```

In this example, the five fields that are created are all children of ParentA. Before the specified field can be created or modified, the broker must navigate the named message tree to locate the point in the message tree that is to be altered. For example:

- To access field 1, the SET statement navigates to ParentA, then to the first field, involving two navigations.
- To access field 5, the SET statement navigates to ParentA, then traverses each of the previous fields until it reaches field 5, involving six navigations.

Navigating over all the fields that precede the specified field causes the loss in performance.

The following example shows repeating fields being accessed in an input message tree:

```
DECLARE myChar CHAR;
DECLARE thisRecord INT 0;
WHILE thisRecord < 10000 DO
  SET thisRecord = thisRecord + 1;
  SET myChar = InputRoot.MRM.myParent.myRepeatingRecord[thisRecord];
END WHILE;
```

When index notation is used, as the count increases, the processing must navigate over all the preceding fields to get the required field; it must count over the previous records to get to the one that is represented by the current indexed reference.

- When accessing `InputRoot.MRM.myParent.myRepeatingRecord[1]`, one navigation takes place to get to the first record
- When accessing `InputRoot.MRM.myParent.myRepeatingRecord[2]`, two navigations take place to get to the second record
- When accessing `InputRoot.MRM.myParent.myRepeatingRecord[N]`, *N* navigations take place to get to the *N*-th record

Therefore, the total number of navigations for this WHILE loop is: $1 + 2 + 3 + \dots + N$, which is not linear.

If you are accessing or creating consecutive fields or records, you can solve this problem by using reference variables.

When you use reference variables, the statement navigates to the main parent, which maintains a pointer to the field in the message tree. The following example shows the ESQL that can be used to reduce the number of navigations when creating new output message tree fields:

```
SET OutputRoot.XMLNS.TestCase.StructureA.ParentA.field1 = '1';
DECLARE outRef REFERENCE TO OutputRoot.XMLNS.TestCase.StructureA.ParentA;
SET outRef.field2 = '2';
SET outRef.field3 = '3';
SET outRef.field4 = '4';
SET outRef.field5 = '5';
```

When referencing repeating input message tree fields, you can use the following ESQL:

```
DECLARE myChar CHAR;
DECLARE inputRef REFERENCE TO InputRoot.MRM.myParent.myRepeatingRecord[1];
WHILE LASTMOVE(inputRef) DO
  SET myChar = inputRef;
  MOVE inputRef NEXTSIBLING NAME 'myRepeatingRecord';
END WHILE;
```

For further information, see [Creating dynamic field references](#).

Related tasks:

[“Performance” on page 450](#)

You can use message flow design and coding techniques to optimize the performance of your message flows. You can then analyze statistical information for your message flows and modify aspects of your broker configuration to tune the performance of your brokers and message flows.

Chapter 3, [“Performance, monitoring, and workload management,” on page 449](#)
You can change various aspects of your broker configuration to tune brokers and message flows, and monitor message flows and publish/subscribe applications.

[“Performance planning” on page 451](#)

When you design your IBM Integration Bus environment and the associated resources, the decisions that you make can affect the performance of your applications.

[“Tuning the broker” on page 543](#)

You can complete several tasks that enable you to tune different aspects of the broker performance.

[“Tuning message flow performance” on page 510](#)

You can configure your message flows to enhance their performance in terms of resource usage, message flow throughput, and response times.

[“Analyzing message flow performance” on page 469](#)

You can configure your broker to collect statistics about the operation and behavior of your message flows. Use this information to assess the performance of your message flows.



Developing message flows

Develop message flows to process your business messages and data. A message flow is a sequence of processing steps that run in an integration node when an input message is received.

[“Code design and performance” on page 455](#)

Consider the performance implications arising from the design of your code.

Related reference:



Compute node

Use the Compute node to construct one or more new output messages.

Java code tips

You can improve message flow performance with Java by using some optimization techniques.

Before you begin

Before you start:

Read the following topics:

- “Performance” on page 450
- “Performance planning” on page 451
- “Tuning message flow performance” on page 510

About this task

A significant proportion of the performance problems relating to Java are caused by memory and garbage collection issues, by interaction with external resources, and by Java code that could benefit from optimization. Follow these steps to identify and solve such performance issues:

- Use the JVM resource statistics to review how much memory is in use by the JVM, and how often garbage collection is occurring in the integration server. For more information, see Resource statistics data: Java Virtual Machine (JVM).
- If you are calling external resources, review the elapsed time of the call.
- Review your Java code to identify any aspects that might benefit from optimization. There are several techniques that you can use to optimize the performance of your Java code in a message flow, including those described in the following sections:
 - “Tree references” on page 463
 - “Java string concatenation” on page 463
 - “Java BLOB processing” on page 463
 - “Java manual garbage collection” on page 463

The JavaCompute node enables you to include any valid Java processing, and it is common to reuse existing Java classes for specific functions. However, this code impacts the performance of the message flow, so ensure that any code used in this way is efficient.

Where possible, use the supplied nodes and functions of IBM Integration Bus to perform processing on a message, rather than recreating function that is already provided. For example, do not use a custom XML parser within the JavaCompute node. The XMLNSC parser that is provided with IBM Integration Bus is a high-performing parser that also offers validation against a schema, and the message tree that it generates can be used throughout the message flow. This is not the case with a custom parser with its own data structures.

Avoid starting new processes in the JavaCompute node to perform a specific piece of work. This is expensive in performance terms and results in additional CPU costs and an increased load on the system, because many short-lived processes are started and terminated when the work is completed. One of the benefits of the

IBM Integration Bus runtime is that processes are long lived and initialization is minimized. This reduces total processing costs, because new processes are not started each time the message flow is invoked.

For information about tools that you can use to assess the current status of a running Java application, see developerWorks IBM Monitoring and Diagnostic Tools for Java - Health Center Version 2.1.

Tree references:

About this task

Avoid building and navigating trees without storing intermediate references. For example, the following code repeatedly navigates from root to build the tree:

```
MbMessage newEnv = new MbMessage(inAssembly.getMessage());
newEnv.getRootElement().createElementAsFirstChild(MbElement.TYPE_NAME, "Destination", null);
newEnv.getRootElement().getFirstChild().createElementAsFirstChild(MbElement.TYPE_NAME, "MQDestinationList", null);
newEnv.getRootElement().getFirstChild().getFirstChild().createElementAsFirstChild(MbElement.TYPE_NAME, "DestinationData", null);
```

However, it is more efficient to store references as shown in the following example:

```
MbMessage newEnv = new MbMessage(inAssembly.getMessage());
MbElement destination = newEnv.getRootElement().createElementAsFirstChild(MbElement.TYPE_NAME, "Destination", null);
MbElement mqDestinationList = destination.createElementAsFirstChild(MbElement.TYPE_NAME, "MQDestinationList",
    null);
mqDestinationList.createElementAsFirstChild(MbElement.TYPE_NAME, "DestinationData", null);
```

Java string concatenation:

About this task

The + operator has a negative impact on performance because it involves creating a new String object for each concatenation; for example:

```
keyforCache = hostSystem + CommonFunctions.separator
+ sourceQueueValue + CommonFunctions.separator
+ smiKey + CommonFunctions.separator
+ newElement;
```

To improve performance, use the StringBuffer class and append method to concatenate java.lang.String objects, rather than the + operator, as shown in the following example:

```
StringBuffer keyforCacheBuf = new StringBuffer();
keyforCacheBuf.append(hostSystem);
keyforCacheBuf.append(CommonFunctions.separator);
keyforCacheBuf.append(sourceQueueValue);
keyforCacheBuf.append(CommonFunctions.separator);
keyforCacheBuf.append(smiKey);
keyforCacheBuf.append(CommonFunctions.separator);
keyforCacheBuf.append(newElement);
keyforCache = keyforCacheBuf.toString();
```

Java BLOB processing:

About this task

If you want to process a BLOB (by cutting it into chunks or by inserting characters, for example), it is efficient to use a JavaCompute node with its strong processing capabilities. If you are using a JavaCompute node, use ByteArrays and ByteArrayOutputStream to process the BLOB.

Java manual garbage collection:

About this task

Avoid manually invoking garbage collection, because it suspends the processing in the JVM, which significantly inhibits message throughput. In one example, a Java-based transformation message flow that was processing at the rate of 1300 messages per second without manual garbage collection was reduced to 100 messages per second when one manual garbage collection call was added to each message being processed.

Related tasks:

“Performance” on page 450

You can use message flow design and coding techniques to optimize the performance of your message flows. You can then analyze statistical information for your message flows and modify aspects of your broker configuration to tune the performance of your brokers and message flows.

“Tuning message flow performance” on page 510

You can configure your message flows to enhance their performance in terms of resource usage, message flow throughput, and response times.

“Analyzing message flow performance” on page 469

You can configure your broker to collect statistics about the operation and behavior of your message flows. Use this information to assess the performance of your message flows.



Developing message flows

Develop message flows to process your business messages and data. A message flow is a sequence of processing steps that run in an integration node when an input message is received.

“Using Activity logs” on page 623

Use Activity logs to get an overview of recent activities in your message flows and associated external resources.

“Analyzing resource performance” on page 529

You can collect statistics to assess the performance of resources used by integration servers.

Related reference:



JavaCompute node

Use the JavaCompute node to work with messages by using the Java language.



Resource statistics data: Java Virtual Machine (JVM)

Learn about the data that is returned for the JVM resource type when you activate resource statistics collection.



Activity logs

Activity logs contain information about message flows and how they interact with external resources. They can be generated for a message flow or a resource type. Learn about the structure and content of Activity logs, and about the *tags* that enrich the log data and can be used to filter its content.

XPath and XSLT code tips

You can improve message flow performance with XPath and XSLT by using some optimization techniques.

Before you begin

Before you start:

Read the following topics:

- “Performance” on page 450
- “Performance planning” on page 451
- “Tuning message flow performance” on page 510

About this task

You can improve the performance of XPath by explicitly specifying the number of instances that are required. For example:

- `/element[1]` finds the first occurrence of the element and then stops
- `/element[2]` finds the second occurrence of the element and then stops
- `/element` returns a node set of all instances in the message, which involves a full parse of the message

For more information about techniques that you can use to optimize your XML code, see Top 10 tips for using XPath.

XSLT has advantages in terms of its potential for code reuse, and caching is available for loaded stylesheets in IBM Integration Bus. However, the stylesheet requires a BLOB as input and produces a BLOB as output, and it is less efficient than ESQL and Java in terms of interaction with the message tree. When it is used mid-stream with other IBM Integration Bus technologies, it results in increased message parsing and serialization.

You can optimize the performance of your XSLT code by using a templates object (with different transformers for each transformation) to do multiple transformations with the same set of stylesheet instructions. You can also improve the efficiency of your stylesheets, by using the following techniques:

- Use `xmlns:key` elements and the `key()` function as an efficient way to retrieve node sets
- Where possible, use pattern matching rather than `xmlns:if` or `xmlns:when` statements
- Avoid the use of `xmlns:/*` (descendant axes) patterns near the root of a large document

Also consider the following points:

- When you are creating variables, `<xmlns:variable name="abcElem" select="abc"/>` is usually faster than `<xmlns:variable name="abcElem"><xmlns:value-of select="abc"/></xmlns:variable>`
- `xmlns:for-each` is fast because it does not require pattern matching
- `xmlns:sort` prevents incremental processing
- Use of index predicates within match patterns can be costly in terms of performance
- Decoding and encoding are costly in terms of performance
- If the XSLT is cached, the performance is improved because the XSLT is not parsed every time it is used; for more information, see XSLTransform node

Related tasks:

“Performance” on page 450

You can use message flow design and coding techniques to optimize the performance of your message flows. You can then analyze statistical information for your message flows and modify aspects of your broker configuration to tune the performance of your brokers and message flows.

“Tuning message flow performance” on page 510

You can configure your message flows to enhance their performance in terms of resource usage, message flow throughput, and response times.



Developing message flows

Develop message flows to process your business messages and data. A message flow is a sequence of processing steps that run in an integration node when an input message is received.

“Analyzing message flow performance” on page 469

You can configure your broker to collect statistics about the operation and behavior of your message flows. Use this information to assess the performance of your message flows.

“Performance planning” on page 451

When you design your IBM Integration Bus environment and the associated resources, the decisions that you make can affect the performance of your applications.

“Message flow design and performance” on page 453

Consider the performance implications arising from the design of your message flow.

Parsing and message flow performance

Understand the impact of parsing on message flow performance, and use techniques to limit the effect and improve performance.

Before you begin

Before you start:

Read the following topics:

- “Performance” on page 450
- “Performance planning” on page 451
- “Tuning message flow performance” on page 510

About this task

Parsing is the means of populating and serializing the message tree from the data, and it can occur whenever the message body is accessed. Multiple parsers are available, and the message complexity varies together with the cost in terms of performance. You can minimize the impact of parsing on message flow performance in the following ways:

Procedure

- Identify the message type as quickly as possible, and avoid using multiple parsers to find it.
- Use the most efficient parser available, such as XMLNSC for XML parsing, and DFDL for non-XML parsing. For more information about parsing, see Parsers and Available parsers.
- Use parser optimization techniques, such as parsing avoidance, partial parsing (parsing on demand), and opaque parsing:

- If possible, avoid the use of parsing completely, or consider sending only changed data. If you promote or copy key data structures to MQMD, MQRFH2, or JMS properties, you might be able to avoid the need to parse the user data; this technique can be particularly useful for message routing.
- Partial parsing (known as parsing on demand) involves parsing a message only when it is necessary to resolve the reference to a particular part of its content. For more information, see Parsing on demand.
- Opaque parsing reduces the size of the message tree, which results in improved performance. You can configure opaque parsing on the **Parser Options** tab of input nodes. If you enable validation, you cannot use opaque parsing.

Related concepts:

 **Parsers**

A parser is a program that interprets the physical bit stream of an incoming message, and creates an internal logical representation of the message in a tree structure. The parser also regenerates a bit stream for an outgoing message from the internal message tree representation.

 **Which body parser should you use?**

The characteristics of the messages that your applications exchange indicate which body parser you must use.


Related tasks:

“Performance” on page 450

You can use message flow design and coding techniques to optimize the performance of your message flows. You can then analyze statistical information for your message flows and modify aspects of your broker configuration to tune the performance of your brokers and message flows.

“Tuning message flow performance” on page 510

You can configure your message flows to enhance their performance in terms of resource usage, message flow throughput, and response times.


 **Developing message flows**

Develop message flows to process your business messages and data. A message flow is a sequence of processing steps that run in an integration node when an input message is received.

Related reference:

 **Available parsers**

A parser is called by the broker only when that parser is required. The parser that is called depends upon the parser that has been specified.

 **Parsing on demand**

On-demand parsing, referred to as partial parsing, is used to parse an input message bit stream only as far as is necessary to satisfy the current reference. The parsers that can perform partial parsing of input messages are the DFDL, JSON, MRM, XML, XMLNS, and XMLNSC parsers.

Message tree navigation and copying

Message tree navigation and message tree copying can reduce message flow performance, so it is important to use them appropriately and limit their usage where possible.

Before you begin

Before you start:

Read the following topics:

- “Performance” on page 450
- “Performance planning” on page 451
- “Tuning message flow performance” on page 510

About this task

Long paths are inefficient, so ensure that you minimize their usage, particularly in loops. Using reference variables and pointers in ESQL and Java can improve performance. Where possible, build a smaller message tree by using compact parsers such as XMLNSC, DFDL, MRM XML, RFH2C, and use opaque parsing.

Message tree copying is also costly in terms of resources, because it causes the logical tree to be duplicated in memory. You can improve performance by reducing the number of times that the message tree is copied, by using the following techniques:

- Reduce the number of Compute and JavaCompute nodes in a message flow
- If possible, set the Compute Mode property on the node to not include the message
- Copy at an appropriate level in the message tree; for example, copy once rather than for multiple branch nodes
- Copy data to the environment; however, remember that changes are not backed out

In addition to these techniques, the effect of copying the message tree is reduced if you also reduce the size of the message tree by using compact parsers and opaque parsing. For more information, see “Parsing and message flow performance” on page 466.

Related tasks:

“Performance” on page 450

You can use message flow design and coding techniques to optimize the performance of your message flows. You can then analyze statistical information for your message flows and modify aspects of your broker configuration to tune the performance of your brokers and message flows.

“Tuning message flow performance” on page 510

You can configure your message flows to enhance their performance in terms of resource usage, message flow throughput, and response times.



Developing message flows

Develop message flows to process your business messages and data. A message flow is a sequence of processing steps that run in an integration node when an input message is received.

Related reference:



Compute node

Use the Compute node to construct one or more new output messages.



JavaCompute node

Use the JavaCompute node to work with messages by using the Java language.

Analyzing message flow performance

You can configure your broker to collect statistics about the operation and behavior of your message flows. Use this information to assess the performance of your message flows.

Before you begin

Before you start:

Read the concept topic “Message flow statistics and accounting data” on page 471

About this task

Measures of performance typically include product speed in terms of processing rate and response times, and resource usage in terms of the CPU and memory consumed. In order to assess the performance of an application, metrics are used to compare the actual performance with the required or expected performance, using measures such as the number of messages per second, elapsed time, CPU utilization, or CPU cost per message. You can use the message flow statistics and accounting data to assess the performance of your message flows and applications using some of these metrics, and to identify the message flows and nodes that might benefit from tuning to improve performance.

You can start and stop the collection of statistics data by using the web user interface, the IBM Integration Explorer, or the command line. You can also view the collected statistics in tabular or graphical form by using the web user interface or the IBM Integration Explorer.

The following topics describe the tasks that you can complete to control the collection and display of message flow accounting and statistics data by using the web user interface:

- “Starting collection of accounting and statistics data in the web user interface” on page 487
- “Viewing accounting and statistics data in the web user interface” on page 499
- “Stopping collection of accounting and statistics data in the web user interface” on page 495

The following topics describe the tasks that you can complete to control the collection and display of message flow accounting and statistics data by using the IBM Integration Explorer:

- “Starting accounting and statistics data collection in the IBM Integration Explorer” on page 485
- “Viewing accounting and statistics data in the IBM Integration Explorer” on page 497
- “Filtering message flow accounting and statistics data in the IBM Integration Explorer” on page 508
- “Stopping collection of accounting and statistics data in the IBM Integration Explorer” on page 494

The following topics describe the tasks that you can complete to control the collection of message flow accounting and statistics data by using the command line:

- “Starting collection of message flow accounting and statistics data with the **mqsichangeflowstats** command” on page 482
- “Stopping collection of message flow accounting and statistics data with the **mqsichangeflowstats** command” on page 492
- “Viewing message flow accounting and statistics data collection parameters” on page 504
- “Modifying message flow accounting and statistics data collection parameters” on page 505
- “Resetting message flow accounting and statistics archive data” on page 507

These topics show examples of how to issue the accounting and statistics commands. The examples for z/OS are shown for SDSF; if you are using another interface, you must modify the example shown according to the requirements of that interface. For details of other z/OS options, see Issuing commands to the z/OS console.

For information about how to improve the performance of your message flows, see “Tuning message flow performance” on page 510.

For troubleshooting information that includes guidance on dealing with performance issues, see “Troubleshooting performance problems” on page 560.

The following topics contain reference information that you might find helpful when analyzing the performance of your message flows:

- Message flow accounting and statistics details
- Message flow accounting and statistics output formats
- Example message flow accounting and statistics data
- Metrics for accounting and statistics data in the IBM Integration Explorer

Related concepts:

“Message flow statistics and accounting data” on page 471

Message flow statistics and accounting data can be collected to record performance and operating details of one or more message flows.



Message flows overview

A message flow is a sequence of processing steps that run in the broker when an input message is received.

Related tasks:



Developing integration solutions

IBM Integration Bus provides a flexible environment in which you can develop integration solutions to transform, enrich, route, and process your business messages and data. You can integrate client applications that use different protocols and message formats.

“Viewing message flow accounting and statistics data” on page 496

You can view snapshot accounting and statistics data as it is produced by the broker.

“Troubleshooting performance problems” on page 560

Follow this guidance to resolve common problems with IBM Integration Bus performance.

“Performance” on page 450

You can use message flow design and coding techniques to optimize the performance of your message flows. You can then analyze statistical information for your message flows and modify aspects of your broker configuration to tune

the performance of your brokers and message flows.

“Tuning message flow performance” on page 510

You can configure your message flows to enhance their performance in terms of resource usage, message flow throughput, and response times.

“Subscribing to statistics reports” on page 542

You can subscribe to topics that return statistics about the operation of your message flows and resource managers.

Related reference:

 **mqsichangeflowstats** command

Use the **mqsichangeflowstats** command to control the accumulation of statistics about message flow operation.

 **mqsireportflowstats** command

Use the **mqsireportflowstats** command to display the current options for accounting and statistics that have been set using the **mqsichangeflowstats** command.

 Issuing commands to the z/OS console

You operate the broker using the z/OS START, STOP, and MODIFY commands.

Message flow statistics and accounting data

Message flow statistics and accounting data can be collected to record performance and operating details of one or more message flows.

Message flow statistics and accounting data captures dynamic information about the runtime behavior of a message flow. For example, it indicates how many messages are processed and how large those messages are, as well as processor usage and elapsed processing times. The statistical data is collected and recorded in a specified location when an event occurs, such as when a snapshot interval expires or when the integration server that you are collecting information about stops.

You can use the statistics generated for the following purposes:

- You can use snapshot data to assess the execution of a message flow to determine why it, or a node within it, is not performing as you expect.
- You can determine the route that messages are taking through a message flow. For example, you might find that an error path is taken more frequently than you expect and you can use the statistics to understand when the messages are routed to this error path.

Check the information provided by snapshot data for routing information; if this is insufficient for your needs, use archive data.

- You can record the load that applications, trading partners, or other users put on the broker. This allows you to record the relative use that different users make of the broker, and perhaps to charge them accordingly. For example, you could levy a nominal charge on every message that is processed by a broker, or by a specific message flow.

You can use archive data to carry out an assessment of this kind.

The broker takes information about statistics and accounting from the operating system. On some operating systems, such as Windows, Linux, and UNIX, rounding can occur because the system calls that are used to determine the processor times are not sufficiently granular. This rounding might affect the accuracy of the data.

The following restrictions apply to data collection:

- Data relating to the size of messages is not collected for WebSphere Adapters nodes (for example, the SAPIInput node), the FileInput node, the JMSInput node, or any user-defined input node that does not create a message tree from a bit stream.

Collecting message flow accounting and statistics data is optional; by default it is switched off. To use this facility, request it on a message flow or integration server basis. The settings for accounting and statistics data collection are reset to the defaults when an integration server is redeployed. Previous settings for message flows in an integration server are not passed on to the new message flows deployed to that integration server.

You can start and stop data collection by using the `mqsichangeflowstats` command, the web user interface, or the IBM Integration Explorer; you do not need to modify the broker or the message flow, or redeploy the message flow, to request statistics collection.

You can activate data collection on both your production and test systems. If you collect the default level of statistics (message flow), the effect on broker performance is minimal. However, collecting more data than the default message flow statistics can generate high volumes of report data that might affect performance slightly.

When you plan data collection, consider the following points:

- Collection options
- Accounting origin
- Output formats

The following topics contain reference information that you might find helpful when analyzing and tuning the performance of your message flows:

- Message flow accounting and statistics details
- Message flow accounting and statistics output formats
- Example message flow accounting and statistics data
- Metrics for accounting and statistics data in the IBM Integration Explorer

You can find more information about how to use the accounting and statistics facility to monitor the performance of a message flow in this developerWorks article on WebSphere Message Broker Explorer accounting and statistics.

You can also find information about how to use accounting and statistics data in this developerWorks article on message flow performance.

Related concepts:



Message flows overview

A message flow is a sequence of processing steps that run in the broker when an input message is received.

Related tasks:

“Analyzing message flow performance” on page 469

You can configure your broker to collect statistics about the operation and behavior of your message flows. Use this information to assess the performance of your message flows.

“Tuning message flow performance” on page 510

You can configure your message flows to enhance their performance in terms of

resource usage, message flow throughput, and response times.

Developing integration solutions

IBM Integration Bus provides a flexible environment in which you can develop integration solutions to transform, enrich, route, and process your business messages and data. You can integrate client applications that use different protocols and message formats.

“Performance” on page 450

You can use message flow design and coding techniques to optimize the performance of your message flows. You can then analyze statistical information for your message flows and modify aspects of your broker configuration to tune the performance of your brokers and message flows.

Related reference:

Message flow statistics and accounting data

You can collect message flow statistics and accounting data in various output formats.

mqsichangeflowstats command

Use the **mqsichangeflowstats** command to control the accumulation of statistics about message flow operation.

mqsireportflowstats command

Use the **mqsireportflowstats** command to display the current options for accounting and statistics that have been set using the **mqsichangeflowstats** command.


Message flow accounting and statistics collection options:

The options that you specify for message flow accounting and statistics collection determine what information is collected. You can request two types of data collection, depending on the user interface that you are using. You can use the IBM Integration Explorer or the web user interface to collect and view snapshot data, or you can use the **mqsichangeflowstats** command to collect snapshot and archive data.

- Snapshot data is collected for an interval of approximately 20 seconds. The exact length of the interval depends on system loading and the level of current broker activity. You cannot modify the length of time for which snapshot data is collected. At the end of this interval, the recorded statistics are written to the output destination and the interval is restarted.
- Archive data is collected for an interval that you have set for the broker on the **mqsicreatebroker** or **mqsichangebroker** command. You can specify an interval in the range 1 through 43200 minutes; the default value is 60 minutes. At the end of this interval, the recorded statistics are written to the output destination and the interval is restarted.

An interval ends and restarts when any of the following events occur, to preserve the integrity of the data that has been collected prior to the event:

- The message flow is redeployed
- The set of statistics data to be collected is modified
- The broker is shut down

 On z/OS, you can set the command parameter to 0, which means that the interval is controlled by an external timer mechanism. This support is provided by the Event Notification Facility (ENF), which you can use instead of the broker command parameter if you want to coordinate the expiration of this timer with other system events.

You can request snapshot data collection by using the web user interface or the IBM Integration Explorer. Alternatively, by using the **mqsichangeflowstats** command, you can specify snapshot data collection, archive data collection, or both. You can activate snapshot data collection while archive data collection is active. The data recorded in both reports is the same, but is collected for different intervals. If you activate both snapshot and archive data collection, be careful not to combine information from the two different reports, because you might count information twice.

You can view statistics data as it is generated, by using the **Broker Statistics** view in the IBM Integration Explorer or by selecting the **Statistics** tab in the web user interface.

Related concepts:



Message flows overview

A message flow is a sequence of processing steps that run in the broker when an input message is received.

Related tasks:

“Analyzing message flow performance” on page 469

You can configure your broker to collect statistics about the operation and behavior of your message flows. Use this information to assess the performance of your message flows.

“Starting accounting and statistics data collection in the IBM Integration Explorer” on page 485

Use the IBM Integration Explorer to start collecting snapshot accounting and statistics data for your integration nodes (brokers), integration servers, and message flows. You can then view the accounting and statistics data in the Broker Statistics and Broker Statistics Graph views.

Related reference:



Message flow statistics and accounting data

You can collect message flow statistics and accounting data in various output formats.



mqsichangebroker command

Use the **mqsichangebroker** command to change one or more of the configuration parameters of the broker.



mqsichangeflowstats command

Use the **mqsichangeflowstats** command to control the accumulation of statistics about message flow operation.



mqsicreatebroker command

Use the **mqsicreatebroker** command to create a broker and its associated resources.

Message flow accounting and statistics accounting origin:

Accounting and statistics data can be accumulated and reported with reference to an identifier associated with a message in a message flow. This identifier is the accounting origin, which provides a method of producing individual accounting and statistics data for multiple accounting origins that generate input to message flows. The accounting origin can be a fixed value, or it can be dynamically set according to your criteria.

For example, if your broker hosts a set of message flows associated with a particular client in a single integration server, you can set a specific value for the accounting origin for all these flows. You can then analyze the output provided to assess the use that the client or department makes of the broker, and charge them accordingly.

If you want to track the behavior of a particular message flow, you can set a unique accounting origin for this message flow, and analyze its activity over a specified period.

To make use of the accounting origin, you must perform the following tasks:

- Activate data collection, specifying the correct parameter to request basic support (the default is none, or no support). For details, see **mqsichangeflowstats** command.
- Configure each message flow for which you want a specific origin to include ESQL statements that set the unique value that is to be associated with the data collected. Data for message flows for which a specific value has not been set are identified with the value *Anonymous*.

The ESQL statements can be coded in a Compute, Database, or Filter node.

You can configure the message flow either to set a fixed value, or to determine a dynamic value, and can therefore create a very flexible method of recording sets of data that are specific to particular messages or circumstances. For more information, refer to “Setting message flow accounting and statistics accounting origin” on page 489.

You can complete these tasks in either order; if you configure the message flow before starting data collection, the broker ignores the setting. If you start data collection, specifying accounting origin support, before configuring the message flow, all data is collected with the Accounting Origin set to *Anonymous*. The broker acknowledges the origin when you redeploy the message flow. You can also modify data collection that has already started to request accounting origin support from the time that you issue the command. In both cases, data that has already been collected is written out and collection is restarted.

When data has been collected, you can review information for one or more specific origins. For example, if you select XML publication messages as your output format, you can start an application that subscribes to the origin in which you are interested.

Related concepts:



Message flows overview

A message flow is a sequence of processing steps that run in the broker when an input message is received.

Related tasks:

“Analyzing message flow performance” on page 469

You can configure your broker to collect statistics about the operation and behavior of your message flows. Use this information to assess the performance of your message flows.

“Setting message flow accounting and statistics accounting origin” on page 489

When you request accounting origin support for collecting message flow accounting and statistics data on the **mqsichangeflowstats** command, you must also configure your message flows to provide the correct identification values that indicate what the data is associated with.

Developing integration solutions

IBM Integration Bus provides a flexible environment in which you can develop integration solutions to transform, enrich, route, and process your business messages and data. You can integrate client applications that use different protocols and message formats.

Writing ESQL

How you can use ESQL to customize nodes.

Related reference:

Message flow statistics and accounting data

You can collect message flow statistics and accounting data in various output formats.

`mqsichangeflowstats` command

Use the `mqsichangeflowstats` command to control the accumulation of statistics about message flow operation.

Output formats for message flow accounting and statistics data:

When you collect message flow statistics, you can choose the output destination for the data.

You can select one or more of the following destinations, by using the `mqsichangeflowstats` command:

- User trace
- XML publication
- SMF
- “JSON publication” on page 477

If no format is specified, accounting and statistics data is sent to the user trace log by default. For more information about specifying an output format, see `mqsichangeflowstats` command.

If you start the collection of message flow statistics data by using the IBM Integration Explorer, the statistics are emitted in XML format in addition to any other formats that are being emitted. If you start the collection of message flow statistics data by using the web user interface, the statistics are emitted in JSON format in addition to any other formats that are already being emitted. As a result, if statistics collection is started through both the IBM Integration Explorer and the web user interface, data is emitted in both XML and JSON formats. If the output format was previously not specified and therefore defaulted to the user trace, the newly specified format replaces the default, and the data is no longer emitted to the user trace. However, if user trace has been explicitly specified, any additional formats that are selected subsequently are emitted in addition to the user trace.

If you use the `mqsichangeflowstats` command to explicitly specify the required output formats, the formats specified by the command replace the formats that are currently being emitted for the message flow (they are not added to them).

If you stop statistics collection from either the web user interface or the IBM Integration Explorer, all output formats are turned off. If statistics collection is subsequently restarted by using the `mqsichangeflowstats` command, the output format is reset to the default value of user trace, unless other formats are specified on the command. However, if statistics collection is restarted by using the

IBM Integration Explorer, the data is collected in XML format; if statistics collection is restarted through the web user interface, data is collected in JSON format.

Statistics data is written to the specified output location in the following circumstances:

- When the archive data interval expires.
- When the snapshot interval expires.
- When the broker shuts down. Any data that has been collected by the broker, but has not yet been written to the specified output destination, is written during shutdown. It might therefore represent data for an incomplete interval.
- When any part of the broker configuration is redeployed. Redeployed configuration data might contain an updated configuration that is not consistent with the existing record structure (for example, a message flow might include an additional node, or an integration server might include a new message flow). Therefore the current data, which might represent an incomplete interval, is written to the output destination. Data collection continues for the redeployed configuration until you change data collection parameters or stop data collection.
- When data collection parameters are modified. If you update the parameters that you have set for data collection, all data that is collected for the message flow (or message flows) is written to the output destination to retain data integrity. Statistics collection is restarted according to the new parameters.
- When an error occurs that terminates data collection. You must restart data collection yourself in this case.

User trace

You can specify that the data that is collected is written to the user trace log. The data is written even when trace is switched off.

If no output destination is specified for accounting and statistics, the default is the user trace log. If one or more output formats are subsequently specified, the specified formats replace the default, and the data is no longer emitted to the user trace. However, if user trace has been explicitly specified, any additional formats that are selected subsequently are emitted in addition to the user trace.

The data is written to one of the following locations:

Windows Windows

If you set the work path by using the **-w** parameter of the **mqsicreatebroker** command, the location is `workpath\Common\log`.

If you have not specified the broker work path, the location is:

- On Windows: `C:\ProgramData\IBM\MQSI\Common\log`.

Linux UNIX Linux and UNIX

`/var/mqsi/common/log`

z/OS z/OS

`/component_filesystem/log`

JSON publication

You can specify that the data that is collected is published in JSON format, which is available for viewing in the web user interface. If statistics collection is started

through the web user interface, statistics data is emitted in JSON format in addition to any other formats that are already being emitted.

The topic on which the data is published has the following structure:

```
$/SYS/Broker/brokerName/Statistics/JSON/SnapShot/integrationServerName/applications/applicationName/libraries/libraryName/messageflows/messageFlowName
```

The variables correspond to the following values:

brokerName

The name of the broker for which statistics are collected

integrationServerName

The name of the integration server for which statistics are collected

applicationName

The name of the application for which statistics are collected

libraryName

The name of the library for which statistics are collected

messageFlowName

The name of the message flow for which statistics are collected

XML publication

You can specify that the data that is collected is published in XML format and is available to subscribers registered in the broker network that subscribe to the correct topic. If statistics collection is started through the IBM Integration Explorer, statistics data is emitted in XML format in addition to any other formats that are already being emitted.

The topic on which the data is published has the following structure:

```
$/SYS/Broker/brokerName/StatisticsAccounting/recordType/integrationServerLabel/messageFlowLabel
```

The variables correspond to the following values:

brokerName

The name of the broker for which statistics are collected.

recordType

Set to SnapShot or Archive, depending on the type of data to which you are subscribing. Alternatively, use # to register for both snapshot and archive data if it is being produced. This value is case sensitive and must be entered as SnapShot.

integrationServerLabel

The name of the integration server for which statistics are collected.

messageFlowLabel

The label on the message flow for which statistics are collected.

Subscribers can include filter expressions to limit the publications that they receive. For example, they can choose to see only snapshot data, or to see data that is collected for a single broker. Subscribers can specify wild cards (+ and #) to receive publications that refer to multiple resources.

The following examples show the topic with which a subscriber registers to receive different sorts of data:

- Register the following topic for the subscriber to receive data for all message flows running on *BrokerA*:
\$SYS/Broker/*BrokerA*/StatisticsAccounting/#
- Register the following topic for the subscriber to receive only archive statistics that relate to a message flow *Flow1* running on integration server *Execution* on broker *BrokerA*:
\$SYS/Broker/*BrokerA*/StatisticsAccounting/Archive/*Execution*/*Flow1*
- Register the following topic for the subscriber to receive both snapshot and archive data for message flow *Flow1* running on integration server *Execution* on broker *BrokerA*:
\$SYS/Broker/*BrokerA*/StatisticsAccounting/+/Execution/*Flow1*

For help with registering your subscriber, see Message display, test and performance utilities SupportPac (IH03).

SMF

On z/OS, you can specify that the data collected is written to SMF. Accounting and statistics data uses SMF type 117 records. SMF supports the collection of data from multiple subsystems, and you might therefore be able to synchronize the information that is recorded from different sources.

To interpret the information that is recorded, use any utility program that processes SMF records.

Related concepts:

 Message flows overview

A message flow is a sequence of processing steps that run in the broker when an input message is received.

Related tasks:

 Developing integration solutions

IBM Integration Bus provides a flexible environment in which you can develop integration solutions to transform, enrich, route, and process your business messages and data. You can integrate client applications that use different protocols and message formats.

“Analyzing message flow performance” on page 469

You can configure your broker to collect statistics about the operation and behavior of your message flows. Use this information to assess the performance of your message flows.

“Viewing message flow accounting and statistics data” on page 496

You can view snapshot accounting and statistics data as it is produced by the broker.

Related reference:

 Message flow statistics and accounting data

You can collect message flow statistics and accounting data in various output formats.

 **mqsichangeflowstats** command

Use the **mqsichangeflowstats** command to control the accumulation of statistics about message flow operation.

mqsicreatebroker command

Use the **mqsicreatebroker** command to create a broker and its associated resources.

mqsireportflowstats command

Use the **mqsireportflowstats** command to display the current options for accounting and statistics that have been set using the **mqsichangeflowstats** command.

Message flow accounting and statistics details

You can collect message flow, thread, node, and terminal statistics for message flows.

Message flow accounting and statistics output formats

The message flow accounting and statistics data can be written in the following four formats: user trace, XML, JSON, and SMF.

Example message flow accounting and statistics data

You can view message flow accounting and statistic data in an XML publication or user trace entries. To view z/OS SMF records, use a utility program that processes SMF records.

Metrics for accounting and statistics data in the IBM Integration Explorer

You can filter the metrics displayed for accounting and statistics data in the Broker Statistics and Broker Statistics Graph views in the IBM Integration Explorer.

Starting collection of message flow accounting and statistics data

You can start collecting message flow accounting and statistics data for an active broker at any time. You can specify what type of statistics you want to collect, and where to send the data.

Before you begin

Before you start:

- Ensure that you have created a message flow; for more information, see [Creating a message flow](#).
- Ensure that you have deployed a broker archive (BAR) file; for more information, see [Deploying a broker archive file](#).
- Read the concept topic, “Message flow accounting and statistics collection options” on page 473.

About this task

You can start collecting message flow accounting and statistics data by using the IBM Integration Explorer, the web user interface, or the **mqsichangeflowstats** command.

If you start the collection of statistics data by using the IBM Integration Explorer, the statistics are emitted in XML format in addition to any other formats that are already being emitted. If you start the collection of statistics data by using the web user interface, the statistics are emitted in JSON format in addition to any other formats that are already being emitted. As a result, if statistics collection is started through both the IBM Integration Explorer and the web user interface, data is emitted in both XML and JSON formats, and can be viewed in both the IBM Integration Explorer and the web user interface.

If you use the **mqsichangeflowstats** command to start statistics collection and specify the required output formats, the specified formats replace the output formats that were previously being emitted for the message flow (they are not added to them). If you use the **mqsichangeflowstats** command to start the collection of statistics, without specifying an output format on the command, the statistics data is emitted to the user trace log by default.

For more information about starting statistics collection, see the following topics:

- “Starting collection of accounting and statistics data in the web user interface” on page 487
- “Starting accounting and statistics data collection in the IBM Integration Explorer” on page 485
- “Starting collection of message flow accounting and statistics data with the **mqsichangeflowstats** command” on page 482

What to do next

Next:

You can now complete the following tasks:

- “Setting message flow accounting and statistics accounting origin” on page 489
- “Viewing message flow accounting and statistics data” on page 496
- “Viewing and modifying data collection settings” on page 503
- “Stopping collection of message flow accounting and statistics data” on page 491

Related concepts:

“Message flow statistics and accounting data” on page 471

Message flow statistics and accounting data can be collected to record performance and operating details of one or more message flows.



Message flows overview

A message flow is a sequence of processing steps that run in the broker when an input message is received.

“Output formats for message flow accounting and statistics data” on page 476

When you collect message flow statistics, you can choose the output destination for the data.

Related tasks:

“Stopping collection of message flow accounting and statistics data with the **mqsichangeflowstats** command” on page 492

You can use the **mqsichangeflowstats** command stop collecting data for message flow accounting and statistics at any time. You do not have to stop the message flow, integration server, or broker to make this change, nor do you have to redeploy the message flow.

“Viewing message flow accounting and statistics data collection parameters” on page 504

You can view the parameters that are currently in effect for message flow accounting and statistics data collection by using the **mqsireportflowstats** command.

“Modifying message flow accounting and statistics data collection parameters” on page 505

You can modify the parameters that you have set for message flow accounting and statistics data collection. For example, you can start collecting data for a new message flow that you have deployed to an integration server for which you are already collecting data. You can modify parameters while data collection is active;

you do not have to stop data collection and restart it.



Developing integration solutions

IBM Integration Bus provides a flexible environment in which you can develop integration solutions to transform, enrich, route, and process your business messages and data. You can integrate client applications that use different protocols and message formats.

Related reference:



mqsichangeflowstats command

Use the **mqsichangeflowstats** command to control the accumulation of statistics about message flow operation.



mqsireportflowstats command

Use the **mqsireportflowstats** command to display the current options for accounting and statistics that have been set using the **mqsichangeflowstats** command.



Message flow statistics and accounting data

You can collect message flow statistics and accounting data in various output formats.



Message flow accounting and statistics output formats

The message flow accounting and statistics data can be written in the following four formats: user trace, XML, JSON, and SMF.

Starting collection of message flow accounting and statistics data with the mqsichangeflowstats command:

You can use the **mqsichangeflowstats** command to start collecting message flow accounting and statistics data for an active broker at any time. You can specify what type of statistics you want to collect, and where to send the data.

Before you begin

Before you start:

- Ensure that you have created a message flow; for more information, see [Creating a message flow](#).
- Ensure that you have deployed a broker archive (BAR) file; for more information, see [Deploying a broker archive file](#).
- Read the concept topic, “Message flow accounting and statistics collection options” on page 473.

About this task

Select the granularity of the data that you want to be collected by specifying the appropriate parameters on the **mqsichangeflowstats** command. You must request statistics collection on a broker basis. If you want to collect information for more than one broker, you must issue the corresponding number of commands.

To start collecting message flow accounting and statistics data:

Procedure

1. Identify the broker for which you want to collect statistics.

2. Decide the resource for which you want to collect statistics. You can collect statistics for a specific integration server, or for all integration servers for the specified broker.
 - If you indicate a specific integration server, you can request that data is recorded for a specific message flow or all message flows in that group.
 - If you specify all integration servers, you must specify all message flows.
3. Decide if you want to collect thread-related statistics.
4. Decide if you want to collect node-related statistics. If you do, you can also collect information about terminals for the nodes.
5. Decide if you want to associate data collection with a particular accounting origin. This option is valid for snapshot and archive data, and for message flows and integration servers. However, when active, you must set its origin value in each message flow to which it refers. If you do not configure the participating message flows to set the appropriate origin identifier, the data collected for that message flow is collected with the origin set to *Anonymous*. For more information, see “Setting message flow accounting and statistics accounting origin” on page 489.
6. Decide the target destination. You can choose multiple destinations by specifying multiple comma-separated values for the output format property.
 - User trace log (the default setting). The output data can be processed using **mqsireadlog** and **mqsiformatlog**.
 - XML format publication message. If you chose this target destination, register the following topic for the subscriber:

`$/SYS/Broker/brokerName/StatisticsAccounting/recordType/integrationServerLabel/messageFlowLabel`

where *brokerName*, *integrationServerLabel*, and *messageFlowLabel* represent the broker, integration server, and message flow on which you want to receive data. *recordType* is the type of data collection on which you want to receive publications (snapshot, archive, or the number sign (#) to receive both snapshot and archive). The value that you specify for record type is case sensitive; therefore, if you choose to receive snapshot data, set the record type to *Snapshot*.

- JSON publication message. Register the following topic for the subscriber:
`$/SYS/Broker/brokerName/Statistics/JSON/SnapShot/integrationServerName/applications/applicationName/libraries/libraryName/messageFlows/messageFlowName`
 where *brokerName*, *integrationServerName*, *applicationName*, *libraryName*, and *messageFlowName* represent the broker, integration server, application, library, and message flow on which you want to receive data.
- **z/OS** SMF (on z/OS only)

7. Decide the type of data collection that you want:
 - Snapshot
 - Archive

You can collect snapshot and archive data at the same time, but you have to configure them separately.

8. Issue the **mqsichangeflowstats** command with the appropriate parameters to reflect the decisions that you have made.

For example, to turn on snapshot data for all message flows in the default integration server for BrokerA, and include node data with the basic message flow statistics, use the following command:

```
mqsichangeflowstats BrokerA -s -e default -j -c active -n basic
```

z/OS Using SDSF on z/OS, enter:

```
/F BrokerA,cs s=yes,e=default,j=yes,c=active,n=basic
```

For more examples, see **mqsichangeflowstats** command.

Results

When the command completes successfully, data collection for the specified resources is started:

- If you have requested snapshot data, information is collected for approximately 20 seconds, and the results are written to the specified output.
- If you have requested archive data, information is collected for the interval defined for the broker (on the **mqsicreatebroker** or **mqsichangebroker** command, or by the external timer facility ENF). The results are written to the specified output, the interval is reset, and data collection starts again.

What to do next

Next:

You can now complete the following tasks:

- “Setting message flow accounting and statistics accounting origin” on page 489
- “Stopping collection of message flow accounting and statistics data” on page 491
- “Viewing message flow accounting and statistics data” on page 496
- “Viewing and modifying data collection settings” on page 503

Related concepts:

“Message flow statistics and accounting data” on page 471

Message flow statistics and accounting data can be collected to record performance and operating details of one or more message flows.



Message flows overview

A message flow is a sequence of processing steps that run in the broker when an input message is received.

Related tasks:

“Setting message flow accounting and statistics accounting origin” on page 489

When you request accounting origin support for collecting message flow accounting and statistics data on the **mqsichangeflowstats** command, you must also configure your message flows to provide the correct identification values that indicate what the data is associated with.

“Stopping collection of message flow accounting and statistics data with the **mqsichangeflowstats** command” on page 492

You can use the **mqsichangeflowstats** command stop collecting data for message flow accounting and statistics at any time. You do not have to stop the message flow, integration server, or broker to make this change, nor do you have to redeploy the message flow.

“Viewing message flow accounting and statistics data collection parameters” on page 504

You can view the parameters that are currently in effect for message flow accounting and statistics data collection by using the **mqsireportflowstats** command.

“Modifying message flow accounting and statistics data collection parameters” on page 505

You can modify the parameters that you have set for message flow accounting and statistics data collection. For example, you can start collecting data for a new message flow that you have deployed to an integration server for which you are already collecting data. You can modify parameters while data collection is active; you do not have to stop data collection and restart it.

“Resetting message flow accounting and statistics archive data” on page 507
You can reset message flow accounting and statistics archive data to purge any accounting and statistics data not yet reported for that collecting interval. This removes unwanted data. You can request this at any time; you do not have to stop data collection and restart it to perform reset. You cannot reset snapshot data.



Developing integration solutions

IBM Integration Bus provides a flexible environment in which you can develop integration solutions to transform, enrich, route, and process your business messages and data. You can integrate client applications that use different protocols and message formats.

Related reference:



Issuing commands to the z/OS console

You operate the broker using the z/OS START, STOP, and MODIFY commands.



mqsichangeflowstats command

Use the **mqsichangeflowstats** command to control the accumulation of statistics about message flow operation.



mqsireportflowstats command

Use the **mqsireportflowstats** command to display the current options for accounting and statistics that have been set using the **mqsichangeflowstats** command.



Message flow statistics and accounting data

You can collect message flow statistics and accounting data in various output formats.

Starting accounting and statistics data collection in the IBM Integration Explorer:

Use the IBM Integration Explorer to start collecting snapshot accounting and statistics data for your integration nodes (brokers), integration servers, and message flows. You can then view the accounting and statistics data in the Broker Statistics and Broker Statistics Graph views.

Before you begin

Before you start:

- Ensure that you have created a message flow; for more information, see [Creating a message flow](#).
- Ensure that you have deployed a broker archive (BAR) file; for more information, see [Deploying a broker archive file](#).
- Read the concept topic, “Message flow statistics and accounting data” on page 471.

About this task

Use the snapshot accounting and statistics data displayed in the Broker Statistics and Broker Statistics Graph views to monitor the performance and resource usage of your integration nodes (brokers) or integration servers at the message flow, message flow node, or terminal level.

You can start collecting accounting and statistics data for an active integration node at any time, and you can start collecting accounting and statistics data for multiple integration nodes at the same time.

If you start the collection of statistics data by using the IBM Integration Explorer, the statistics are emitted in XML format in addition to any other formats that are already being emitted, and can be viewed in the IBM Integration Explorer.

To start collecting snapshot message flow accounting and statistics data by using the IBM Integration Explorer:

Procedure

1. In the WebSphere MQ Explorer - Navigator view, expand the Integration Nodes folder.
2. Right-click the integration server or message flow for which you want to collect statistics.
 - If you selected an integration server, click **Statistics All Flows > Start Statistics**.
 - If you selected a message flow click **Statistics > Start Statistics**.

A message is sent to the integration node to start collecting accounting and statistics data for the selected resource.

3. Click **Window > Show View > Broker Statistics** to open the Broker Statistics and Broker Statistics Graph view. These two views are displayed together. If you close one of the views, the other view is also closed.

Results

Accounting and statistics data for the selected integration node, integration server, or message flow is displayed in the Broker Statistics and Broker Statistics Graph views. It can take up to 30 seconds for accounting and statistics data to be received from the integration node and displayed in the Broker Statistics Graph view. The message **Waiting for Data** is displayed in the title bar is displayed until accounting and statistics data is received from the selected integration node, integration server, or message flow. When data is received from the integration node, it is displayed in numeric form in the Broker Statistics view, and a visual representation of this data is shown in the Broker Statistics Graph view. Select individual or multiple items in the Broker Statistics view to change the information displayed in the Broker Statistics Graph view.

What to do next

Next:

To examine the current data in the two views, click **Pause** in the Broker Statistics Graph view to prevent the current data from being overwritten with new data. Click **Play** to resume displaying more accounting and statistics data. You can change the visual representation of the data by clicking **Linear**, **Logarithmic**, and **Stacked**.

You can also select the metrics that are displayed in the graph view; for more information, see “Filtering message flow accounting and statistics data in the IBM Integration Explorer” on page 508.

Related concepts:



IBM Integration Explorer

The IBM Integration Explorer is a graphical user interface based on the Eclipse platform for administering your integration nodes (brokers).

“Message flow statistics and accounting data” on page 471

Message flow statistics and accounting data can be collected to record performance and operating details of one or more message flows.

Related tasks:

“Viewing accounting and statistics data in the IBM Integration Explorer” on page 497

You can use the Broker Statistics and Broker Statistics Graph views in the IBM Integration Explorer to view snapshot accounting and statistics data as it is produced by the integration node (broker).

“Stopping collection of accounting and statistics data in the IBM Integration Explorer” on page 494

Use the IBM Integration Explorer to stop collecting accounting and statistics data for your integration nodes (brokers), integration servers, and message flows.

“Filtering message flow accounting and statistics data in the IBM Integration Explorer” on page 508

You can select the metrics for the snapshot accounting and statistics data that are displayed in the Broker Statistics and Broker Statistics Graph views in the IBM Integration Explorer.

Related reference:



Metrics for accounting and statistics data in the IBM Integration Explorer

You can filter the metrics displayed for accounting and statistics data in the Broker Statistics and Broker Statistics Graph views in the IBM Integration Explorer.

Starting collection of accounting and statistics data in the web user interface:

You can use the web user interface to start collecting snapshot accounting and statistics data for your message flows. You can then view the accounting and statistics data in the web user interface.

Before you begin

Before you start:

- Ensure that you have created a message flow and deployed a broker archive (BAR) file; for more information, see [Creating a message flow and Deploying a broker archive file](#).
- Ensure that a web user interface server has been configured; for more information, see [“Configuring the web user interface server”](#) on page 128.
- Read the concept topic [“Message flow statistics and accounting data”](#) on page 471.

About this task

You can start collecting accounting and statistics data for one or more active message flows at any time.

If you start the collection of statistics data by using the web user interface, the statistics are emitted in JSON format in addition to any other formats that are already being emitted, and they can then be viewed in the web user interface.

To start collecting snapshot message flow accounting and statistics data by using the web user interface, follow these steps:

Procedure

1. Log on to the web user interface server, as described in “Accessing the web user interface” on page 133. The navigation tree is displayed on the left side of the pane, showing the servers (integration servers), message flows, and other resources owned by your broker.
2. Click the down arrow next to the required message flow in the navigation tree, and then click **Statistics on** in the context menu. You can start the collection of statistics data for all message flows in an integration node (broker), server (integration server), or application, by clicking the down arrow next to the required resource and then clicking **Statistics on** in the context menu. Statistics collection for the selected message flows is turned on, and an up-to-date snapshot of information is collected every twenty seconds.

What to do next

When statistics collection is turned on, you can view the snapshot data in the web user interface. For more information, see “Viewing accounting and statistics data in the web user interface” on page 499.

Related concepts:

“Message flow statistics and accounting data” on page 471

Message flow statistics and accounting data can be collected to record performance and operating details of one or more message flows.



IBM Integration Bus web user interface

The IBM Integration Bus web user interface enables web users to access broker resources through an HTTP client, and provides broker administrators with an alternative to the IBM Integration Explorer for administering broker resources.

Related tasks:

“Viewing accounting and statistics data in the web user interface” on page 499
You can use the web user interface to view snapshot accounting and statistics data.

“Stopping collection of accounting and statistics data in the web user interface” on page 495

You can use the web user interface to stop collecting accounting and statistics data for your message flows.

“Starting collection of message flow accounting and statistics data” on page 480

You can start collecting message flow accounting and statistics data for an active broker at any time. You can specify what type of statistics you want to collect, and where to send the data.

“Stopping collection of message flow accounting and statistics data” on page 491

You can stop collecting data for message flow accounting and statistics at any time. You do not have to stop the message flow, integration server, or broker to make this change, nor do you have to redeploy the message flow.

“Viewing message flow accounting and statistics data” on page 496

You can view snapshot accounting and statistics data as it is produced by the broker.

Setting message flow accounting and statistics accounting origin:

When you request accounting origin support for collecting message flow accounting and statistics data on the `mqsichangeflowstats` command, you must also configure your message flows to provide the correct identification values that indicate what the data is associated with.

Before you begin

Before you start:

- Ensure that you have created a message flow; for more information, see [Creating a message flow](#).
- Read the concept topic, “Message flow accounting and statistics accounting origin” on page 474

About this task

Accounting and statistics data is associated with an accounting origin. For more information, see “Message flow statistics and accounting data” on page 471 and “Message flow accounting and statistics accounting origin” on page 474.

You can set a different value for every message flow for which data collection is active, or the same value for a group of message flows (for example, those in a single integration server, or associated with a particular client, department, or application suite).

The accounting origin setting is not used until you deploy the message flow or flows to the brokers on which they are to run. You can activate data collection, or modify it to request accounting origin support, before or after you deploy the message flow. You do not have to stop collecting data when you deploy a message flow that changes accounting origin.

To configure a message flow to specify a particular accounting origin, complete the following steps.

Procedure

1. Open the message flow with which you want to work.
2. Click **Selection** above the palette of nodes.
3. Right-click a Compute, Database, or Filter node in the editor view, then click **Open ESQL**. The associated ESQL file is opened in the editor view, and your cursor is positioned at the start of the correct module. You can include the required ESQL in any of these nodes, so decide which node in each message flow is the most appropriate for this action.

To take advantage of the accounting origin support, include one of these nodes in each message flow for which you want a specific origin set. If you have not configured one of these three nodes in the message flow, you must add one at a suitable point (for example, immediately following the input node) and connect it to other nodes in the flow.

4. Update the ESQL in the node module to set an accounting origin. The broker uses the origin identifier that is set in the Environment tree. You must set a value in the field with correlation name `Environment.Broker.Accounting.Origin`. This field is not created automatically in the Environment tree when the message is first received in the broker. The field is created only when you set it in an ESQL module that is associated with a node in the message flow.

If you do not set a value in the message flow, the default value `Anonymous` is used for all output. If you set a value in more than one place in the message flow, the value that you set immediately before the message flow terminates is used in the output data.

The code that you must add has the following form:

```
SET Environment.Broker.Accounting.Origin = "value";
```

You can set the identifier to a fixed value (as shown previously), or you can determine its value based on a dynamic value that is known only at run time. The value must be character data, and can be a maximum of 32 bytes. For example, you might set its value to the contents of a particular field in the message that is being processed (if you are coding ESQL for a Compute node, you must use correlation name `InputBody` in place of `Body` in the following example):

```
IF Body.DepartmentName <> NULL THEN  
    SET Environment.Broker.Accounting.Origin = Body.DepartmentName;  
END IF;
```

5. Save the ESQL module, and check that you have not introduced any errors.
6. Save the message flow, and check again for errors.

Results

You are now ready to deploy the updated message flow; for more information, see [Deploying resources](#). Accounting and statistics data records that are collected after the message flow has been deployed includes the origin identifier that you have set.

Related concepts:



[Message flows overview](#)

A message flow is a sequence of processing steps that run in the broker when an input message is received.

["Message flow statistics and accounting data" on page 471](#)

Message flow statistics and accounting data can be collected to record performance and operating details of one or more message flows.

Related tasks:



[Developing integration solutions](#)

IBM Integration Bus provides a flexible environment in which you can develop integration solutions to transform, enrich, route, and process your business messages and data. You can integrate client applications that use different protocols and message formats.



[Packaging and deploying](#)

Package resources that you create in the IBM Integration Toolkit, such as message flows, and deploy them to integration servers on brokers.

["Modifying message flow accounting and statistics data collection parameters" on page 505](#)

You can modify the parameters that you have set for message flow accounting and statistics data collection. For example, you can start collecting data for a new message flow that you have deployed to an integration server for which you are already collecting data. You can modify parameters while data collection is active; you do not have to stop data collection and restart it.



[Writing ESQL](#)

How you can use ESQL to customize nodes.

Related reference:

mqsichangeflowstats command

Use the **mqsichangeflowstats** command to control the accumulation of statistics about message flow operation.

Compute node

Use the Compute node to construct one or more new output messages.

Database node

Use the Database node to interact with a database in the specified ODBC data source.

Filter node

Use the Filter node to route a message according to message content.

Message flow statistics and accounting data

You can collect message flow statistics and accounting data in various output formats.

Stopping collection of message flow accounting and statistics data

You can stop collecting data for message flow accounting and statistics at any time. You do not have to stop the message flow, integration server, or broker to make this change, nor do you have to redeploy the message flow.

Before you begin

Before you start:

- Read the concept topic “Message flow statistics and accounting data” on page 471

About this task

You can modify the parameters that are currently in force for collecting message flow accounting and statistics data without stopping data collection. See “Modifying message flow accounting and statistics data collection parameters” on page 505 for further details.

You can stop collecting message flow accounting and statistics data by using the IBM Integration Explorer, the web user interface, or the **mqsichangeflowstats** command. For information about each of these methods, see the following topics:

- “Stopping collection of accounting and statistics data in the web user interface” on page 495
- “Stopping collection of accounting and statistics data in the IBM Integration Explorer” on page 494
- “Stopping collection of message flow accounting and statistics data with the **mqsichangeflowstats** command” on page 492

Related concepts:

Message flows overview

A message flow is a sequence of processing steps that run in the broker when an input message is received.

“Message flow statistics and accounting data” on page 471

Message flow statistics and accounting data can be collected to record performance

and operating details of one or more message flows.

Related tasks:



Developing integration solutions

IBM Integration Bus provides a flexible environment in which you can develop integration solutions to transform, enrich, route, and process your business messages and data. You can integrate client applications that use different protocols and message formats.

“Starting collection of message flow accounting and statistics data with the **mqsichangeflowstats** command” on page 482

You can use the **mqsichangeflowstats** command to start collecting message flow accounting and statistics data for an active broker at any time. You can specify what type of statistics you want to collect, and where to send the data.

“Modifying message flow accounting and statistics data collection parameters” on page 505

You can modify the parameters that you have set for message flow accounting and statistics data collection. For example, you can start collecting data for a new message flow that you have deployed to an integration server for which you are already collecting data. You can modify parameters while data collection is active; you do not have to stop data collection and restart it.

Related reference:



Message flow statistics and accounting data

You can collect message flow statistics and accounting data in various output formats.



mqsichangeflowstats command

Use the **mqsichangeflowstats** command to control the accumulation of statistics about message flow operation.



mqsireportflowstats command

Use the **mqsireportflowstats** command to display the current options for accounting and statistics that have been set using the **mqsichangeflowstats** command.



Issuing commands to the z/OS console

You operate the broker using the z/OS START, STOP, and MODIFY commands.

Stopping collection of message flow accounting and statistics data with the mqsichangeflowstats command:

You can use the **mqsichangeflowstats** command stop collecting data for message flow accounting and statistics at any time. You do not have to stop the message flow, integration server, or broker to make this change, nor do you have to redeploy the message flow.

Before you begin

Before you start:

- Start collecting accounting and statistics data; for more information, see “Starting collection of message flow accounting and statistics data” on page 480
- Read the concept topic about “Message flow statistics and accounting data” on page 471

About this task

You can stop collecting data for message flow accounting and statistics at any time. You do not have to stop the message flow, integration server, or broker to make this change, nor do you have to redeploy the message flow.

You can modify the parameters that are currently in force for collecting message flow accounting and statistics data without stopping data collection. See “Modifying message flow accounting and statistics data collection parameters” on page 505 for further details.

To stop collecting data:

Procedure

1. Check the resources for which you want to stop collecting data.

You do not have to stop all active data collection. You can stop a subset of data collection. For example, if you started collecting statistics for all message flows in a particular integration server, you can stop doing so for a specific message flow in that integration server. Data collection for all other message flows in that integration server continues.

2. Issue the **mqsichangeflowstats** command with the appropriate parameters to stop collecting data for some or all resources.

For example, to switch off snapshot data for all message flows in all integration servers for BrokerA, enter:

```
mqsichangeflowstats BrokerA -s -g -j -c inactive
```

 Using SDSF on z/OS, enter:

```
/F BrokerA,cs s=yes g=yes j=yes c=inactive
```

Refer to the **mqsichangeflowstats** command for further examples.

Results

When the command completes successfully, data collection for the specified resources is stopped. Any outstanding data that has been collected is written to the output destination when you issue this command, to ensure the integrity of data collection.

Related concepts:

 Message flows overview

A message flow is a sequence of processing steps that run in the broker when an input message is received.

Related tasks:

 Developing integration solutions

IBM Integration Bus provides a flexible environment in which you can develop integration solutions to transform, enrich, route, and process your business messages and data. You can integrate client applications that use different protocols and message formats.

“Starting collection of message flow accounting and statistics data” on page 480

You can start collecting message flow accounting and statistics data for an active broker at any time. You can specify what type of statistics you want to collect, and where to send the data.

“Viewing and modifying data collection settings” on page 503

You can view and modify the settings that control the collection of message flow

accounting and statistics data.

Related reference:



Message flow statistics and accounting data

You can collect message flow statistics and accounting data in various output formats.



mqsichangeflowstats command

Use the **mqsichangeflowstats** command to control the accumulation of statistics about message flow operation.



mqsireportflowstats command

Use the **mqsireportflowstats** command to display the current options for accounting and statistics that have been set using the **mqsichangeflowstats** command.



Issuing commands to the z/OS console

You operate the broker using the z/OS START, STOP, and MODIFY commands.

Stopping collection of accounting and statistics data in the IBM Integration Explorer:

Use the IBM Integration Explorer to stop collecting accounting and statistics data for your integration nodes (brokers), integration servers, and message flows.

About this task

You can stop collecting data for message flow accounting and statistics at any time. You do not have to stop the message flow, integration server, or integration node to make this change, nor do you have to redeploy the message flow. You do not have to stop all active data collection. You can stop a subset of data collection. For example, if you started collecting statistics for all message flows in a particular integration server, you can stop doing so for a specific message flow in that integration server. Data collection for all other message flows in that integration server continues.

To stop collecting message flow accounting and statistics data:

Procedure

1. In the WebSphere MQ Explorer - Navigator view, expand the Integration Nodes folder.
2. Right-click the integration server or message flow for which you want to stop collecting statistics.
 - If you selected an integration server, click **Statistics All Flows > Stop Statistics**.
 - If you selected a message flow click **Statistics > Stop Statistics**.

Results

A message is sent to the integration node to stop collecting accounting and statistics data for the selected resource.

Related concepts:



IBM Integration Explorer

The IBM Integration Explorer is a graphical user interface based on the Eclipse

platform for administering your integration nodes (brokers).

“Message flow statistics and accounting data” on page 471

Message flow statistics and accounting data can be collected to record performance and operating details of one or more message flows.

Related tasks:

“Viewing accounting and statistics data in the IBM Integration Explorer” on page 497

You can use the Broker Statistics and Broker Statistics Graph views in the IBM Integration Explorer to view snapshot accounting and statistics data as it is produced by the integration node (broker).

“Starting accounting and statistics data collection in the IBM Integration Explorer” on page 485

Use the IBM Integration Explorer to start collecting snapshot accounting and statistics data for your integration nodes (brokers), integration servers, and message flows. You can then view the accounting and statistics data in the Broker Statistics and Broker Statistics Graph views.

“Filtering message flow accounting and statistics data in the IBM Integration Explorer” on page 508

You can select the metrics for the snapshot accounting and statistics data that are displayed in the Broker Statistics and Broker Statistics Graph views in the IBM Integration Explorer.

Related reference:



Metrics for accounting and statistics data in the IBM Integration Explorer

You can filter the metrics displayed for accounting and statistics data in the Broker Statistics and Broker Statistics Graph views in the IBM Integration Explorer.

Stopping collection of accounting and statistics data in the web user interface:

You can use the web user interface to stop collecting accounting and statistics data for your message flows.

About this task

You can stop collecting message flow accounting and statistics data for one or more flows by clicking the down arrow next to the required flows in the navigation tree and then clicking **Statistics off**. Data collection stops for only the selected flows, and continues for all other active flows in the broker that have statistics collection turned on.

You can stop the data collection at any time; you do not have to stop the message flow, server (integration server), or integration node (broker) to make this change, and you do not have to redeploy the flow. When statistics collection is stopped for a message flow, the message flow is not stopped, but continues to run uninterrupted.

If you stop statistics collection from either the web user interface or the IBM Integration Explorer, all output formats are turned off. If statistics collection is subsequently restarted by using the **mqsichangeflowstats** command, the output format is reset to the default value of user trace, unless other formats are specified on the command. However, if statistics collection is restarted by using the IBM Integration Explorer, the data is collected in XML format; if statistics collection is restarted through the web user interface, data is collected in JSON format.

Results

A message is sent to the broker to stop collecting accounting and statistics data for the selected flow.

Related concepts:

“Message flow statistics and accounting data” on page 471

Message flow statistics and accounting data can be collected to record performance and operating details of one or more message flows.

Related tasks:

“Starting collection of accounting and statistics data in the web user interface” on page 487

You can use the web user interface to start collecting snapshot accounting and statistics data for your message flows. You can then view the accounting and statistics data in the web user interface.

“Viewing accounting and statistics data in the web user interface” on page 499

You can use the web user interface to view snapshot accounting and statistics data.

“Starting collection of message flow accounting and statistics data” on page 480

You can start collecting message flow accounting and statistics data for an active broker at any time. You can specify what type of statistics you want to collect, and where to send the data.

“Stopping collection of message flow accounting and statistics data” on page 491

You can stop collecting data for message flow accounting and statistics at any time. You do not have to stop the message flow, integration server, or broker to make this change, nor do you have to redeploy the message flow.

“Viewing message flow accounting and statistics data”

You can view snapshot accounting and statistics data as it is produced by the broker.

Viewing message flow accounting and statistics data

You can view snapshot accounting and statistics data as it is produced by the broker.

Before you begin

Before you start:

- Start collecting accounting and statistics data. For more information, see “Starting collection of message flow accounting and statistics data” on page 480.
- Read the concept topic “Message flow statistics and accounting data” on page 471.

About this task

You can view message flow accounting and statistics data using either the IBM Integration Explorer or the web user interface. For information about using either of these methods, see the following topics:

- “Viewing accounting and statistics data in the IBM Integration Explorer” on page 497
- “Viewing accounting and statistics data in the web user interface” on page 499

Related concepts:

“Message flow statistics and accounting data” on page 471

Message flow statistics and accounting data can be collected to record performance and operating details of one or more message flows.

Related tasks:

“Starting collection of message flow accounting and statistics data” on page 480
You can start collecting message flow accounting and statistics data for an active broker at any time. You can specify what type of statistics you want to collect, and where to send the data.

“Filtering message flow accounting and statistics data in the IBM Integration Explorer” on page 508

You can select the metrics for the snapshot accounting and statistics data that are displayed in the Broker Statistics and Broker Statistics Graph views in the IBM Integration Explorer.

“Viewing and modifying data collection settings” on page 503

You can view and modify the settings that control the collection of message flow accounting and statistics data.

“Stopping collection of message flow accounting and statistics data” on page 491
You can stop collecting data for message flow accounting and statistics at any time. You do not have to stop the message flow, integration server, or broker to make this change, nor do you have to redeploy the message flow.

“Analyzing message flow performance” on page 469

You can configure your broker to collect statistics about the operation and behavior of your message flows. Use this information to assess the performance of your message flows.

“Tuning message flow performance” on page 510

You can configure your message flows to enhance their performance in terms of resource usage, message flow throughput, and response times.

“Performance” on page 450

You can use message flow design and coding techniques to optimize the performance of your message flows. You can then analyze statistical information for your message flows and modify aspects of your broker configuration to tune the performance of your brokers and message flows.

Related reference:

 **mqsichangeflowstats** command

Use the **mqsichangeflowstats** command to control the accumulation of statistics about message flow operation.

 **mqsireportflowstats** command

Use the **mqsireportflowstats** command to display the current options for accounting and statistics that have been set using the **mqsichangeflowstats** command.

 Issuing commands to the z/OS console

You operate the broker using the z/OS START, STOP, and MODIFY commands.

Viewing accounting and statistics data in the IBM Integration Explorer:

You can use the Broker Statistics and Broker Statistics Graph views in the IBM Integration Explorer to view snapshot accounting and statistics data as it is produced by the integration node (broker).

Before you begin**Before you start:**

- “Starting accounting and statistics data collection in the IBM Integration Explorer” on page 485

- Read the concept topic about message flow accounting and statistics data

About this task

Use the accounting and statistics data displayed in the Broker Statistics and Broker Statistics Graph views to monitor the performance and resource usage of your integration node (broker) or integration server at the message flow, node or terminal level. The IBM Integration Explorer displays snapshot accounting and statistics data.

When you open the Broker Statistics and Broker Statistics Graph views, the IBM Integration Explorer connects to all the integration nodes that you have in your workspace and attempts to collect statistics data from the integration nodes. The message **Waiting for Data** is displayed in the title bar is displayed until accounting and statistics data are received from the selected integration node, integration server or message flow. When data is received from the integration node it is displayed in numeric form in the Broker Statistics view, and a visual representation of this data is shown in the Broker Statistics Graph view.

To view message flow accounting and statistics data using the IBM Integration Explorer:

Procedure

1. Click **Window > Show View > Broker Statistics** to open the Broker Statistics and Broker Statistics Graph view. These two views are displayed together. If you close one of the views, the other view is also be closed.
2. In the **WebSphere MQ Explorer - Navigator** view select the integration node, integration server or message flow for which you want to view accounting and statistics data.

Results

Accounting and statistics data for the selected integration node, integration server, or message flow is displayed in the Broker Statistics and Broker Statistics Graph views. It can take up to thirty seconds for accounting and statistics data to be received from the integration node and displayed in the Broker Statistics Graph view. Select individual or multiple items in the Broker Statistics view to change the information displayed in the Broker Statistics Graph view.

What to do next

If you want to examine the current data in the two views, click the **Pause** button in the Broker Statistics Graph view to prevent the current data from being overwritten with new data. Click the **Play** button to resume displaying new accounting and statistics data. You can change the visual representation of the data by clicking on the **Linear**, **Logarithmic**, and **Stacked** buttons.

You can also select the metrics that are displayed in the graph view, see “Filtering message flow accounting and statistics data in the IBM Integration Explorer” on page 508.

Related concepts:

“Message flow statistics and accounting data” on page 471

Message flow statistics and accounting data can be collected to record performance and operating details of one or more message flows.

Related tasks:

“Starting accounting and statistics data collection in the IBM Integration Explorer” on page 485

Use the IBM Integration Explorer to start collecting snapshot accounting and statistics data for your integration nodes (brokers), integration servers, and message flows. You can then view the accounting and statistics data in the Broker Statistics and Broker Statistics Graph views.

“Filtering message flow accounting and statistics data in the IBM Integration Explorer” on page 508

You can select the metrics for the snapshot accounting and statistics data that are displayed in the Broker Statistics and Broker Statistics Graph views in the IBM Integration Explorer.

“Starting collection of message flow accounting and statistics data with the **mqsichangeflowstats** command” on page 482

You can use the **mqsichangeflowstats** command to start collecting message flow accounting and statistics data for an active broker at any time. You can specify what type of statistics you want to collect, and where to send the data.

“Modifying message flow accounting and statistics data collection parameters” on page 505

You can modify the parameters that you have set for message flow accounting and statistics data collection. For example, you can start collecting data for a new message flow that you have deployed to an integration server for which you are already collecting data. You can modify parameters while data collection is active; you do not have to stop data collection and restart it.

“Analyzing message flow performance” on page 469

You can configure your broker to collect statistics about the operation and behavior of your message flows. Use this information to assess the performance of your message flows.

“Tuning message flow performance” on page 510

You can configure your message flows to enhance their performance in terms of resource usage, message flow throughput, and response times.

“Performance” on page 450

You can use message flow design and coding techniques to optimize the performance of your message flows. You can then analyze statistical information for your message flows and modify aspects of your broker configuration to tune the performance of your brokers and message flows.

Related reference:

 **mqsichangeflowstats** command

Use the **mqsichangeflowstats** command to control the accumulation of statistics about message flow operation.

 **mqsireportflowstats** command

Use the **mqsireportflowstats** command to display the current options for accounting and statistics that have been set using the **mqsichangeflowstats** command.

 Issuing commands to the z/OS console

You operate the broker using the z/OS START, STOP, and MODIFY commands.

Viewing accounting and statistics data in the web user interface:

You can use the web user interface to view snapshot accounting and statistics data.

Before you begin

Before you start:

- Read the topic “Message flow statistics and accounting data” on page 471
- Ensure that statistics collection is turned on; for more information, see “Starting collection of accounting and statistics data in the web user interface” on page 487

About this task

Use the snapshot statistics data displayed in the web user interface to monitor the performance of your integration node (broker) at the message flow or message flow node level. The snapshot data displayed in the web user interface is updated automatically every 20 seconds.

You can display high-level statistical data that provides an overview of all the message flows in an integration node, or you can display statistical information relating to a specific message flow or node. The level of information displayed depends on the type of resource that you have selected in the navigation tree.

You can use the statistics data in these views to compare the performance of message flows in your applications, and to help you identify the possible reasons for reduced performance in a message flow. For more information, see “Comparing the performance of message flows in an application” on page 515 and “Identifying the cause of a slow message flow” on page 512.

Procedure

- Use the information in the **Flow comparison** view to help you identify aspects of your message flows that might benefit from optimization. To see an overview of statistical information relating to all message flows in an integration node, integration server, application, library, or service, click the name of the required resource in the navigation tree, and then click the **Statistics** tab.

The **Flow comparison** view displays statistical information relating to all message flows in the selected resource, and you can use this information to see where you might enhance performance in terms of resource usage, message flow throughput, and response times. When you have identified a message flow that requires attention, you can use the **Flow analysis** view to see more detailed statistical information about the nodes in the flow.

- The **Throughput per message flow** section shows statistics relating to the throughput of messages in all message flows in the selected integration node, integration server, application, or library, during the last 20-second reporting period. The following information is provided for each message flow:
 - The number of messages per second processed by each message flow (the message rate)
 - The average elapsed time (in milliseconds) per message in the message flow
 - The average CPU time (in milliseconds)
 - The total number of messages processed by each message flow
 - The number of active threads in the message flow
 - The number of backouts processed by the message flow
 - The immediate parent of the resource (for example, the application that contains the message flow)

You can sort the information in the table based on any one of these statistics by clicking the header of the required column.

If you select a message flow in this table, the nodes contained by the flow are highlighted in the **Nodes for all flows** section.

A **Go to Flow analysis view** icon (a magnifying glass) is displayed next to each message flow in the table. You can click this icon to display the **Flow analysis** view, which contains detailed statistics for each node in the selected message flow.

- The **Nodes for all flows** section shows statistical information about all the message flow nodes in the running message flows that have statistics collection turned on, for the selected integration node, integration server, application, or library. Nodes for message flows that have statistics turned off are not shown in this table. The following information is provided for each node in the message flows:
 - The average CPU time (in milliseconds) for the message transactions in the node
 - The average elapsed time (in milliseconds) for the message transactions in the node
 - The total elapsed time (in milliseconds) for the message transactions in the node
 - The total CPU time (in milliseconds) for the message transactions in the node
 - The number of times that the node has been invoked
 - The type of node
 - The parent message flow for the node

You can sort the information in the table based on any one of these statistics by clicking the header of the required column. If you select a node in this table, its parent flow is highlighted in the **Throughput per message flow** section.

- Use the information in the **Flow analysis** view to assess the performance of a message flow. This view contains statistical information about the selected message flow, including data such as the message rate, CPU time, and elapsed time for messages processed by each node in the message flow. To view statistics about a specific message flow, click the name of the message flow in the navigation tree, and then click the **Statistics** tab.
 - The charts in this view show data for three statistics at a time. By default the charts show the message rate, average elapsed time, and average CPU time for the selected message flow; however, you can select any three statistics to display in the charts by clicking the arrow in the table next to each graph and then selecting the required statistic from the list. The tables show the highest, lowest, and average data for the whole of the current browser session, and the data for the latest 20-second reporting period.
 - The **Latest data per node** section shows statistical data for each node in the selected message flow, including the elapsed and CPU time per message processed by the node. You can choose to display the minimum, average, or maximum CPU and elapsed times per message, by clicking **Minimum**, **Average**, or **Maximum** in the section heading. You can also sort the data in the table by clicking on the heading of a column; for example, you can sort the data by CPU time by clicking the heading **CPU time (ms)**.

This table also shows the type of each node, which can help you to identify the potential cause of any performance problems; for more information, see “Identifying the cause of a slow message flow” on page 512.

- The **Flow profile** section shows a graphical representation of the selected message flow, displaying all nodes in the flow.

Related concepts:

“Message flow statistics and accounting data” on page 471

Message flow statistics and accounting data can be collected to record performance and operating details of one or more message flows.

Related tasks:

“Starting collection of accounting and statistics data in the web user interface” on page 487

You can use the web user interface to start collecting snapshot accounting and statistics data for your message flows. You can then view the accounting and statistics data in the web user interface.

“Stopping collection of accounting and statistics data in the web user interface” on page 495

You can use the web user interface to stop collecting accounting and statistics data for your message flows.

“Starting collection of message flow accounting and statistics data” on page 480

You can start collecting message flow accounting and statistics data for an active broker at any time. You can specify what type of statistics you want to collect, and where to send the data.

“Stopping collection of message flow accounting and statistics data” on page 491

You can stop collecting data for message flow accounting and statistics at any time. You do not have to stop the message flow, integration server, or broker to make this change, nor do you have to redeploy the message flow.

“Viewing message flow accounting and statistics data” on page 496

You can view snapshot accounting and statistics data as it is produced by the broker.

“Optimizing message flow throughput” on page 551

Each message flow that you design must provide a complete set of processing for messages received from a particular source. This design might result in very complex message flows that include large numbers of nodes that can cause a performance overhead, and might create potential bottlenecks. You can increase the number of message flows that process your messages to provide the opportunity for parallel processing and therefore improved throughput.

“Optimizing message flow response times” on page 554

You can use different solutions to improve message flow response times.

“Identifying the cause of a slow message flow” on page 512

You can use the message flow statistics data to help you identify aspects of a message flow that might be reducing the performance of the flow, and to help you understand how you can optimize it.

“Comparing the performance of message flows in an application” on page 515

You can compare the performance of message flows in an application to identify the flows that might benefit from tuning, and to find out where any specific performance problems might exist.

“Troubleshooting performance problems” on page 560

Follow this guidance to resolve common problems with IBM Integration Bus performance.

Related reference:



mqsireportflowstats command

Use the **mqsireportflowstats** command to display the current options for accounting and statistics that have been set using the **mqsichangeflowstats** command.

`mqsichangeflowstats` command

Use the `mqsichangeflowstats` command to control the accumulation of statistics about message flow operation.

Viewing and modifying data collection settings

You can view and modify the settings that control the collection of message flow accounting and statistics data.

Before you begin

Before you start:

- Start collecting accounting and statistics data; for more information, see “Starting collection of message flow accounting and statistics data” on page 480
- Read the concept topic “Message flow statistics and accounting data” on page 471

About this task

You can view and modify the settings that are used for the collection of message flow accounting and statistics data by completing the following tasks:

Procedure

- “Viewing message flow accounting and statistics data collection parameters” on page 504
- “Modifying message flow accounting and statistics data collection parameters” on page 505
- “Resetting message flow accounting and statistics archive data” on page 507

Related concepts:

“Message flow statistics and accounting data” on page 471

Message flow statistics and accounting data can be collected to record performance and operating details of one or more message flows.

Message flows overview

A message flow is a sequence of processing steps that run in the broker when an input message is received.

Related tasks:

“Starting collection of message flow accounting and statistics data” on page 480

You can start collecting message flow accounting and statistics data for an active broker at any time. You can specify what type of statistics you want to collect, and where to send the data.

Developing integration solutions

IBM Integration Bus provides a flexible environment in which you can develop integration solutions to transform, enrich, route, and process your business messages and data. You can integrate client applications that use different protocols and message formats.

Related reference:

Message flow statistics and accounting data

You can collect message flow statistics and accounting data in various output formats.

Issuing commands to the z/OS console

You operate the broker using the z/OS START, STOP, and MODIFY commands.

mqsichangeflowstats command

Use the **mqsichangeflowstats** command to control the accumulation of statistics about message flow operation.

mqsireportflowstats command

Use the **mqsireportflowstats** command to display the current options for accounting and statistics that have been set using the **mqsichangeflowstats** command.

Viewing message flow accounting and statistics data collection parameters:

You can view the parameters that are currently in effect for message flow accounting and statistics data collection by using the **mqsireportflowstats** command.

Before you begin

Before you start:

- Read the concept topic about message flow accounting and statistics data.
- Ensure that message flow statistics and accounting data are being collected. For more information, see “Starting collection of message flow accounting and statistics data” on page 480.

Procedure

Issue the **mqsireportflowstats** command to view the parameters that are currently being used to control archive or snapshot data collection. You can view the parameters for a broker, an integration server, or an individual message flow. For example, to view parameters for snapshot data for all message flows in all integration servers for BrokerA, enter:

```
mqsireportflowstats BrokerA -s -g -j
```

 Using SDSF on z/OS, enter:

```
/F BrokerA,rs s=yes,g=yes,j=yes
```

Refer to the **mqsireportflowstats** command for further examples.

Results

The **mqsireportflowstats** command displays the current status in BIP81871I.

What to do next

Next:

You can now modify the data collection parameters.

Related concepts:

Message flows overview

A message flow is a sequence of processing steps that run in the broker when an input message is received.

“Message flow statistics and accounting data” on page 471

Message flow statistics and accounting data can be collected to record performance and operating details of one or more message flows.

Related tasks:

Developing integration solutions

IBM Integration Bus provides a flexible environment in which you can develop integration solutions to transform, enrich, route, and process your business messages and data. You can integrate client applications that use different protocols and message formats.

“Starting collection of message flow accounting and statistics data with the **mqsichangeflowstats** command” on page 482

You can use the **mqsichangeflowstats** command to start collecting message flow accounting and statistics data for an active broker at any time. You can specify what type of statistics you want to collect, and where to send the data.

“Modifying message flow accounting and statistics data collection parameters”

You can modify the parameters that you have set for message flow accounting and statistics data collection. For example, you can start collecting data for a new message flow that you have deployed to an integration server for which you are already collecting data. You can modify parameters while data collection is active; you do not have to stop data collection and restart it.

Related reference:

Message flow statistics and accounting data

You can collect message flow statistics and accounting data in various output formats.

Issuing commands to the z/OS console

You operate the broker using the z/OS START, STOP, and MODIFY commands.

mqsichangeflowstats command

Use the **mqsichangeflowstats** command to control the accumulation of statistics about message flow operation.

mqsireportflowstats command

Use the **mqsireportflowstats** command to display the current options for accounting and statistics that have been set using the **mqsichangeflowstats** command.

“Diagnostic messages” on page 891

Diagnostic messages are listed in this section in numeric order, grouped according to the component to which they relate.

Modifying message flow accounting and statistics data collection parameters:

You can modify the parameters that you have set for message flow accounting and statistics data collection. For example, you can start collecting data for a new message flow that you have deployed to an integration server for which you are already collecting data. You can modify parameters while data collection is active; you do not have to stop data collection and restart it.

Before you begin

Before you start:

- Start to collect message flow accounting and statistics data
- Read the concept topic about message flow accounting and statistics data

About this task

To modify message flow accounting and statistics parameters:

Procedure

1. Decide which data collection parameters you want to change. You can modify the parameters that are in force for a broker, an integration server, or an individual message flow.
2. Issue the **mqsichangeflowstats** command with the appropriate parameters to modify the parameters that are currently being used by the broker to control archive data collection or snapshot data collection.

For example, to modify parameters to extend snapshot data collection to a new message flow MFlow2 in integration server EG2 for BrokerA, enter:

```
mqsichangeflowstats BrokerA -s -e EG2 -f MFlow2 -c active
```

 Using SDSF on z/OS, enter:

```
/F BrokerA,cs s=yes,e=EG2,f=MFlow2,c=active
```

If you want to specify an accounting origin for archive data for a particular message flow in an integration server, enter:

```
mqsichangeflowstats BrokerA -a -e EG4 -f MFlowX -b basic
```

 Using SDSF on z/OS, enter:

```
/F BrokerA,cs a=yes,e=EG4,f=MFlowX,b=basic
```

Refer to the **mqsichangeflowstats** command for further examples.

Results

When the command completes successfully, the new parameters that you have specified for data collection are in force. These parameters remain in force until you stop data collection or make further modifications.

Related concepts:

“Message flow statistics and accounting data” on page 471

Message flow statistics and accounting data can be collected to record performance and operating details of one or more message flows.

 Message flows overview

A message flow is a sequence of processing steps that run in the broker when an input message is received.

Related tasks:

“Starting collection of message flow accounting and statistics data with the **mqsichangeflowstats** command” on page 482

You can use the **mqsichangeflowstats** command to start collecting message flow accounting and statistics data for an active broker at any time. You can specify what type of statistics you want to collect, and where to send the data.

 Developing integration solutions

IBM Integration Bus provides a flexible environment in which you can develop integration solutions to transform, enrich, route, and process your business messages and data. You can integrate client applications that use different protocols and message formats.

Related reference:

 Message flow statistics and accounting data

You can collect message flow statistics and accounting data in various output

formats.



Issuing commands to the z/OS console

You operate the broker using the z/OS START, STOP, and MODIFY commands.



mqsichangeflowstats command

Use the **mqsichangeflowstats** command to control the accumulation of statistics about message flow operation.



mqsireportflowstats command

Use the **mqsireportflowstats** command to display the current options for accounting and statistics that have been set using the **mqsichangeflowstats** command.

Resetting message flow accounting and statistics archive data:

You can reset message flow accounting and statistics archive data to purge any accounting and statistics data not yet reported for that collecting interval. This removes unwanted data. You can request this at any time; you do not have to stop data collection and restart it to perform reset. You cannot reset snapshot data.

Before you begin

Before you start:

- Start to collect message flow accounting and statistics data
- Read the concept topic about message flow accounting and statistics data

About this task

To reset message flow accounting and statistics archive data:

Procedure

1. Identify the broker, and optionally the integration server, for which you want to reset archive data. You cannot reset archive data on a message flow basis.
2. Issue the **mqsichangeflowstats** command with the appropriate parameters to reset archive data.

For example, to reset archive data for BrokerA, enter:

```
mqsichangeflowstats BrokerA -a -g -j -r
```

 Using SDSF on z/OS, enter:

```
/F BrokerA,cs a=yes,g=yes,j=yes,r=yes
```

Results

When this command completes, all accounting and statistics data accumulated so far for this interval are purged and will not be included in the reports. Data collection is restarted from this point. All archive data for all flows (indicated by -j or j=yes) in all integration servers (indicated by -g or g=yes) is reset.

This command has a minimal effect on snapshot data because the accumulation interval is much shorter than for archive data. It does not effect the settings for archive or snapshot data collection that are currently in force. When the command has completed, data collection resumes according to the current settings.

You can change any other options that are currently in effect when you reset archive data, for example accounting origin settings or output type.

Related concepts:



Message flows overview

A message flow is a sequence of processing steps that run in the broker when an input message is received.

“Message flow statistics and accounting data” on page 471

Message flow statistics and accounting data can be collected to record performance and operating details of one or more message flows.

Related tasks:

“Starting collection of message flow accounting and statistics data with the **mqsichangeflowstats** command” on page 482

You can use the **mqsichangeflowstats** command to start collecting message flow accounting and statistics data for an active broker at any time. You can specify what type of statistics you want to collect, and where to send the data.

“Modifying message flow accounting and statistics data collection parameters” on page 505

You can modify the parameters that you have set for message flow accounting and statistics data collection. For example, you can start collecting data for a new message flow that you have deployed to an integration server for which you are already collecting data. You can modify parameters while data collection is active; you do not have to stop data collection and restart it.



Developing integration solutions

IBM Integration Bus provides a flexible environment in which you can develop integration solutions to transform, enrich, route, and process your business messages and data. You can integrate client applications that use different protocols and message formats.

Related reference:



Message flow statistics and accounting data

You can collect message flow statistics and accounting data in various output formats.



Issuing commands to the z/OS console

You operate the broker using the z/OS START, STOP, and MODIFY commands.



mqsichangeflowstats command

Use the **mqsichangeflowstats** command to control the accumulation of statistics about message flow operation.



mqsireportflowstats command

Use the **mqsireportflowstats** command to display the current options for accounting and statistics that have been set using the **mqsichangeflowstats** command.

Filtering message flow accounting and statistics data in the IBM Integration Explorer

You can select the metrics for the snapshot accounting and statistics data that are displayed in the Broker Statistics and Broker Statistics Graph views in the IBM Integration Explorer.

Before you begin

Before you start:

- Read the concept topic about message flow accounting and statistics data
- “Starting accounting and statistics data collection in the IBM Integration Explorer” on page 485
- “Viewing accounting and statistics data in the IBM Integration Explorer” on page 497

About this task

When you open the Broker Statistics and Broker Statistics Graph views, the IBM Integration Explorer connects to all the brokers that you have in your workspace and attempts to collect snapshot statistics data from the brokers. The message Waiting for Data is displayed in the title bar until accounting and statistics data are received from the selected broker, integration server, or message flow.

When data is received from the broker, it is displayed in numeric form in the Broker Statistics view, and a visual representation of this data is shown in the Broker Statistics Graph view. When accounting and data statistics data has started to be collected and displayed for your broker, integration server, or message flow, you can select the metrics that are displayed in the Broker Statistics and Broker Statistics Graph views.

The metrics that you can select depend on whether you are viewing the accounting and statistics data from a broker, an integration server, or a message flow. To view a list of the metrics that you can filter on, see Metrics for accounting and statistics data in the IBM Integration Explorer.

To filter the message flow accounting and statistics data in the Broker Statistics and Broker Statistics Graph views:

Procedure

1. Click **Window > Show View > Broker Statistics** to open the Broker Statistics and Broker Statistics Graph view.
2. In the **WebSphere MQ Explorer - Navigator** view, select the broker, integration server, or message flow for which you want to view accounting and statistics data.
3. Click the **Filter** button in the Broker Statistics Graph view. The Select metrics to graph window is displayed.
4. Select the metrics that you want to display from the list in the Broker Statistics and Broker Statistics Graph view. Clear all the metrics that you do not want to display in the Broker Statistics and Broker Statistics Graph view.
5. Click **OK**.

Results

The selected metrics are displayed in the Broker Statistics and Broker Statistics Graph views. Data for additional metrics that you have selected are displayed the next time the statistics data is refreshed.

Related concepts:

“Message flow statistics and accounting data” on page 471

Message flow statistics and accounting data can be collected to record performance and operating details of one or more message flows.

Related tasks:

“Starting accounting and statistics data collection in the IBM Integration Explorer” on page 485

Use the IBM Integration Explorer to start collecting snapshot accounting and statistics data for your integration nodes (brokers), integration servers, and message flows. You can then view the accounting and statistics data in the Broker Statistics and Broker Statistics Graph views.

“Viewing accounting and statistics data in the IBM Integration Explorer” on page 497

You can use the Broker Statistics and Broker Statistics Graph views in the IBM Integration Explorer to view snapshot accounting and statistics data as it is produced by the integration node (broker).

“Starting collection of message flow accounting and statistics data with the **mqsichangeflowstats** command” on page 482

You can use the **mqsichangeflowstats** command to start collecting message flow accounting and statistics data for an active broker at any time. You can specify what type of statistics you want to collect, and where to send the data.

“Modifying message flow accounting and statistics data collection parameters” on page 505

You can modify the parameters that you have set for message flow accounting and statistics data collection. For example, you can start collecting data for a new message flow that you have deployed to an integration server for which you are already collecting data. You can modify parameters while data collection is active; you do not have to stop data collection and restart it.

Related reference:



mqsichangeflowstats command

Use the **mqsichangeflowstats** command to control the accumulation of statistics about message flow operation.



mqsireportflowstats command

Use the **mqsireportflowstats** command to display the current options for accounting and statistics that have been set using the **mqsichangeflowstats** command.



Issuing commands to the z/OS console

You operate the broker using the z/OS START, STOP, and MODIFY commands.

Tuning message flow performance

You can configure your message flows to enhance their performance in terms of resource usage, message flow throughput, and response times.

Before you begin

Before you start:

Read the following topics:

- “Performance” on page 450
- “Performance planning” on page 451
- “Analyzing message flow performance” on page 469

About this task

The following aspects of message flow design typically have the greatest costs in terms of message flow performance, and you can optimize performance by limiting the number of times that each of these actions occurs:

- Parsing
- Tree navigation and copying
- Accessing resources
- Processing logic

The following topics provide information about how to improve the performance of your message flows:

- “Identifying the cause of a slow message flow” on page 512
- “Comparing the performance of message flows in an application” on page 515
- “Message flow design and performance” on page 453
- “Code design and performance” on page 455

For information about how you can assess the current performance of your message flows, see “Analyzing message flow performance” on page 469.

For troubleshooting information about how to deal with specific performance issues, see “Troubleshooting performance problems” on page 560.

The following topics contain reference information that you might find helpful when tuning the performance of your message flows:

- Message flow accounting and statistics details
- Message flow accounting and statistics output formats
- Example message flow accounting and statistics data
- Metrics for accounting and statistics data in the IBM Integration Explorer

Related concepts:

“Message flow statistics and accounting data” on page 471

Message flow statistics and accounting data can be collected to record performance and operating details of one or more message flows.



Message flows overview

A message flow is a sequence of processing steps that run in the broker when an input message is received.

“System resources for message flow development” on page 517

Configure your message flows to make the best use of computer resources, especially if you will process large messages.

Related tasks:



Developing integration solutions

IBM Integration Bus provides a flexible environment in which you can develop integration solutions to transform, enrich, route, and process your business messages and data. You can integrate client applications that use different protocols and message formats.

“Viewing message flow accounting and statistics data” on page 496

You can view snapshot accounting and statistics data as it is produced by the broker.

“Analyzing message flow performance” on page 469

You can configure your broker to collect statistics about the operation and behavior of your message flows. Use this information to assess the performance of your

message flows.

“Optimizing message flow throughput” on page 551

Each message flow that you design must provide a complete set of processing for messages received from a particular source. This design might result in very complex message flows that include large numbers of nodes that can cause a performance overhead, and might create potential bottlenecks. You can increase the number of message flows that process your messages to provide the opportunity for parallel processing and therefore improved throughput.

“Optimizing message flow response times” on page 554

You can use different solutions to improve message flow response times.

Identifying the cause of a slow message flow

You can use the message flow statistics data to help you identify aspects of a message flow that might be reducing the performance of the flow, and to help you understand how you can optimize it.

Before you begin

Before you start:

Read the following topics:

- “Analyzing message flow performance” on page 469
- “Tuning message flow performance” on page 510
- “Message flow statistics and accounting data” on page 471

About this task

Follow these steps to identify the factors that might be limiting the speed of a message flow, and to find out how you can increase the overall performance:

Procedure

1. Use the web user interface to display the statistics for your message flow. You can use the information in the **Flow analysis** view to assess the performance of your message flows. This view contains detailed statistical information about the selected message flow, including statistics such as the message rate, CPU time, and elapsed time for each node in the message flow. You can use these statistics to help you identify the nodes in your message flow that are the most expensive in terms of performance.

Use the charts in the **Flow analysis** view to show data for any three statistics at a time; by default, the charts show the message rate, average elapsed time, and average CPU time for the nodes in the selected message flow:

- Use the **Message rate** chart to check the period of time during which the message rate has been low.
- Use the **Average elapsed time** chart with the **Message rate** chart to check whether there is a correlation between the two statistics for this flow. If there is an increase in **Average elapsed time** and a reduction in **Message rate**, check the **Total input messages** to see whether messages have been entering the message flow. An increase in **Average elapsed time** might indicate that there is a bottleneck in message processing that might be caused by waiting for an external resource, such as a database. Check the data in the **Latest data per node** table to identify nodes with high elapsed times.
- Use the **Average CPU time** chart to see whether the CPU time has increased or decreased in correlation with the **Message rate**. If the CPU time increases and the message rate decreases, it might indicate that there is a problem with

the throughput of the message flow. If the **Average elapsed time** is high and the **Average CPU time** is low, it is likely that the message flow is waiting for something other than CPU; this is typically a slow node or invocation of an application or service. If the **Average elapsed time** and **Average CPU time** are both high, the message flow is likely to be CPU bound; in this case, verify that all known design, tuning, and coding optimizations have been applied. For example, check the size of the messages that are being processed to see if it has increased in line with the drop in rate. If the CPU has recently increased, check the commit counts, errors, and backouts. For more information about designing for performance, see “Message flow design and performance” on page 453 and “Code design and performance” on page 455.

For more information about viewing statistics data, see “Viewing accounting and statistics data in the web user interface” on page 499.

2. When you have identified the nodes that are the most expensive in terms of performance, identify the type of node; some of the techniques that you can use to optimize the performance vary according to the node type:
 - If the most expensive node is a Compute node, there might be issues with your ESQL code. For information about how to optimize ESQL, see “ESQL code tips” on page 456.
 - If the most expensive node is a JavaCompute node, see “Java code tips” on page 462 for information about how you might optimize performance for the node.
 - If the most expensive node uses XPath, see “XPath and XSLT code tips” on page 464 to find out how you might optimize performance for the node.
 - If the most expensive node is a request node, it might be waiting for input from an external resource. You can use the activity log to see where resources such as databases are taking a long time to respond or process data; for more information, see “Using Activity logs” on page 623.
3. Consider the overall design of your message flow; the following design decisions can have a significant impact on message flow throughput and response times:
 - The number of threads processing messages in a single message flow
 - The number of copies of a message flow
 - The scope of the message flow
 - The frequency of commits
 - The number of nodes in the message flow
 - The way in which the message flow routes and processes messages
 - The use of loops
 - The use of persistent and transactional messages
 - Message size
 - Message format

For more information about these design decisions, see “Message flow design and performance” on page 453.

4. Parsing can have a significant impact on message flow performance, and there are techniques that you can use to limit the effect. For more information, see “Parsing and message flow performance” on page 466.
5. Message tree navigation and message tree copying can reduce message flow performance, so it is important to use them appropriately and limit their use where possible. For more information, see “Message tree navigation and copying” on page 467.

What to do next

For more information about improving performance the performance of your message flows, see “Tuning message flow performance” on page 510. For information about solving specific performance issues, see “Troubleshooting performance problems” on page 560.

Related tasks:

“Performance” on page 450

You can use message flow design and coding techniques to optimize the performance of your message flows. You can then analyze statistical information for your message flows and modify aspects of your broker configuration to tune the performance of your brokers and message flows.

“Viewing accounting and statistics data in the web user interface” on page 499

You can use the web user interface to view snapshot accounting and statistics data.

“Viewing accounting and statistics data in the IBM Integration Explorer” on page 497

You can use the Broker Statistics and Broker Statistics Graph views in the IBM Integration Explorer to view snapshot accounting and statistics data as it is produced by the integration node (broker).

“Tuning message flow performance” on page 510

You can configure your message flows to enhance their performance in terms of resource usage, message flow throughput, and response times.



Developing message flows

Develop message flows to process your business messages and data. A message flow is a sequence of processing steps that run in an integration node when an input message is received.

“Using Activity logs” on page 623

Use Activity logs to get an overview of recent activities in your message flows and associated external resources.

“Message flow design and performance” on page 453

Consider the performance implications arising from the design of your message flow.

“Parsing and message flow performance” on page 466

Understand the impact of parsing on message flow performance, and use techniques to limit the effect and improve performance.

“Message tree navigation and copying” on page 467

Message tree navigation and message tree copying can reduce message flow performance, so it is important to use them appropriately and limit their usage where possible.

“ESQL code tips” on page 456

You can improve message flow performance with ESQL by using some optimization techniques.

“Java code tips” on page 462

You can improve message flow performance with Java by using some optimization techniques.

“XPath and XSLT code tips” on page 464

You can improve message flow performance with XPath and XSLT by using some optimization techniques.

“Analyzing message flow performance” on page 469

You can configure your broker to collect statistics about the operation and behavior of your message flows. Use this information to assess the performance of your

message flows.

“Troubleshooting performance problems” on page 560

Follow this guidance to resolve common problems with IBM Integration Bus performance.

“Comparing the performance of message flows in an application”

You can compare the performance of message flows in an application to identify the flows that might benefit from tuning, and to find out where any specific performance problems might exist.

Related reference:



Activity logs

Activity logs contain information about message flows and how they interact with external resources. They can be generated for a message flow or a resource type. Learn about the structure and content of Activity logs, and about the *tags* that enrich the log data and can be used to filter its content.

Comparing the performance of message flows in an application

You can compare the performance of message flows in an application to identify the flows that might benefit from tuning, and to find out where any specific performance problems might exist.

Before you begin

Before you start:

Read the following topics:

- “Performance” on page 450
- “Performance planning” on page 451
- “Tuning message flow performance” on page 510
- “Analyzing message flow performance” on page 469

About this task

You can use the web user interface to display snapshot accounting and statistics information about all the message flows in an integration node (broker). The **Flow comparison** view shows statistical information relating to all message flows in the selected resource, which can be an integration node (broker), integration server (integration server), application, or library. You can use this data to compare the performance of message flows with the performance of other message flows, enabling you to identify those that might benefit from tuning to enhance their performance in terms of resource usage, message flow throughput, and response times.

When you have identified a message flow that requires attention, you can use the **Flow analysis** view in the web user interface to see more specific statistical information about the nodes in the message flow. This data can help you to identify the aspects of the message flow that might be reducing the overall performance. For more information about the statistical information that you can use to understand the performance of your message flows, see “Viewing accounting and statistics data in the web user interface” on page 499.

For information about the steps that you can take to optimize the performance of the message flows, see “Identifying the cause of a slow message flow” on page 512.

For more information about improving the performance of your message flows, see “Tuning message flow performance” on page 510, and for information about solving specific performance issues, see “Troubleshooting performance problems” on page 560.

Related tasks:

“Performance” on page 450

You can use message flow design and coding techniques to optimize the performance of your message flows. You can then analyze statistical information for your message flows and modify aspects of your broker configuration to tune the performance of your brokers and message flows.

“Viewing accounting and statistics data in the web user interface” on page 499

You can use the web user interface to view snapshot accounting and statistics data.

“Viewing accounting and statistics data in the IBM Integration Explorer” on page 497

You can use the Broker Statistics and Broker Statistics Graph views in the IBM Integration Explorer to view snapshot accounting and statistics data as it is produced by the integration node (broker).

“Tuning message flow performance” on page 510

You can configure your message flows to enhance their performance in terms of resource usage, message flow throughput, and response times.



Developing message flows

Develop message flows to process your business messages and data. A message flow is a sequence of processing steps that run in an integration node when an input message is received.

“Using Activity logs” on page 623

Use Activity logs to get an overview of recent activities in your message flows and associated external resources.

“Message flow design and performance” on page 453

Consider the performance implications arising from the design of your message flow.

“Code design and performance” on page 455

Consider the performance implications arising from the design of your code.

“Parsing and message flow performance” on page 466

Understand the impact of parsing on message flow performance, and use techniques to limit the effect and improve performance.

“Message tree navigation and copying” on page 467

Message tree navigation and message tree copying can reduce message flow performance, so it is important to use them appropriately and limit their usage where possible.

“ESQL code tips” on page 456

You can improve message flow performance with ESQL by using some optimization techniques.

“Java code tips” on page 462

You can improve message flow performance with Java by using some optimization techniques.

“XPath and XSLT code tips” on page 464

You can improve message flow performance with XPath and XSLT by using some optimization techniques.

“Analyzing message flow performance” on page 469

You can configure your broker to collect statistics about the operation and behavior of your message flows. Use this information to assess the performance of your

message flows.

“Identifying the cause of a slow message flow” on page 512

You can use the message flow statistics data to help you identify aspects of a message flow that might be reducing the performance of the flow, and to help you understand how you can optimize it.

“Troubleshooting performance problems” on page 560

Follow this guidance to resolve common problems with IBM Integration Bus performance.

Related reference:



Activity logs

Activity logs contain information about message flows and how they interact with external resources. They can be generated for a message flow or a resource type. Learn about the structure and content of Activity logs, and about the *tags* that enrich the log data and can be used to filter its content.

System resources for message flow development

Configure your message flows to make the best use of computer resources, especially if you will process large messages.

As well as designing your message flow to optimize throughput, you must ensure that particular areas of storage are efficiently used so that your system does not suffer from capacity issues, and that processes do not end because of lack of resources.

Consider the following storage issues when developing your message flows:

- “Stack storage”
- “JVM heap sizing” on page 518
- “Configuring the storage of events for aggregation nodes” on page 520
- “Configuring the storage of events for Collector nodes” on page 522
- “Configuring the storage of events for Resequencing nodes” on page 524
- “Configuring the storage of events for timeout nodes” on page 527

Related tasks:

“Optimizing message flow response times” on page 554

You can use different solutions to improve message flow response times.

Related reference:



Message flows

Use the reference information in this section to develop your message flows and related resources.

Stack storage:

Depending on the design of your message flow, you might need to increase the stack size.

When a message flow thread starts, it requires storage to perform the instructions that are defined by the message flow nodes. This storage comes from the integration server's heap and stack size. The default stack size that is allocated to a message flow thread depends on the operating system that is used:



On Windows, each message flow thread is allocated 1 MB of stack space.

Linux

On Linux, each message flow thread is allocated 1 MB of stack space.

UNIX

On UNIX, each message flow thread is allocated 1 MB of stack space.

AIX

On AIX, each message flow thread is allocated 2 MB of stack space.

z/OS

On z/OS, each message flow thread is allocated 512 KB of downward stack space and 50 KB of upward stack space.

In a message flow, a node typically uses 2 KB of the stack space. A typical message flow can therefore include 250 nodes on z/OS, 500 nodes on UNIX systems and 500 nodes on Windows. This amount can be higher or lower depending on the type of nodes used and the processing that they perform.

In IBM Integration Bus, any processing that involves nested or recursive processing can cause extensive usage of the stack. For example, in the following situations you might need to increase the stack size:

- When a message flow is processing a message that contains a large number of repetitions or complex nesting.
- When a message flow is executing ESQL that calls the same procedure or function recursively, or when an operator (for example, the concatenation operator) is used repeatedly in an ESQL statement.

You can increase the stack size to improve performance. For details, see:

- “Increasing the stack size on Windows, Linux, and UNIX systems” on page 545
- “Increasing the stack size on z/OS” on page 546

Related tasks:

“Optimizing message flow response times” on page 554

You can use different solutions to improve message flow response times.

Related reference:**Message flows**

Use the reference information in this section to develop your message flows and related resources.

JVM heap sizing:

The Java virtual machine (JVM) heap is an independent memory allocation that can reduce the capacity of the main memory heap.

Every integration server creates its own JVM. The integration server uses the JVM to execute the internal administration threads that require Java. This usage is typically minimal. The primary use of the JVM is for the message flow nodes that include the IBM primitives, user defined extensions, and Java routines that are called from ESQL. You cannot control how much of the JVM heap the IBM primitives use, but you can affect usage of the JVM heap in the Java that is implemented in the following resources:

- A Java user-defined extension node
- A JavaCompute node
- A Java routine that is called from ESQL
- Mapping notes running a graphical data map
- XSLTransform node

- SOAP nodes

From WebSphere Message Broker Version 6.1 onwards, the JVM is created with a minimum of 32 MB of space, and a maximum of 256 MB, allocated and reserved for its use. As with any JVM, you can pass parameters in to set the minimum and maximum heap sizes. Note that on 32-bit platforms, the JVM reserves heap space based on the maximum heap size. On 32-bit Windows, you might experience warnings if you set the JVM heap size to 512 MB or higher.

You might need to increase the maximum heap size allocated if you plan to run large messages through the Java primitive nodes listed above.

To give more capacity to a message flow that is going to process large messages, reduce the minimum JVM heap size to allow the main memory heap to occupy more address space. For details of how to reduce the minimum JVM heap size, see “Setting the JVM heap size” on page 544.

The key indicators of a successful JVM heap setting are:

- GC overhead - the percentage of time spent in stop-the-world GC versus the time spent running the application
- GC pause times - the length of the stop-the-world GC cycles

The requirements on these two values vary according to what you are trying to achieve; for a batch application, a low GC overhead would be important, but pause times might be less relevant. For real-time applications, the pause times are more important than the overall GC overhead.

It is often possible to achieve GC overheads of less than 1 percent and pause times under 1 second (with the exception of compaction cycles). Typically, if the GC overhead is more than 10 percent, it can be reduced by tuning. Pause times of more than 2 seconds are often avoidable, typically by tuning to avoid compaction.

The optimum values for any situation can only be determined by experimenting with values and monitoring the results, because there are so many variables, including the complexity of the message flow and the size of the messages being processed.

You could begin by running a single instance of the message flow using the default heap size, and then increasing the message size and number of instances in a controlled way to their maximum values. Ensure that you make one change at a time. While you are doing this, monitor JVM heap usage and GC activity using resource statistics to ensure that the heap size is sufficiently large to process the largest messages without exceeding the targets for GC overhead and GC pause times. Also ensure that you monitor JVM heap usage on a regular basis, and ensure that the recommended thresholds are not being exceeded.

For more detailed analysis, you can use the Java Health Center, which is available through the IBM Support Assistant.

Related concepts:

“Stack storage” on page 517

Depending on the design of your message flow, you might need to increase the stack size.

Related tasks:

“Setting the JVM heap size” on page 544

When you start an integration server, it creates a Java virtual machine (JVM) for executing a Java user-defined node.

“Optimizing message flow response times” on page 554

You can use different solutions to improve message flow response times.

Related reference:

 Message flows

Use the reference information in this section to develop your message flows and related resources.

Configuring the storage of events for aggregation nodes:

You can use an Aggregation configurable service to control the storage of events for AggregateControl and AggregateReply nodes.

About this task

By default, the storage queues used by all aggregation nodes are:

- SYSTEM.BROKER.AGGR.CONTROL
- SYSTEM.BROKER.AGGR.REPLY
- SYSTEM.BROKER.AGGR.REQUEST
- SYSTEM.BROKER.AGGR.UNKNOWN
- SYSTEM.BROKER.AGGR.TIMEOUT

However, you can control the queues that are used by different aggregation nodes by creating alternative queues containing a *QueuePrefix*, and using an Aggregation configurable service to specify the names of those queues for storing events.

Follow these steps to specify the queues that are used to store event states, and to set the expiry time of an aggregation:

Procedure

1. Create the storage queues to be used by the aggregation nodes. The following queues are required:
 - SYSTEM.BROKER.AGGR.*QueuePrefix*.CONTROL
 - SYSTEM.BROKER.AGGR.*QueuePrefix*.REPLY
 - SYSTEM.BROKER.AGGR.*QueuePrefix*.REQUEST
 - SYSTEM.BROKER.AGGR.*QueuePrefix*.UNKNOWN
 - SYSTEM.BROKER.AGGR.*QueuePrefix*.TIMEOUT

The *QueuePrefix* variable can contain any characters that are valid in a WebSphere MQ queue name, but must be no longer than eight characters and must not begin or end with a period (.). For example, SET1 and SET.1 are valid queue prefixes, but .SET1 and SET1. are invalid.

If you do not create the storage queues, IBM Integration Bus creates the set of queues when the node is deployed; these queues are based on the default queues. If the queues cannot be created, the message flow is not deployed.

2. Use the **mqsicreateconfigurableservice** command to create an Aggregation configurable service. You can create a configurable service to be used with either a specific aggregation or with all aggregations in an integration server.
 - a. If the configurable service is to be used with a specific aggregation, ensure that the name of the configurable service is the same as the name that you

specify in the Aggregate name property on the AggregateControl and AggregateReply nodes. If the configurable service is to be used with all aggregations in an integration server, create the configurable service with the same name as the integration server.

- b. Set the **Queue prefix** property to the required value.
- c. Optional: Set the **Timeout** property to control the expiry time of an aggregation.

For example, create a configurable service called myAggregation, which specifies queues prefixed with SYSTEM.BROKER.AGGR.SET1 and a timeout of 60 seconds:

```
mqscreateconfigurableservice MYBROKER -c Aggregation -o myAggregation  
-n queuePrefix,timeoutSeconds -v SET1,60
```


You can use the **mqsdeleteconfigurable**service command to delete the Aggregation configurable service. However, the storage queues are not deleted automatically when the configurable service is deleted, so you must delete them separately. For more information, see Configurable services properties

3. In the AggregateControl and AggregateReply nodes:
 - a. Ensure that the name of the Aggregation configurable service is the same as the name specified in the Aggregate name property on the **Basic** tab; for example, myAggregation. If no Aggregation configurable service exists with the same name as the Aggregate name property, and if a configurable service exists with the same name as the integration server, that configurable service is used instead.
 - b. Optional: Use the **mqschange**properties and **mqsireport**properties commands to change or view the properties of the configurable service. Alternatively, you can use the IBM Integration Explorer to view or modify a configurable service. For more information about working with configurable services, see Using the IBM Integration Explorer to work with configurable services.


What to do next


The properties for the configurable service are not used by the broker until you restart or redeploy the message flow, or restart the broker.


Related concepts:


 Using the IBM Integration Explorer to work with configurable services
Configurable services are used to define properties that are related to external services on which the integration node (broker) relies. Use the IBM Integration Explorer to view, add, modify and delete configurable services.

Related tasks:


 Viewing configurable services
Configurable services are used to define properties that are related to external services on which the integration node (broker) relies. Use the IBM Integration Explorer to view the properties of configurable services defined on your integration node.


 Modifying an IBM defined configurable service
You can create a new configurable service for an external service on which the broker relies, by modifying predefined configurable services provided by IBM. Use the IBM Integration Explorer to view and modify existing configurable services.


 **Modifying a configurable service**
Use the IBM Integration Explorer to view and modify existing configurable services.


 **Deleting a configurable service**
Use the IBM Integration Explorer to delete custom configurable services.


Related reference:


 **AggregateControl node**
Use the AggregateControl node to mark the beginning of a fan-out of requests that are part of an aggregation.


 **AggregateRequest node**
Use the AggregateRequest node to record the fact that request messages have been sent. This node also collects information that helps the AggregateReply node to construct the compound response message.

 **AggregateReply node**
Use the AggregateReply node to mark the end of an aggregation fan-in. This node collects replies and combines them into a single compound message.

 **Configurable services properties**
The supplied configurable services, and the configurable services that you create, are defined by their names and properties. You can use the supplied services.

 **mqsicreateconfigurable service** command
Use the **mqsicreateconfigurable service** command to create an object name for a broker external resource.

 **mqsichangeproperties** command
Use the **mqsichangeproperties** command to modify broker properties and properties of broker resources.

 **mqsireportproperties** command
Use the **mqsireportproperties** command to display properties that relate to a broker, an integration server, or a configurable service.

Configuring the storage of events for Collector nodes:

You can use a Collector configurable service to control the storage of events for Collector nodes.

About this task

By default, the storage queues used by all Collector nodes are:

- SYSTEM.BROKER.EDA.EVENTS
- SYSTEM.BROKER.EDA.COLLECTIONS

These queues are also used by the Resequencing node.

However, you can control the queues that are used by different Collector nodes by creating alternative queues that contain a *QueuePrefix* variable, and by using a Collector configurable service to specify the names of those queues for storing events.

Follow these steps to specify the queues that are used to store event states, and to set the expiry for the collection:

Procedure

1. Create the storage queues to be used by the Collector node. The following queues are required:

- SYSTEM.BROKER.EDA.QueuePrefix.EVENTS
- SYSTEM.BROKER.EDA.QueuePrefix.COLLECTIONS

The *QueuePrefix* variable can contain any characters that are valid in a WebSphere MQ queue name, but must be no longer than eight characters and must not begin or end with a period (.). For example, SET1 and SET.1 are valid queue prefixes, but .SET1 and SET1. are invalid.

If you do not create the storage queues, IBM Integration Bus creates the set of queues when the node is deployed; these queues are based on the default queues. If the queues cannot be created, the message flow is not deployed.

2. Use the **mqsicreateconfigurable** command to create a Collector configurable service. You can create a configurable service to be used with either a specific collection or with all collections in an integration server.
 - a. If you are creating a configurable service to be used with a specific collection, ensure that the name of the configurable service is the same as the name that you specify in the Configurable service property on the Collector node. If you are creating a configurable service to be used with all collections in the integration server, ensure that the configurable service has the same name as the integration server.
 - b. Set the **Queue prefix** property to the required value.
 - c. Optional: Set the **Collection expiry** property.

For example, create a Collector configurable service called `myCollectorService`, which uses queues prefixed with `SYSTEM.BROKER.EDA.SET1`, and with a collection expiry of 60 seconds:

```
mqsicreateconfigurable MYBROKER -c Collector -o myCollectorService  
-n queuePrefix,collectionExpirySeconds -v SET1,60
```

You can use the **mqsdeleteconfigurable** command to delete the Collector configurable service. However, the storage queues are not deleted automatically when the configurable service is deleted, so you must delete them separately.


For more information, see Configurable services properties

3. In the Collector node:
 - a. If the configurable service is to be used for a specific collection, specify the name of the configurable service in the Configurable service property on the **Advanced** tab; for example, `myCollectorService`. If you do not set the Configurable service property, and if a configurable service exists with the same name as the integration server, that configurable service is used instead.
 - b. Optional: Use the **mqsichangeproperties** and **mqsireportproperties** commands to change or view the properties of the configurable service. Alternatively, you can use the IBM Integration Explorer to view or modify a configurable service. For more information about working with configurable services, see Using the IBM Integration Explorer to work with configurable services.


What to do next


The properties for the configurable service are not used by the broker until you restart or redeploy the message flow, or restart the broker.


Related concepts:


 Using the IBM Integration Explorer to work with configurable services
Configurable services are used to define properties that are related to external services on which the integration node (broker) relies. Use the IBM Integration Explorer to view, add, modify and delete configurable services.

Related tasks:


 Viewing configurable services
Configurable services are used to define properties that are related to external services on which the integration node (broker) relies. Use the IBM Integration Explorer to view the properties of configurable services defined on your integration node.


 Modifying an IBM defined configurable service
You can create a new configurable service for an external service on which the broker relies, by modifying predefined configurable services provided by IBM. Use the IBM Integration Explorer to view and modify existing configurable services.


 Modifying a configurable service
Use the IBM Integration Explorer to view and modify existing configurable services.


 Deleting a configurable service
Use the IBM Integration Explorer to delete custom configurable services.


Related reference:

 Collector node
Use the Collector node to create message collections based on rules that you configure in the node.

 Configurable services properties
The supplied configurable services, and the configurable services that you create, are defined by their names and properties. You can use the supplied services.

 **mqsicreateconfigurableservice** command
Use the **mqsicreateconfigurableservice** command to create an object name for a broker external resource.

 **mqsichangeproperties** command
Use the **mqsichangeproperties** command to modify broker properties and properties of broker resources.

 **mqsireportproperties** command
Use the **mqsireportproperties** command to display properties that relate to a broker, an integration server, or a configurable service.

Configuring the storage of events for Resequencing nodes:

You can use a Resequencing configurable service to control the storage of events for Resequencing nodes.

About this task

By default, the storage queues used by all Resequencing nodes are:

- SYSTEM.BROKER.EDA.EVENTS
- SYSTEM.BROKER.EDA.COLLECTIONS

These queues are also used by the Collector node.

However, you can control the queues that are used by different Resequencing nodes by creating alternative queues that contain a *QueuePrefix* variable, and by using a Resequencing configurable service to specify the names of those queues for storing events.

Follow these steps to specify the queues that are used to store event states, and to set the timeout and the start and end of the sequence:

Procedure

1. Create the storage queues to be used by the Resequencing node. The following queues are required:

- SYSTEM.BROKER.EDA.*QueuePrefix*.EVENTS
- SYSTEM.BROKER.EDA.*QueuePrefix*.COLLECTIONS

The *QueuePrefix* variable can contain any characters that are valid in a WebSphere MQ queue name, but must be no longer than eight characters and must not begin or end with a period (.). For example, SET1 and SET.1 are valid queue prefixes, but .SET1 and SET1. are invalid.

If you do not create the storage queues, IBM Integration Bus creates the set of queues when the node is deployed; these queues are based on the default queues. If the queues cannot be created, the message flow is not deployed.

2. Use the **mqsicreateconfigurable-service** command to create a Resequencing configurable service. You can create a configurable service to be used with either a specific sequence or with all sequences in an integration server.
 - a. If you are creating a configurable service to be used with a specific sequence, ensure that the name of the configurable service is the same as the name that you specify in the Configurable service property on the Resequencing node. If you are creating a configurable service to be used with all sequences in the integration server, ensure that the configurable service has the same name as the integration server.
 - b. Set the **Queue prefix** property to the required value.
 - c. Optional: Set the **Missing message timeout**, **Start of sequence**, and **End of sequence** properties.

For example, create a Resequencing configurable service called `myResequencingService`, which uses queues prefixed with `SYSTEM.BROKER.EDA.SET1`, with a missing message timeout of 60 seconds, and which waits five seconds before determining the start and end numbers in a sequence:

```
mqsicreateconfigurable-service MYBROKER -c Resequencing -o myResequencingService  
-n queuePrefix,missingMessageTimeoutSeconds,startSequenceSeconds,endSequenceSeconds -v SET1,60,5,5
```

You can use the **mqsdeleteconfigurable-service** command to delete the Resequencing configurable service. However, the storage queues are not deleted automatically when the configurable service is deleted, so you must delete them separately. For more information, see Configurable services properties


3. In the Resequencing node:

- a. If the configurable service is to be used for a specific sequence, specify the name of the configurable service on the **Advanced** tab; for example, myResequenceService. If you do not set the Configurable service property, and if a configurable service exists with the same name as the integration server, that configurable service is used instead.
- b. Optional: Use the **mqsichangeproperties** and **mqsireportproperties** commands to change or view the properties of the configurable service. Alternatively, you can use the IBM Integration Explorer to view or modify a configurable service. For more information about working with configurable services, see Using the IBM Integration Explorer to work with configurable services.


What to do next


The properties for the configurable service are not used by the broker until you restart or redeploy the message flow, or restart the broker.


Related concepts:


 Using the IBM Integration Explorer to work with configurable services
Configurable services are used to define properties that are related to external services on which the integration node (broker) relies. Use the IBM Integration Explorer to view, add, modify and delete configurable services.


Related tasks:

 Viewing configurable services
Configurable services are used to define properties that are related to external services on which the integration node (broker) relies. Use the IBM Integration Explorer to view the properties of configurable services defined on your integration node.


 Modifying an IBM defined configurable service
You can create a new configurable service for an external service on which the broker relies, by modifying predefined configurable services provided by IBM. Use the IBM Integration Explorer to view and modify existing configurable services.


 Modifying a configurable service
Use the IBM Integration Explorer to view and modify existing configurable services.

 Deleting a configurable service
Use the IBM Integration Explorer to delete custom configurable services.

 Developing integration solutions
IBM Integration Bus provides a flexible environment in which you can develop integration solutions to transform, enrich, route, and process your business messages and data. You can integrate client applications that use different protocols and message formats.

Related reference:

 Resequence node
Use the Resequence node to control the sequence in which a group (or groups) of incoming messages are propagated in a message flow.

 Configurable services properties
The supplied configurable services, and the configurable services that you create,

are defined by their names and properties. You can use the supplied services.

mqsicreateconfigurable-service command

Use the **mqsicreateconfigurable-service** command to create an object name for a broker external resource.

mqsichangeproperties command

Use the **mqsichangeproperties** command to modify broker properties and properties of broker resources.

mqsireportproperties command

Use the **mqsireportproperties** command to display properties that relate to a broker, an integration server, or a configurable service.

Configuring the storage of events for timeout nodes:

You can use a Timer configurable service to control the storage of events for TimeoutNotification and TimeoutControl nodes.

About this task

By default, the storage queue used by all timeout nodes is the SYSTEM.BROKER.TIMEOUT.QUEUE.

However, you can control the queues that are used by different timeout nodes by creating alternative queues that contain a *QueuePrefix* variable, and by using a Timer configurable service to specify the names of those queues for storing events.

Follow these steps to specify the queue that is used to store event states:

Procedure

1. Create the storage queue to be used by the timeout nodes. The following queue is required:

- SYSTEM.BROKER.TIMEOUT.*QueuePrefix*.QUEUE

The *QueuePrefix* variable can contain any characters that are valid in a WebSphere MQ queue name, but must be no longer than eight characters and must not begin or end with a period (.). For example, SET1 and SET.1 are valid queue prefixes, but .SET1 and SET1. are invalid.

If you do not create the storage queue, IBM Integration Bus creates the queue when the node is deployed; this queue is based on the default queue. If the queue cannot be created, the message flow is not deployed.

2. Use the **mqsicreateconfigurable-service** command to create a Timer configurable service. You can create a configurable service to be used with either specific timeout requests or with all timeout requests in an integration server.
 - a. If the configurable service is to be used with specific timeout requests, create the configurable service with the same name as the Unique identifier property on the TimeoutNotification and TimeoutControl nodes. If the configurable service is to be used with all timeout requests in an integration server, create the configurable service with the same name as the integration server.
 - b. Set the **Queue prefix** property to the required value.

For example, create a Timer configurable service that uses a queue prefixed with SYSTEM.BROKER.TIMEOUT.SET1:

```
mqsicreateconfigurableservice IB9NODE -c Timer -o myTimer  
-n queuePrefix -v SET1
```


You can use the **mqsdeleteconfigurable**service command to delete the Timer configurable service. However, the storage queue is not deleted automatically when the configurable service is deleted, so you must delete it separately. For more information, see Configurable services properties.


3. In the TimeoutNotification and TimeoutControl nodes:
 - a. Ensure that the name of the Timer configurable service is the same as the name specified in the Unique Identifier property on the **Basic** tab; for example, myTimer. If there is no Timer configurable service with the same name as the Unique Identifier, and if there is a configurable service with the same name as the integration server, that configurable service is used instead.
 - b. Optional: Use the **mqschange**properties and **mqsireport**properties commands to change or view the properties of the configurable service. Alternatively, you can use the IBM Integration Explorer to view or modify a configurable service. For more information about working with configurable services, see Using the IBM Integration Explorer to work with configurable services.


What to do next

The properties for the configurable service are not used by the broker until you restart or redeploy the message flow, or restart the broker.


Related concepts:


 Using the IBM Integration Explorer to work with configurable services
Configurable services are used to define properties that are related to external services on which the integration node (broker) relies. Use the IBM Integration Explorer to view, add, modify and delete configurable services.


 Configuring timeout flows
Use the TimeoutControl and TimeoutNotification nodes in message flows to process timeout requests or to generate timeout notifications at specified intervals.


 Sending timeout request messages
To set a controlled timeout, send a message with a set of elements with well known names to a TimeoutControl node. These elements control the properties of the timeout to be created or deleted.

Related tasks:


 Viewing configurable services
Configurable services are used to define properties that are related to external services on which the integration node (broker) relies. Use the IBM Integration Explorer to view the properties of configurable services defined on your integration node.


 Modifying an IBM defined configurable service
You can create a new configurable service for an external service on which the broker relies, by modifying predefined configurable services provided by IBM. Use the IBM Integration Explorer to view and modify existing configurable services.


 Modifying a configurable service
Use the IBM Integration Explorer to view and modify existing configurable services.


 Deleting a configurable service
Use the IBM Integration Explorer to delete custom configurable services.


Related reference:


 TimeoutNotification node
Use the TimeoutNotification node to manage timeout-dependent message flows.

 TimeoutControl node
Use the TimeoutControl node to process an input message that contains a timeout request.

 Configurable services properties
The supplied configurable services, and the configurable services that you create, are defined by their names and properties. You can use the supplied services.

 **mqsicreateconfigurable** command
Use the **mqsicreateconfigurable** command to create an object name for a broker external resource.

 **mqsichangeproperties** command
Use the **mqsichangeproperties** command to modify broker properties and properties of broker resources.

 **mqsireportproperties** command
Use the **mqsireportproperties** command to display properties that relate to a broker, an integration server, or a configurable service.

Analyzing resource performance

You can collect statistics to assess the performance of resources used by integration servers.

Before you begin

Before you start:

Read the concept topic about resource statistics.

About this task

- “Starting resource statistics collection” on page 532
- “Stopping resource statistics collection” on page 533
- “Viewing status of resource statistics collection” on page 534
- “Starting resource statistics collection in the IBM Integration Explorer” on page 535
- “Stopping resource statistics collection in the IBM Integration Explorer” on page 536
- “Viewing resource statistics data in the IBM Integration Explorer” on page 537
- “Viewing the status of resource statistics collection in the IBM Integration Explorer” on page 541

You can also work with resource statistics from CMP applications. See “Working with resource statistics in a CMP application” on page 103 for further details.

Related concepts:

“Resource statistics”

Resource statistics are collected by a broker to record performance and operating details of resources that are used by integration servers.

Related reference:



Resource statistics data

Learn about the measurements for which data is returned when you activate resource statistics collection.



mqsichangeresourcestats command

Use the **mqsichangeresourcestats** command to control statistics gathering for resources in the broker.



mqsireportresourcestats command

Use the **mqsireportresourcestats** command to display current statistics gathering options for resources in the broker.

Resource statistics

Resource statistics are collected by a broker to record performance and operating details of resources that are used by integration servers.

As a system administrator, you can use the resource statistics to ensure that your systems are using the available resources in the most efficient manner. By monitoring systems and analyzing statistical trends, you can keep system resource usage within boundaries that you consider acceptable, and help to pre-empt situations where system resources are overstretched and might become unavailable. Analysis of the data returned potentially requires specialist skills and knowledge of each resource type.

If you detect that system resources are under pressure, you can examine the statistics collected by the broker to assess whether the cause of the concern is the use of those resources by processes in IBM Integration Bus.

You must activate statistics collection; by default, collection is not active. If you activate statistics, you might experience a minor degradation in operating performance of the broker or brokers for which you are collecting data. You can collect data on one or more integration servers, as well as all integration servers on a broker, so that you can limit the statistics gathering activity if appropriate.

Resource statistics complement the accounting and statistics data that you can collect on message flows, which are also available in the IBM Integration Explorer; for details, see “Analyzing message flow performance” on page 469.

To start, stop, or check the status of resource statistics collection, use one or more of the following options:

- The IBM Integration Explorer
- The **mqsichangeresourcestats** and **mqsireportresourcestats** commands
- A CMP application

To view the output that is generated by statistics collection, use one or more of the following options:

- The IBM Integration Explorer; numeric data and graphs are displayed for each integration server for which you have activated statistics collection.
- An application that subscribes to a publication message, that is published by the broker every 20 seconds. The message contains the data collected for each

integration server for which you have activated statistics collection. The published message is available in XML format and in JSON format.

The topic for each message has the following structure:

- For XML format:

```
$SYS/Broker/broker_name/ResourceStatistics/integration_server_name
```

- For JSON format:

```
$SYS/Broker/broker_name/Statistics/JSON/Resource/integration_server_name
```

You can set up subscriptions for a specific integration server on a specific broker.

For example:

- For XML format:

```
$SYS/Broker/IB9NODE/ResourceStatistics/default
```

- For JSON format:

```
$SYS/Broker/IB9NODE/Statistics/JSON/Resource/default
```

You can also use wildcards in the subscriptions to broaden the scope of what is returned. For example, to subscribe to reports for all integration servers on all brokers, use the following values:

- For XML format:

```
$SYS/Broker/+/ResourceStatistics/#
```

- For JSON format:

```
$SYS/Broker/+/Statistics/JSON/Resource/#
```

For details of all the statistics that are reported for each resource manager, and the publication content, see Resource statistics data.

Related concepts:

 [IBM Integration Explorer](#)

The IBM Integration Explorer is a graphical user interface based on the Eclipse platform for administering your integration nodes (brokers).

Related tasks:

[“Analyzing resource performance” on page 529](#)

You can collect statistics to assess the performance of resources used by integration servers.


[“Subscribing to statistics reports” on page 542](#)

You can subscribe to topics that return statistics about the operation of your message flows and resource managers.

[“Working with resource statistics in a CMP application” on page 103](#)

Start, stop, and review status of resource statistics collection in your CMP applications.

Related reference:

 [Resource statistics data](#)

Learn about the measurements for which data is returned when you activate resource statistics collection.

 [Special characters in topics](#)

A topic can contain any character in the Unicode character set, but some characters have a special meaning.

 [mqsichangeresourcestats command](#)

Use the `mqsichangeresourcestats` command to control statistics gathering for resources in the broker.

mqsireportresourcestats command

Use the **mqsireportresourcestats** command to display current statistics gathering options for resources in the broker.

Starting resource statistics collection

Use the **mqsichangeresourcestats** command to start collecting resource statistics.

Before you begin

Before you start:

- Ensure that you have created a message flow; for more information, see [Creating a message flow](#).
- Ensure that you have deployed a broker archive (BAR) file; for more information, see [Deploying a broker archive file](#).
- Read the concept topic, “Resource statistics” on page 530.

About this task

Use resource statistics data to monitor the performance and resource usage of your integration servers. You can start collecting data for active integration servers at any time.

If you prefer, you can start collecting resource statistics by using the IBM Integration Explorer; see “Starting resource statistics collection in the IBM Integration Explorer” on page 535.

To start resource statistics collection:

Procedure

1. If your broker is running on Linux, UNIX, or Windows systems, set up the correct command environment. For details of how to complete this task, see [Setting up a command environment](#).
2. Decide whether you want to collect statistics for a specific integration server, or for all integration servers on the broker.
3. Run the **mqsichangeresourcestats** command with the appropriate parameters. For example, to start collecting resource statistics for the default integration server for BrokerA, enter the following command:

```
mqsichangeresourcestats BrokerA -c active -e default
```

For further examples, see **mqsichangeresourcestats** command.

Related concepts:

“Resource statistics” on page 530

Resource statistics are collected by a broker to record performance and operating details of resources that are used by integration servers.

Related tasks:

“Stopping resource statistics collection” on page 533

Use the **mqsichangeresourcestats** command to stop collecting resource statistics.

“Viewing status of resource statistics collection” on page 534

Use the **mqsireportresourcestats** command to view the status of resource statistics collection.

Related reference:



Resource statistics data

Learn about the measurements for which data is returned when you activate resource statistics collection.



mqsichangeresourcestats command

Use the **mqsichangeresourcestats** command to control statistics gathering for resources in the broker.

Stopping resource statistics collection

Use the **mqsichangeresourcestats** command to stop collecting resource statistics.

Before you begin

Before you start:

- Read the concept topic about resource statistics.
- Start integration server resource statistics collection.

About this task

You can stop collecting data for an active integration server at any time.

If you prefer, you can stop collecting resource statistics by using the IBM Integration Explorer; see “Stopping resource statistics collection in the IBM Integration Explorer” on page 536.

To stop resource statistics collection:

Procedure

1. If your broker is running on Linux, UNIX, or Windows systems, set up the correct command environment. For details of how to complete this task, see *Setting up a command environment*.
2. Decide whether you want to stop collection for a specific integration server, or for all integration servers on the broker. The way in which you started collection is not important; you can stop collection for one integration server, even if you started it for all integration servers. You can also stop collection for all integration servers, even if you started only one, or each one separately.
3. Run the **mqsichangeresourcestats** command with the appropriate parameters.

For example, to stop collecting resource statistics for the default integration server for BrokerA, enter:

```
mqsichangeresourcestats BrokerA -c inactive -e default
```

See the **mqsichangeresourcestats** command for further examples.

Related concepts:

“Resource statistics” on page 530

Resource statistics are collected by a broker to record performance and operating details of resources that are used by integration servers.

Related tasks:

“Starting resource statistics collection” on page 532

Use the **mqsichangeresourcestats** command to start collecting resource statistics.

“Viewing status of resource statistics collection” on page 534

Use the **mqsireportresourcestats** command to view the status of resource statistics collection.

Related reference:



Resource statistics data

Learn about the measurements for which data is returned when you activate resource statistics collection.



`mqsichangeresourcestats` command

Use the `mqsichangeresourcestats` command to control statistics gathering for resources in the broker.

Viewing status of resource statistics collection

Use the `mqsireportresourcestats` command to view the status of resource statistics collection.

Before you begin

Before you start:

- Read the concept topic about resource statistics.

About this task

The command indicates whether resource statistics collection is active or inactive for each integration server that you have specified on the command.

If you prefer, you can view resource statistics collection status by using the IBM Integration Explorer; see “Viewing the status of resource statistics collection in the IBM Integration Explorer” on page 541.

To view resource statistics collection status:

Procedure

1. If your broker is running on Linux, UNIX, or Windows systems, set up the correct command environment. For details of how to complete this task, see [Setting up a command environment](#).
2. Decide whether you want to view status for a specific integration server, or for all integration servers on the broker.
3. Run the `mqsireportresourcestats` command with the appropriate parameters.
For example, to view status for the default integration server on BrokerA, enter:

```
mqsireportresourcestats BrokerA -e default
```

See the `mqsireportresourcestats` command for further examples.

Related concepts:

“Resource statistics” on page 530

Resource statistics are collected by a broker to record performance and operating details of resources that are used by integration servers.

Related tasks:

“Starting resource statistics collection” on page 532

Use the `mqsichangeresourcestats` command to start collecting resource statistics.

“Stopping resource statistics collection” on page 533

Use the `mqsichangeresourcestats` command to stop collecting resource statistics.

Related reference:



Resource statistics data

Learn about the measurements for which data is returned when you activate resource statistics collection.

mqsichangeresourcestats command

Use the **mqsichangeresourcestats** command to control statistics gathering for resources in the broker.

mqsireportresourcestats command

Use the **mqsireportresourcestats** command to display current statistics gathering options for resources in the broker.

Starting resource statistics collection in the IBM Integration Explorer

Use the IBM Integration Explorer to start collecting resource statistics data for your integration servers. You can then view the data in the Broker Resources and Broker Resources Graph views.

Before you begin

Before you start:

- Ensure that you have created a message flow; for more information, see [Creating a message flow](#).
- Ensure that you have deployed a broker archive (BAR) file; for more information, see [Deploying a broker archive file](#).
- Read the concept topic, “Resource statistics” on page 530.

About this task

Use the resource statistics data to monitor the performance and resource usage in your integration servers. You can start collecting data for active integration servers at any time.

If you prefer, you can start resource statistics collection by using the **mqsichangeresourcestats** command; for more information, see “Starting resource statistics collection” on page 532.

To start resource statistics collection in the IBM Integration Explorer, complete the following steps.

Procedure

1. In the WebSphere MQ Explorer - Navigator view, expand the Integration Nodes folder, then expand the folder of the integration node with which you are working.
2. Right-click the integration server for which you want to collect statistics. If you want statistics for several integration servers on this integration node, select more than one by using standard operating system interfaces. For example, on Windows systems, hold down the **Ctrl** key and select each integration server before you right-click to open the menu.
3. Click **Statistics > Start Resource Statistics**. A message is sent to the integration node to start collecting data for the selected integration servers. The property Resource Statistics Active is updated in the properties view of each affected integration server to indicate that data collection is now active.
A warning message is displayed in the integration server properties window to warn you that performance might be affected by your action.
4. To view statistics, click **Window > Show View > Resource Statistics** to open the Broker Resources and Broker Resources Graph views. If you are displaying

statistics for this integration server for the first time, the views might be empty until the first data is received. Update messages are sent every 20 seconds, and the views refresh automatically.

The Broker Resources and Broker Resources Graph views are displayed together. If you close one of the views, the other view is also closed.

Related concepts:



IBM Integration Explorer

The IBM Integration Explorer is a graphical user interface based on the Eclipse platform for administering your integration nodes (brokers).

“Resource statistics” on page 530

Resource statistics are collected by a broker to record performance and operating details of resources that are used by integration servers.

Related tasks:

“Stopping resource statistics collection in the IBM Integration Explorer”

Use the IBM Integration Explorer to stop collecting resource statistics data for your integration servers.

“Viewing resource statistics data in the IBM Integration Explorer” on page 537

Use the IBM Integration Explorer to view resource statistics data for your integration servers in the Broker Resources and Broker Resources Graph views.

Related reference:



Resource statistics data

Learn about the measurements for which data is returned when you activate resource statistics collection.

Stopping resource statistics collection in the IBM Integration Explorer

Use the IBM Integration Explorer to stop collecting resource statistics data for your integration servers.

Before you begin

Before you start:

- Read the concept topic about resource statistics.
- Start resource statistics collection

About this task

You can stop collecting data for active integration servers at any time.

If you prefer, you can stop collecting resource statistics by using the `mqsichangeresourcestats` command; see “Stopping resource statistics collection” on page 533.

To stop collecting resource statistics in the IBM Integration Explorer:

Procedure

1. In the **WebSphere MQ Explorer - Navigator** view, expand the Integration Nodes folder.
2. Right-click the integration node or integration server for which you want to stop statistics collection. If you want to stop statistics collection for several integration servers, you can select more than one by using standard operating

system interfaces; for example, on Windows systems, hold down the Ctrl key and select each integration server before you right-click to open the menu.

3. Click **Statistics > Stop Resource Statistics**. A message is sent to the integration node to stop collecting resource data for the selected integration servers. The status of data collection is updated in the properties view for each affected integration server.

Results

If you click **Window > Show View > Resource Statistics** to open the Broker Resources and Broker Resources Graph views when statistics collection is not active, the data that is displayed represents the last update message received by the IBM Integration Explorer. If you have never started statistics collection for this integration server, the views are displayed but contain no data.

Related concepts:



IBM Integration Explorer

The IBM Integration Explorer is a graphical user interface based on the Eclipse platform for administering your integration nodes (brokers).

“Resource statistics” on page 530

Resource statistics are collected by a broker to record performance and operating details of resources that are used by integration servers.

Related tasks:

“Starting resource statistics collection in the IBM Integration Explorer” on page 535
Use the IBM Integration Explorer to start collecting resource statistics data for your integration servers. You can then view the data in the Broker Resources and Broker Resources Graph views.

“Viewing resource statistics data in the IBM Integration Explorer”

Use the IBM Integration Explorer to view resource statistics data for your integration servers in the Broker Resources and Broker Resources Graph views.

Related reference:



Resource statistics data

Learn about the measurements for which data is returned when you activate resource statistics collection.

Viewing resource statistics data in the IBM Integration Explorer

Use the IBM Integration Explorer to view resource statistics data for your integration servers in the Broker Resources and Broker Resources Graph views.

Before you begin

Before you start:

- Read the concept topic, “Resource statistics” on page 530.
- Start resource statistics collection: “Starting resource statistics collection in the IBM Integration Explorer” on page 535

About this task

You can also view resource statistics collection by subscribing to the topic on which statistics are published. For further details, see “Subscribing to statistics reports” on page 542.

To view resource statistics in the IBM Integration Explorer, complete the following steps.

Procedure

1. In the **WebSphere MQ Explorer - Navigator** view, expand the Integration Nodes folder.
2. To open the Resource Statistics and Resource Statistics Graph views, click **Window > Show View > Resource Statistics**. These two views are displayed together. If you close one of the views, the other view is also closed.
3. Use the information in these views to review the use of resources for which statistics are available.

The following examples demonstrate the types of question that can be answered by collecting resource statistics. This list is not exhaustive, and does include all resource types. For a full list of resource types, and the type of information that is collected for each one, see Resource statistics data

JVM statistics

How much memory is the JVM using?

Many tools that are specific to an operating system give you the total memory that is used by the integration server, but they do not show you how that memory is divided between Java processing and other processing in the integration server. By looking at the field `CommittedMemoryInMB`, you can see how much memory is currently allocated to the JVM. Then look at the field `MaxMemoryInMB` to see the maximum amount of memory that can be allocated.

How often is garbage collection done? Is it affecting the performance of the integration server?

To see how often the JVM is doing garbage collection, check the `CumulativeNumberOfGCCollections` field to see if the rate of collections is increasing. Garbage collection is a normal process, and is therefore expected to some degree. However, excessive garbage collection can affect performance.

To see if current garbage collection is excessive, monitor the `CumulativeGCTimeInSeconds` value. If this value is increasing by more than 2 seconds in each 20-second statistics interval, try increasing the JVM maximum heap size for your integration server by using the `mqsichangeproperties` command. You might also want to inspect all the Java user-defined nodes and `JavaCompute` nodes that are included in your deployed message flows, to ensure that they do not create and delete many objects that could be reused; frequent deletions can contribute to excessive garbage collection.

Do I need to change the minimum or maximum heap sizes?

- If the `CumulativeGCTimeInSeconds` value is increasing by more than 2 seconds in each 20 second statistics interval, increase the maximum heap size to reduce this increase.
- If the `UsedMemoryInMB` value is never close to the `InitialMemoryInMB` value, you might have allocated more memory for the heap than is required. Therefore, reduce the JVM minimum heap size value for the integration server to a value that is closer to the `UsedMemoryInMB` value.

Change these values gradually, and check the results to find the optimum settings for your environment.

Parsers

Are message parsers using more memory than expected?

A message flow parses input messages and can create many output messages. These messages may have large bit streams or large message trees. The parsers created to perform this message processing might consume a large amount of memory. Use the Parsers statistics to determine if message flow parsers are using more memory than expected. If so, consider deploying such flows into separate integration servers or improving ESQL or Java plugin API processing to efficiently handle large messages or transformations.

Is message parsing or writing failing frequently for a particular message flow?

If a message flow receives or attempts to write an invalid message, it is likely that this will be rejected by a parser. Use the message parsers statistics to see if a message flow is rejecting a large amount of input or output messages compared with successful processing.

Outbound sockets

Are the nodes reusing outbound sockets?

Creating outbound sockets can be an expensive operation, and the number of sockets available on a computer is a finite resource. Therefore, increasing socket reuse can enhance performance. If the workload is continuous and consistent, the TotalSockets value indicates an initial period of activity, which then reduces when the integration server starts to reuse sockets.

A steady increase in the TotalSockets value over time is expected because sockets are closed after a period of inactivity, or when they have been used many times.

If the TotalSockets value increases significantly over time, this trend might indicate that outbound sockets are not being reused.

If your message flows include HTTPRequest nodes, check that you have set the keepalive property Enable HTTP/1.1 keepalive.

Check also whether the endpoint that is called uses keepalive sockets.

Which endpoints are most used?

The values TotalMessages indicates how busy each endpoint is. The value in the summary record tells you how much activity occurred across the whole integration server.

How large are sent and received messages?

The values of the SentMessageSize_* and ReceivedMessageSize_* fields give a profile of the message sizes flowing to and from each endpoint.

JDBC connection pools

Do I need to change the size of the connection pool?

If the statistics show that the count of callers waiting for connections is high, and the wait time is increasing, consider increasing the size of the pool using the `MaxConnectionPoolSize` property for the `JDBCProvider` configurable service.

Alternatively, try reducing the number of additional instances configured for the message flow.

TCPIPClientNodes

Are the nodes reusing outbound sockets?

Creating outbound sockets can be an expensive operation, and the number of sockets available on a computer is a finite resource. Therefore, increasing socket reuse can enhance performance. If the workload is continuous and consistent, the `TotalSockets` value indicates an initial period of activity, which then reduces when the integration server starts to reuse sockets.

A steady increase in the `TotalSockets` value over time is expected because sockets are closed after a period of inactivity, or when they have been used many times.

If the `TotalSockets` value increases significantly over time, this trend might indicate that outbound sockets are not being reused.

If your message flows include `HTTPRequest` nodes, check that you have set the keepalive property `Enable HTTP/1.1 keepalive`.

Check also whether the endpoint that is called uses keepalive sockets.

How does the information that I see in IBM Integration Explorer relate to my TCP/IP flows?

An entry is displayed for each configurable service, not for each flow.

Related concepts:

IBM Integration Explorer

The IBM Integration Explorer is a graphical user interface based on the Eclipse platform for administering your integration nodes (brokers).

“Resource statistics” on page 530

Resource statistics are collected by a broker to record performance and operating details of resources that are used by integration servers.

Related tasks:

“Starting resource statistics collection in the IBM Integration Explorer” on page 535
Use the IBM Integration Explorer to start collecting resource statistics data for your integration servers. You can then view the data in the Broker Resources and Broker Resources Graph views.

“Stopping resource statistics collection in the IBM Integration Explorer” on page 536

Use the IBM Integration Explorer to stop collecting resource statistics data for your integration servers.

“Setting the JVM heap size” on page 544

When you start an integration server, it creates a Java virtual machine (JVM) for executing a Java user-defined node.

“Subscribing to statistics reports” on page 542

You can subscribe to topics that return statistics about the operation of your message flows and resource managers.

Related reference:



Resource statistics data

Learn about the measurements for which data is returned when you activate resource statistics collection.



mqsichangeproperties command

Use the **mqsichangeproperties** command to modify broker properties and properties of broker resources.

Viewing the status of resource statistics collection in the IBM Integration Explorer

Use the IBM Integration Explorer to view the status of resource statistics collection in the Broker Resources view.

Before you begin

Before you start:

- Read the concept topic about resource statistics.

About this task

If you prefer, you can view resource statistics collection status by using the **mqsireportresourcestats** command; see “Viewing status of resource statistics collection” on page 534.

To view the status of resource statistics collection in the IBM Integration Explorer:

Procedure

1. In the **WebSphere MQ Explorer - Navigator** view, expand the Integration Nodes folder.
2. Click the integration server for which you want to view statistics collection status. You can view status for only one integration server at a time.

In the **WebSphere MQ Explorer - Content** view, the **Resource Statistics Active** field indicates whether statistics collection is active. If the field contains the value None, statistics collection is inactive. If collection is active, the field contains the value CICS, CORBA, FTEAgent, JVM, Parsers, Security, Sockets or JDBC Connection Pooling.

Related concepts:



IBM Integration Explorer

The IBM Integration Explorer is a graphical user interface based on the Eclipse platform for administering your integration nodes (brokers).

“Resource statistics” on page 530

Resource statistics are collected by a broker to record performance and operating details of resources that are used by integration servers.

Related tasks:

“Starting resource statistics collection in the IBM Integration Explorer” on page 535
Use the IBM Integration Explorer to start collecting resource statistics data for your integration servers. You can then view the data in the Broker Resources and Broker Resources Graph views.

“Stopping resource statistics collection in the IBM Integration Explorer” on page 536
Use the IBM Integration Explorer to stop collecting resource statistics data for your integration servers.

Related reference:



Resource statistics data

Learn about the measurements for which data is returned when you activate resource statistics collection.

Subscribing to statistics reports

You can subscribe to topics that return statistics about the operation of your message flows and resource managers.

About this task

Message flow performance

If you enable message flow accounting and statistics collection for a broker, you can subscribe to the messages that the broker publishes on the following topic:

```
$SYS/Broker/brokerName/StatisticsAccounting/recordType/integrationServerLabel/messageFlowLabel
```

where *broker_name* is the name of the broker, *recordType* is the type of record (Snapshot or Archive), *integrationServerLabel* is the name of the integration server that you created on that broker, and *messageFlowLabel* is the name of the message flow that you deployed to the integration server.

These messages contain statistics reports and are published at a regular interval, which you control by setting the `statsInterval` property of the broker. Each publication is a JMS TextMessage that contains the statistics report in XML format.

Note: If you need to revert to using a JMS BytesMessage format, this can be achieved by setting the environment variable `MQSI_STATS_MQSTR=false`.

Resource performance

If you enable resource statistics collection for one or more integration servers on a broker, you can subscribe to the messages that the broker publishes at 20-second intervals on the following topic:

```
$SYS/Broker/brokerName/ResourceStatistics/integrationServerLabel
```

For more information about how to interpret the resource statistics that are included in the publication, see “Viewing resource statistics data in the IBM Integration Explorer” on page 537.

Using wildcards in subscriptions

Procedure

You can use wild cards when you subscribe to statistics reports. For example, to receive message flow statistics reports for all brokers and all integration servers, subscribe to the following topic:

```
$SYS/Broker+/StatisticsAccounting/#
```

To receive integration server resource statistics reports for all brokers and all integration servers, subscribe to the following topic:

```
$SYS/Broker+/ResourceStatistics/#
```

For further details about how you can use wildcards, see [Special characters in topics](#).

Results

Subscribers receive statistics reports only from those brokers that are enabled to produce statistics.

Related tasks:

[“Analyzing message flow performance” on page 469](#)

You can configure your broker to collect statistics about the operation and behavior of your message flows. Use this information to assess the performance of your message flows.

[“Analyzing resource performance” on page 529](#)

You can collect statistics to assess the performance of resources used by integration servers.

[“Viewing resource statistics data in the IBM Integration Explorer” on page 537](#)

Use the IBM Integration Explorer to view resource statistics data for your integration servers in the Broker Resources and Broker Resources Graph views.

Related reference:



[Special characters in topics](#)

A topic can contain any character in the Unicode character set, but some characters have a special meaning.

Tuning the broker

You can complete several tasks that enable you to tune different aspects of the broker performance.

Before you begin

Before you start:

Ensure that the following requirements are met:

- Your user ID has the correct authorizations to perform the task. Refer to [Security requirements for administrative tasks](#).

About this task

Select the tasks that are relevant to your enterprise:

Procedure

- “Setting the JVM heap size”
- “Increasing the stack size on Windows, Linux, and UNIX systems” on page 545
- “Increasing the stack size on z/OS” on page 546
- “Tuning the HEAP settings on z/OS” on page 547
- “Setting configuration timeout values” on page 548

Related concepts:



The IBM Integration Bus environment

An integration node (broker) is a set of execution processes that hosts one or more message flows to route, transform, and enrich in flight messages.

Related tasks:



Configuring brokers

Create and configure the brokers that you want on the operating system of your choice.

“Analyzing message flow performance” on page 469

You can configure your broker to collect statistics about the operation and behavior of your message flows. Use this information to assess the performance of your message flows.

“Tuning message flow performance” on page 510

You can configure your message flows to enhance their performance in terms of resource usage, message flow throughput, and response times.

“Performance” on page 450

You can use message flow design and coding techniques to optimize the performance of your message flows. You can then analyze statistical information for your message flows and modify aspects of your broker configuration to tune the performance of your brokers and message flows.

“Analyzing resource performance” on page 529

You can collect statistics to assess the performance of resources used by integration servers.

“Troubleshooting performance problems” on page 560

Follow this guidance to resolve common problems with IBM Integration Bus performance.

Related reference:



Security requirements for administrative tasks

You can configure access permissions to govern which users and groups can manipulate objects in the broker network. Security requirements for administrative tasks depend on the platform that you use.

Setting the JVM heap size

When you start an integration server, it creates a Java virtual machine (JVM) for executing a Java user-defined node.

Before you begin

Before you start:

- Check the resource statistics data to monitor resource usage and identify potential problems with performance; for more information, see “Resource statistics” on page 530.

About this task

You can pass parameters to the JVM to set the minimum and maximum heap sizes; the default maximum heap size is 256 MB. To give more capacity to a message flow that is going to process large messages, reduce the minimum JVM heap size to allow the main memory heap to occupy more address space.

Increase the maximum heap size only if you use Java intensively with, for example, user-defined nodes.

Use caution when you set the maximum heap size, because the Java Runtime Environment takes the values for its initial, maximum, and current heap sizes to calculate how frequently it drives garbage collection. A large maximum heap size drives garbage collection less frequently. If garbage collection is driven less frequently, the heap size associated with the integration server continues to grow.

Use the information on JVM parameter values on the **mqsichangeproperties** command to set the heap size that you require.

Related concepts:

“JVM heap sizing” on page 518

The Java virtual machine (JVM) heap is an independent memory allocation that can reduce the capacity of the main memory heap.

“Stack storage” on page 517

Depending on the design of your message flow, you might need to increase the stack size.

Related tasks:

“Optimizing message flow response times” on page 554

You can use different solutions to improve message flow response times.

Related reference:

 Message flows

Use the reference information in this section to develop your message flows and related resources.

 **mqsichangeproperties** command

Use the **mqsichangeproperties** command to modify broker properties and properties of broker resources.

 JVM parameter values

Select the objects and properties associated with the Java Virtual Machine (JVM) that you want to change.

Increasing the stack size on Windows, Linux, and UNIX systems

Increase the stack size on Windows, Linux, and UNIX systems by setting the **MQSI_THREAD_STACK_SIZE** environment variable to an appropriate value.

About this task

When you restart brokers that are running on the system, they use the new value.

The value, in bytes, of **MQSI_THREAD_STACK_SIZE** that you set is used for every thread that is created in a DataFlowEngine process. If the integration server has many message flows assigned to it, and you set a large value for **MQSI_THREAD_STACK_SIZE**, the DataFlowEngine process needs a large amount of storage for the stacks.

Set this environment variable to at least 48 KB. The default values are:

Linux and UNIX

1 MB

AIX 2 MB

z/OS 50 KB

Windows

Determined by the operating system.

Related concepts:

“Stack storage” on page 517

Depending on the design of your message flow, you might need to increase the stack size.

Related tasks:

“Increasing the stack size on z/OS”

Change the stack size on z/OS by altering or adding the Language Environment® (LE) `_CEE_RUNOPTS` environment variable. This can be done for all integration servers defined to a broker, or a specific integration server.

Increasing the stack size on z/OS

Change the stack size on z/OS by altering or adding the Language Environment (LE) `_CEE_RUNOPTS` environment variable. This can be done for all integration servers defined to a broker, or a specific integration server.

Before you begin

Broker components on z/OS are compiled using the XPLINKage (extra performance linkage), which adds optimization to the runtime code. However, if the initial stack size is not large enough, stack extents are used. The initial stack size is 1 MB, and 1 MB is used in each extent. Ensure that you choose a large enough downward stack size because the performance of XPLINK can be adversely affected when stack extents are used.

To determine suitable stack sizes, you can use the Language Environment Report Storage tool.

To use this tool, use the `RPTSTG` option with the `_CEE_RUNOPTS` environment variable to test a message flow. Set this option in the component profile (BIPBPROF for a broker) during the development and test of message flows that are intended for production; for example:

```
export _CEE_RUNOPTS=RPTSTG\ (ON\)
```

You can then override the default values for the stack sizes on z/OS by altering or adding the `_CEE_RUNOPTS` environment variable.

You can do this for all integration servers defined to a broker, or a specific integration server.

About this task

To update the component profile, take the following steps:

Procedure

1. Stop the broker.
2. Make the necessary changes to the profile.
3. Submit BIPGEN to re-create the broker ENVFILE and any integration server specific ENVFILES.
4. Restart the broker.

Results

To update the stack sizes for a specific integration server in a broker, take the following steps:

1. Stop the broker.
2. Make the necessary changes to the integration server specific profile.
3. Submit BIPGEN to re-create the broker ENVFILE and any integration server specific ENVFILES.
4. Restart the broker.

Example

The following example shows how you can change the default stack value of 1 MB to 2 MB:

```
export _CEE_RUNOPTS=THREADSTACK64\ (0N,2M,1M,128M\)
```

What to do next

When you use the **RPTSTG** option, it increases the time that an application takes to run, so use it as an aid to the development of message flows only, and not in your final production environment. When you have determined the correct stack sizes needed, remove this option from the **_CEE_RUNOPTS** environment variable.

XPLINK stacks grow downward in virtual storage while the standard linkage grows upward. To avoid affecting performance by switching between downward stack space and upward stack space during run time, compile user-defined extensions using the XPLINK option where possible. If your message flow uses user-defined extensions that have been compiled with the standard linkage convention, set a suitable value for the upward stack size.

Related concepts:

“Stack storage” on page 517

Depending on the design of your message flow, you might need to increase the stack size.

Related tasks:

“Increasing the stack size on Windows, Linux, and UNIX systems” on page 545
Increase the stack size on Windows, Linux, and UNIX systems by setting the **MQSI_THREAD_STACK_SIZE** environment variable to an appropriate value.

Tuning the HEAP settings on z/OS

HEAP controls the allocation of the initial heap, controls allocations of additional heap increments, and specifies how that storage is managed.

About this task

IBM Integration Bus requests an initial heap storage allocation and subsequent heap increments, the sizes of which depend on the type of process. For example,

the DFE process requests an initial heap storage allocation of 40 Mb, with subsequent heap increments of 5 Mb. RPTOPTS can be used to generate a report of the runtime options in effect for each process.

- KEEP, which is the default value, specifies that storage allocated to HEAP increments is not released when the last segment of the allocated storage is freed.
- FREE specifies that storage allocated to HEAP increments is released when the last segment of the allocated storage is freed.

For performance reasons, IBM Integration Bus takes the default, KEEP. For most message processing scenarios, when storage allocations are less than 5 Mb, it is more efficient to reuse storage that has been freed within the heap increment. With KEEP, the 5 Mb heap increment remains allocated, even if all of the storage segments have been released.

If storage requests frequently exceed 5 Mb, these requests are allocated directly on the heap. When the object is freed, the allocation remains on the heap, and is reused for subsequent storage requests whose size is less than, or equal to, the size of the heap allocation. Over time, the heap allocation is used for different-sized objects, and this can lead to fragmentation, which in turn can result in high storage usage. In these circumstances, consider setting the HEAP runtime environment variable for the Language Environment to use the FREE parameter.

To set HEAP for all integration servers in a broker, change or add the Language Environment `_CEE_RUNOPTS` environment variable in the component profile (BIPBPROF):

1. Stop the broker.
2. Make the necessary changes to the profile.
3. Submit BIPGEN to re-create the ENVFILE and any integration server specific ENVFILES.
4. Restart the broker.

To set HEAP for a specific integration server, change or add the Language Environment `_CEE_RUNOPTS` environment variable in the integration server specific profile (renamed BIPEPROF):

1. Stop the broker.
2. Make the necessary changes to the integration server specific profile.
3. Submit BIPGEN to re-create the broker ENVFILE and any integration server specific ENVFILES.
4. Restart the broker.

For example, you can change the default values of KEEP to FREE in the following line:

```
_CEE_RUNOPTS=HEAP64(40M,5M,FREE,9M,32K,KEEP,4096,4096,FREE)
```

Setting configuration timeout values

Change timeout values that affect configuration tasks in the broker.

Before you begin

Before you start:

Read Packaging and deployment overview to understand the conditions under which these timeout values apply.

About this task

The broker receives configuration requests from the IBM Integration Toolkit, the IBM Integration Explorer, and CMP applications. You can also change its configuration by running several commands, for example, **mqsideploy**.

Several factors affect the time that a broker takes to apply and respond to these requests. These factors include the load on the computer on which the broker is running, network delays between components, and the work that integration servers are performing at the time the request is received. The number of message flows in an integration server, and their complexity, and large message model schema files, might also affect the time taken.

You can change the length of time that a broker can take to perform these actions by using two parameters that you can set on the **mqsicreatebroker** and **mqsichangebroker** commands. The combined default value for these parameters is approximately 6 minutes (360 seconds).

During development and test of message flows and broker configurations, experiment with the values that you set for these timeout parameters to determine appropriate values for your resources.

- **-g** *ConfigurationChangeTimeout*

This value defines the maximum time (in seconds) that is allowed for a user configuration request to be processed, and defaults to 5 minutes (300 seconds). The value is affected by the system load (including processor use), and by the load of each integration server. If the request is not completed in this time, the broker generates warning message BIP2066, but continues to implement the change. The broker records further diagnosis information in the system and event logs.

- **-k** *InternalConfigurationTimeout*

This value defines the maximum time (in seconds) that is allowed for an internal configuration change to be processed and defaults to 1 minute (60 seconds). For example, it defines the length of time that the broker can take to start an integration server before a response is required.

The broker starts an internal process to start an integration server and to make all the message flows active. Part of this initialization is performed serially (one integration server at a time), therefore if the change affects more than one integration server, the time required increases. If an integration server exceeds this timeout value, the broker generates a warning message BIP2080. However, the initialization continues and the integration server is started. The broker records further diagnosis information in the system and event logs.

The sum of the values of *ConfigurationChangeTimeout* and the *InternalConfigurationTimeout* parameters represents the maximum length of time that a broker can take to process a deployed configuration message before it generates a negative response. Check that typical configurations complete successfully within the time that you specified, to minimize warning messages. If you are using commands that take the *timeoutSecs* (-w) parameter, set this parameter to the sum of *ConfigurationChangeTimeout* and *InternalConfigurationTimeout* values (or higher), to ensure that the command waits long enough to report an accurate completion status for a configuration change. If

`-w` is set to a value that is less than the configuration timeouts for the broker, then you might see timeouts for configuration changes that successfully complete.

Look for success messages in the Administration log in the IBM Integration Explorer, or Deployment Log in the IBM Integration Toolkit. When success messages are displayed the deployment has completed. If you start a deployment and record how long it takes for the success messages to appear, you can use this time interval as the basis for setting these timeout values.

If the broker is on a production system, increase the values for both *ConfigurationChangeTimeout* and *InternalConfigurationTimeout* parameters to allow for application messages that are currently being processed by message flows to be completed before the configuration change is applied. Also consider increasing the value if you have merged message flows into fewer integration servers that you are using for testing.

If the broker is on a development or test system, you might want to reduce timeout lengths (in particular, the value of the *ConfigurationChangeTimeout* parameter) to improve perceived response times, and to force a response from a broker that is not showing expected behavior. However, reducing the timeout values decreases the probability of deploying a configuration change successfully.

During broker startup, the *InternalConfigurationTimeout* parameter is automatically extended based on the number of integration servers which a broker contains. At this time, the timeout period reported in the BIP2080 warning message may not match the value configured for the *InternalConfigurationTimeout* parameter.

Related concepts:



The IBM Integration Bus environment

An integration node (broker) is a set of execution processes that hosts one or more message flows to route, transform, and enrich in flight messages.



Packaging and deployment overview

Deployment is the process of transferring data to an integration server on a broker so that it can take effect in the broker. Message flows and associated resources are packaged in broker archive (BAR) files for deployment.

Related tasks:



Creating a broker

You can create brokers on every platform that is supported by IBM Integration Bus. The broker runs as a 64-bit application on all platforms except Linux on x86 and Windows on x86.



Packaging and deploying

Package resources that you create in the IBM Integration Toolkit, such as message flows, and deploy them to integration servers on brokers.



Checking the results of deployment

After you have made a deployment, check that the operation has completed successfully.

Related reference:



`mqsichangebroker` command

Use the `mqsichangebroker` command to change one or more of the configuration parameters of the broker.

mqsicreatebroker command

Use the **mqsicreatebroker** command to create a broker and its associated resources.

mqsideploy command

Use the **mqsideploy** command to make a deployment request to the broker.

Optimizing message flow throughput

Each message flow that you design must provide a complete set of processing for messages received from a particular source. This design might result in very complex message flows that include large numbers of nodes that can cause a performance overhead, and might create potential bottlenecks. You can increase the number of message flows that process your messages to provide the opportunity for parallel processing and therefore improved throughput.

About this task

The operation mode of your broker can affect the number of message flows that you can use; see IBM Integration features and Restrictions that apply in each operation mode for more information.

You can also consider the way in which the actions taken by the message flow are committed, and the order in which messages are processed.

The web user interface enables you to view message flow statistics and accounting data, including the message flow throughput and the CPU and elapsed times for transactions in the flow. See “Viewing accounting and statistics data in the web user interface” on page 499 for more information about how you can access the statistical data.

Consider the following options for optimizing message flow throughput:

Multiple threads processing messages in a single message flow

When you deploy a message flow, the broker automatically starts an instance of the message flow for each input node that it contains. This behavior is the default. If you have a message flow that handles a very large number of messages, you can enable more messages to be processed by allocating more instances of the flow.

You can update the Additional Instances property of the deployed message flow in the BAR file; the broker starts additional copies of the message flow on separate threads, providing parallel processing. This option is the most efficient way of handling this situation if you are not concerned about the order in which messages are processed.

If the message flow receives messages from a WebSphere MQ queue, you can influence the order in which messages are processed by setting the Order Mode property of the MQInput node:

- If you set Order Mode to By User ID, the node ensures that messages from a specific user (identified by the UserIdentifier field in the MQMD) are processed in guaranteed order. A second message from one user is not processed by an instance of the message flow if a previous message from this user is currently being processed by another instance of the message flow.
- If you set Order Mode to By Queue Order, the node processes one message at a time to preserve the order in which the messages are read

from the queue. Therefore, this node behaves as though you have set the Additional Instances property of the message flow to zero.

- If you set Order Mode to User Defined, you can order messages by any message element, by setting an XPath or ESQL expression in the Order field location property. The node ensures that messages with the same value in the order field message element are processed in guaranteed order. A second message with the same value in the order field message element is not processed by an instance of the message flow if a previous message with the same value is currently being processed by another instance of the message flow.

If the field is missing, an exception is raised, and the message is rolled back. NULL and empty values are processed separately, in parallel.

If you set Order Mode to By User ID or User Defined, and the message flow uses transformation nodes, it is advisable to set the Parse Timing to Immediate.

Multiple copies of the message flow in a broker

You can also deploy several copies of the same message flow to different integration servers in the same broker. This option has similar effects to increasing the number of processing threads in a single message flow.

This option also removes the ability to determine the order in which the messages are processed, because, if there is more than one copy of the message flow active in the broker, each copy can be processing a message at the same time, from the same queue. The time taken to process a message might vary, and multiple message flows accessing the same queue might therefore read messages from the input source in a random order. The order of messages produced by the message flows might not correspond to the order of the original messages.

Ensure that the applications that receive message from these message flows can tolerate out-of-order messages. Additionally, ensure that the input nodes in these message flows are suitable for deployment to different processes.

Copies of the message flow in multiple brokers

You can deploy several copies of the same message flow to different brokers. This option requires changes to your configuration, because you must ensure that applications that supply messages to the message flow can put their messages to the right input queue or port. You can often make these changes when you deploy the message flow by setting the message flow's configurable properties.

The scope of the message flow

You might find that, in some circumstances, you can split a single message flow into several different flows to reduce the scope of work that each message flow performs. If you do split your message flow, be aware that it is not possible to run the separate message flows in the same unit of work, and if transactional aspects to your message flow exist (for example, the updating of multiple databases), this option does not provide a suitable solution.

The following two examples show when it might be beneficial to split a message flow:

1. In a message flow that uses a RouteToLabel node, the input queue might increase in size, indicating that work is arriving faster than it is being processed; this might indicate a need for increased processing capacity. You can use another copy of the message flow in a second

integration server, but this option is not appropriate if you want all of the messages to be handled in the order in which they are shown on the queue. You can consider splitting out each branch of the message flow that starts with a Label node by providing an input queue and input node for each branch. This option might be appropriate, because when the message is routed by the RouteToLabel node to the relevant Label node, it has some level of independence from all other messages. You might also need to provide another input queue and input node to complete any common processing that the Label node branches connect to when unique processing has been done.

2. If you have a message flow that processes very large messages that take a considerable time to process, you might be able to:
 - a. Create other copies of the message flow that use a different input queue (you can set this option up in the message flow itself, or you can update this property when you deploy the message flow).
 - b. Set up WebSphere MQ queue aliases to redirect messages from some applications to the alternative queue and message flow.

You can also create a new message flow that replicates the function of the original message flow, but only processes large messages that are immediately passed on to it by the original message flow, that you modified to check the input message size and redirect the large messages.

The frequency of commits

If a message flow receives input messages on a WebSphere MQ queue, you can improve its throughput for some message flow scenarios by modifying its default properties after you have added it to a BAR file. (These options are not available if the input messages are received by other input nodes; commits in those message flows are performed for each message.)

The following properties control the frequency with which the message flow commits transactions:

- **Commit Count.** This property represents the number of messages processed from the input queue per message flow thread, before an MQCMIT is issued.
- **Commit Interval.** This property represents the time interval that elapses before an MQCMIT is started.

Related concepts:

“Message flow statistics and accounting data” on page 471

Message flow statistics and accounting data can be collected to record performance and operating details of one or more message flows.

Related tasks:

“Optimizing message flow response times” on page 554

You can use different solutions to improve message flow response times.

“Viewing message flow accounting and statistics data” on page 496

You can view snapshot accounting and statistics data as it is produced by the broker.

“Tuning message flow performance” on page 510

You can configure your message flows to enhance their performance in terms of resource usage, message flow throughput, and response times.

“Analyzing message flow performance” on page 469

You can configure your broker to collect statistics about the operation and behavior of your message flows. Use this information to assess the performance of your

message flows.



Creating a message flow

Create a message flow to specify how to process messages in the broker. You can create one or more message flows and deploy them to one or more brokers.



Designing a message flow

A message flow can perform a wide range of operations, depending on your business and operational requirements. For best performance and capability, you must design it to include the most appropriate nodes.



Defining message flow content

You can define message flow content by adding and configuring message flow nodes and connecting them to form flows.



Editing configurable properties

You can edit configurable properties in the deployment descriptor file (`broker.xml`) of your broker archive.

“Configure a workload management policy within Integration Registry” on page 644

Describes how to create and configure a policy, and defines the precedent rules.

“Viewing accounting and statistics data in the web user interface” on page 499

You can use the web user interface to view snapshot accounting and statistics data.

Related reference:



Built-in nodes

IBM Integration Bus supplies built-in nodes that you can use to define your message flows.



Configurable message flow properties

When you add a message flow to a broker archive (BAR) file in preparation for deploying it to a broker, you can set additional properties that influence its run time operation. These properties are available for review and update when you select the **Manage and Configure** tab for the broker archive file.



Publish message

Related information:



[WebSphere MQ Version 7 product documentation](#)

Optimizing message flow response times

You can use different solutions to improve message flow response times.

Before you begin

Before you start:

Read the following concept topics:

- “Analyzing message flow performance” on page 469
- “Tuning message flow performance” on page 510
- Message flow nodes

About this task

When you design a message flow, the flexibility and functional capabilities of the built-in nodes often mean that there are several ways to achieve the processing and results that you require. You might find that different solutions deliver different levels of performance and, if performance is an important consideration for you, take it into account when designing your message flow

Your applications can perceive performance in either of these ways:

- The response time indicates how quickly each message is processed by the message flow. The response time is particularly influenced by how you design your message flows. Response time is discussed in this topic.
- The throughput indicates how many messages of particular sizes can be processed by a message flow in a specified time. The throughput is mainly affected by configuration and system resource factors, and is discussed in “Optimizing message flow throughput” on page 551, with other domain configuration information.

The web user interface enables you to view message flow statistics and accounting data, including the message flow throughput and the CPU and elapsed times for transactions in the flow. See “Viewing accounting and statistics data in the web user interface” on page 499 for more information about how you can access the statistical data.

Several aspects influence message flow response times. However, as you create and modify your message flow design to arrive at the best results for your specific business requirements, also consider the eventual complexity of the message flow. The most efficient message flows are not necessarily the easiest to understand and maintain; experiment with the solutions available to arrive at the best balance for your needs.

Several factors influence message flow response times:

The number of nodes that you include in the message flow

Every node increases the amount of processing required in the broker, therefore, consider the content of the message flow carefully, including the use of subflows.

Use as few nodes as possible in a message flow; every node that you include in the message flow increases the amount of processing required in the broker. The number of nodes in a single flow has an upper limit, which is governed by system resources, particularly the stack size. For more information about stack sizes, see “System resources for message flow development” on page 517.

How the message flow routes and processes messages

In some situations, you might find that the built-in nodes, and perhaps other nodes that are available in your system, provide more than one way of providing the same function. Choose the simplest configuration. Where a message flow is required to process more than a single type of record, you can create an easily extendible framework by developing a message flow structure in which there is a parse of the message to determine the type, followed by a RouteToLabel node and Label nodes for each of the types. Where a higher number of label nodes is expected, consider implementing the message parse and Label selection in one message flow,

and the processing of each of the label types into separate message flows. The interface between these two flows would be through a queue.

The following sample demonstrates how you can use the RouteToLabel and Label nodes instead of using multiple Filter nodes in the XML_PassengerQuery message flow.

- Airline Reservations

The following sample demonstrates how you can store routing information in a database table in an in-memory cache in the message flow.

- Message Routing

You can view information about samples only when you use the information center that is integrated with the IBM Integration Toolkit or the online information center. You can run samples only when you use the information center that is integrated with the IBM Integration Toolkit.

If your message flow includes loops

Avoid loops of repeating nodes, which can be very inefficient and can cause performance and stack problems. You might find that a Compute node with multiple PROPAGATE statements avoids the need to loop around a series of nodes.

The efficiency of the ESQL

Check all the ESQL code that you have created for your message flow nodes. As you develop and test a node, you might maintain statements that are not required when you have finalized your message processing. You might also find that something you have coded as two statements can be coded as one. Taking the time to review and check your ESQL code might provide simplification and performance improvements.

The use of persistent and transactional messages

Persistent messages are saved to disk during message flow processing. You can avoid this situation by specifying that messages are non-persistent on input, output, or both. If your message flow is handling only non-persistent messages, check the configuration of the nodes and the message flow itself; if your messages are non-persistent, transactional support might be unnecessary. The default configuration of some nodes enforces transactionality; if you update these properties and redeploy the message flow, response times might improve.

Message size

A larger message takes longer to process. If you can split large messages into smaller units of information, you might be able to improve the speed at which they are handled by the message flow. The following sample demonstrates how to minimize the virtual storage requirements for the message flow to improve a message flow's performance when processing potentially large messages.

- Large Messaging

You can view information about samples only when you use the information center that is integrated with the IBM Integration Toolkit or the online information center. You can run samples only when you use the information center that is integrated with the IBM Integration Toolkit.

Message format

Although IBM Integration Bus supports multiple message formats, and provides facilities that you can use to transform from one format to

another, this transformation increases the amount of processing required in the broker. Make sure that you do not perform any unnecessary conversions or transformations.

You can find more information about improving the performance of a message flow in a developerWorks article (developerWorks article on message flow performance).

Related concepts:

“Message flow statistics and accounting data” on page 471

Message flow statistics and accounting data can be collected to record performance and operating details of one or more message flows.

“System resources for message flow development” on page 517

Configure your message flows to make the best use of computer resources, especially if you will process large messages.

Related tasks:

“Optimizing message flow throughput” on page 551

Each message flow that you design must provide a complete set of processing for messages received from a particular source. This design might result in very complex message flows that include large numbers of nodes that can cause a performance overhead, and might create potential bottlenecks. You can increase the number of message flows that process your messages to provide the opportunity for parallel processing and therefore improved throughput.

“Viewing message flow accounting and statistics data” on page 496

You can view snapshot accounting and statistics data as it is produced by the broker.

“Tuning message flow performance” on page 510

You can configure your message flows to enhance their performance in terms of resource usage, message flow throughput, and response times.

“Analyzing message flow performance” on page 469

You can configure your broker to collect statistics about the operation and behavior of your message flows. Use this information to assess the performance of your message flows.



Configuring brokers for test and production environments

Create one or more brokers on one or more computers, and configure them on your test and production systems to process messages that contain your business data.



Designing a message flow

A message flow can perform a wide range of operations, depending on your business and operational requirements. For best performance and capability, you must design it to include the most appropriate nodes.



Using more than one input node

You can include more than one input node in a single message flow.



Creating a message flow

Create a message flow to specify how to process messages in the broker. You can create one or more message flows and deploy them to one or more brokers.



Editing configurable properties

You can edit configurable properties in the deployment descriptor file (broker.xml) of your broker archive.

“Viewing accounting and statistics data in the web user interface” on page 499
You can use the web user interface to view snapshot accounting and statistics data.

Related reference:



Built-in nodes

IBM Integration Bus supplies built-in nodes that you can use to define your message flows.

Tuning the SAP adapter for scalability and performance

You can configure the number of listeners on the adapter and the number of additional instances on the message flow to prevent delays when processing synchronous calls from SAP.

Before you begin

Before you start:

- Ensure that you have created and saved a message flow that contains one or more SAP nodes; for more information, see *Developing message flows that use WebSphere Adapters*.
- Read the concept topic *SAP adapter scalability and performance*.

About this task

The following steps describe how to tune the SAP adapter for scalability and performance.

Procedure

1. Deploy the BAR file, as described in *Deploying a message flow that uses WebSphere Adapters*.
2. Start user trace, as described in *Starting user trace*.
3. Start to collect accounting and statistics data, as described in “Starting collection of message flow accounting and statistics data with the `mqsichange flowstats` command” on page 482.
4. Run the SAP programs at a typical load.
5. Check user trace for message BIP3461. This message tells you the highest number of listeners that are waiting for message flow threads at any one time. In an ideal situation, this number should be as low as possible. To reduce the number of listeners that are waiting for message flow threads, increase the number of instances on the SAPInput node, or the message flow that contains it.
6. Inspect the accounting and statistics data for the message flow.

What to do next

Unnecessary data conversion can also affect performance. To maximize performance and avoid unnecessary data conversion, ensure that messages that are passed to a SAPRequest node contain the correct data types. The DataObject domain is the default domain when parsing messages that are produced by the SAPRequest node. However, when passing data to the SAPRequest node (for example, by using an MQInput node), the use of a different domain can improve performance. For example, use the XMLNSC parser with the MQInput node to parse XML messages.

Related concepts:



SAP adapter scalability and performance

You can improve performance by configuring the number of listeners on the adapter and the number of additional instances on the message flow to prevent delays when processing synchronous calls from SAP.



Overview of WebSphere Adapter for SAP Software

With WebSphere Adapter for SAP Software you can create integrated processes that include the exchange of information with an SAP server, without special coding.

Related tasks:



Developing message flows that use WebSphere Adapters

For information about how to develop message flows that use WebSphere Adapters, see the following topics.



Deploying a message flow that uses WebSphere Adapters

Deploy the resources that are generated when you run the Adapter Connection wizard by adding them to a broker archive (BAR) file.



Starting user trace

Use user trace for debugging your applications; you can trace brokers, integration servers, and deployed message flows. Start user trace facilities using the **mqsichangetrace** command or the IBM Integration Explorer.

“Starting collection of message flow accounting and statistics data with the **mqsichangeflowstats** command” on page 482

You can use the **mqsichangeflowstats** command to start collecting message flow accounting and statistics data for an active broker at any time. You can specify what type of statistics you want to collect, and where to send the data.

Related reference:



SAPInput node

Use the SAPInput node to accept input from an SAP application.



Configurable message flow properties

When you add a message flow to a broker archive (BAR) file in preparation for deploying it to a broker, you can set additional properties that influence its run time operation. These properties are available for review and update when you select the **Manage and Configure** tab for the broker archive file.

Tuning SOAP processing for scalability and performance

You can improve performance when processing SOAP messages by configuring the number of additional instances on the message flow.

When processing SOAP messages, you can improve performance by configuring the number of additional instances on the message flow. This configuration increases the number of available threads for processing messages on that node and allows parallel processing of messages. You can configure the number of additional instances at message flow level or on the node itself.

However, when using SOAP with WS-RM (web services reliable messaging), increasing the number of additional instances does not always affect the rate at which messages are processed. If the message flow is processing a group of messages from a single sequence and the WS-RM InOrder assurance is enabled, all

the messages from that sequence are processed by a single instance of the message flow, even if additional instances are configured.

This ensures that messages are processed in the order in which they were sent, but means that the rate of processing stays the same for the messages in that sequence. In this example, the SOAP message flow with WS-RM enabled processes messages from multiple inbound sequences in parallel, but it processes only one request at a time for messages from the same inbound sequence.

For more information about the concepts of WS-RM, read *Web Services Reliable Messaging*.

JVM tuning

When WS-RM is enabled for SOAP message processing, a large JVM heap size is required. Before deploying SOAP message flows that use WS-RM, ensure that your JVM heap size is set to a minimum of 1.5 GB.

Related concepts:



Web Services Reliable Messaging

IBM Integration Bus supports WS-RM (Web Services Reliable Messaging), which allows two systems to reliably exchange messages with each other.

Related tasks:



Configuring WS-RM

Configure reliable messaging for web services.

Related reference:



Configurable message flow properties

When you add a message flow to a broker archive (BAR) file in preparation for deploying it to a broker, you can set additional properties that influence its run time operation. These properties are available for review and update when you select the **Manage and Configure** tab for the broker archive file.

Troubleshooting performance problems

Follow this guidance to resolve common problems with IBM Integration Bus performance.

About this task

- **Scenario:** You are experiencing problems with performance, such as:
 - Poor response times in the IBM Integration Toolkit when developing message flows
 - Poor response time at deployment
 - Individual messages taking a long time to process
 - Poor overall performance, or performance that does not scale well
- **Solution:** Possible solutions are:
 - Tune the broker
 - Speed up WebSphere MQ persistent messaging by optimizing the I/O (input/output)
 - Speed up database access by optimizing I/O
 - Increase system memory
 - Use additional instances or multiple integration servers

- Optimize ESQL statements for best performance

The reports in WebSphere MQ Family Category 2 (freeware) SupportPacs contain additional advice about performance, and they are available for download from the WebSphere MQ SupportPacs web page.

The following topics contain information about dealing with some common performance problems:

- “You are experiencing poor performance in the IBM Integration Toolkit when working with large projects”
- “Performance is reduced when you run Web Services with small message sizes” on page 562
- “The PutTime reported by WebSphere MQ on z/OS, and other times or timestamps are inconsistent” on page 563
- “You have a component that is running slowly” on page 564

The following topics contain guidance to help you optimize the performance of your code and message flows:

- “Message flow design and performance” on page 453
- “Code design and performance” on page 455

Related tasks:

“Performance” on page 450

You can use message flow design and coding techniques to optimize the performance of your message flows. You can then analyze statistical information for your message flows and modify aspects of your broker configuration to tune the performance of your brokers and message flows.

Chapter 3, “Performance, monitoring, and workload management,” on page 449
You can change various aspects of your broker configuration to tune brokers and message flows, and monitor message flows and publish/subscribe applications.

“Performance planning” on page 451

When you design your IBM Integration Bus environment and the associated resources, the decisions that you make can affect the performance of your applications.

“Analyzing message flow performance” on page 469

You can configure your broker to collect statistics about the operation and behavior of your message flows. Use this information to assess the performance of your message flows.

“Tuning message flow performance” on page 510

You can configure your message flows to enhance their performance in terms of resource usage, message flow throughput, and response times.

“Tuning the broker” on page 543

You can complete several tasks that enable you to tune different aspects of the broker performance.

You are experiencing poor performance in the IBM Integration Toolkit when working with large projects

Follow this guidance to resolve the problem of reduced performance when you are working with large projects.

Procedure

- **Scenario:** You are experiencing poor performance in the IBM Integration Toolkit when working with large or complex projects.

- **Explanation:** Performance is reduced because of frequent project changes, such as adding and removing projects, or using **Project > Clean**. Complete project updates use large amounts of memory due to the size, number, and connections between files.
- **Solution:** Increase your system memory.

Related tasks:

“Performance” on page 450

You can use message flow design and coding techniques to optimize the performance of your message flows. You can then analyze statistical information for your message flows and modify aspects of your broker configuration to tune the performance of your brokers and message flows.

Chapter 3, “Performance, monitoring, and workload management,” on page 449

You can change various aspects of your broker configuration to tune brokers and message flows, and monitor message flows and publish/subscribe applications.

“Performance planning” on page 451

When you design your IBM Integration Bus environment and the associated resources, the decisions that you make can affect the performance of your applications.

“Tuning the broker” on page 543

You can complete several tasks that enable you to tune different aspects of the broker performance.

Performance is reduced when you run Web Services with small message sizes

Follow this guidance to resolve the problem of reduced performance when you run Web Services with small message sizes.

Procedure

- **Scenario:** You see poor response times and throughput rates when you run Web Services using HTTP, and send smaller messages sizes (typically less than 32 KB). Throughput rates can fluctuate with message size. IBM Integration Bus running on the AIX platform might be affected.
- **Explanation:** The default configuration of HTTP enables the Nagle algorithm, which seeks to improve the efficiency of Internet Protocol networks by reducing the number of packets sent. It works by buffering small packets together, creating a smaller number of large packets. The HTTPRequest node uses the platform default for the **tcpnodelay** setting of its sockets. You can disable the Nagle algorithm at either the operating system level (system wide) or through IBM Integration Bus (affecting only the IBM Integration Bus HTTP sockets).
- **Solution:** Use the following commands to disable the Nagle algorithm:

HTTPRequest, SOAPRequest, and SCARquest nodes

```
mqschangeproperties <BrokerName> -e <IntegrationServerName>
-o ComIbmSocketConnectionManager -n tcpNoDelay -v true|false
mqschangeproperties <BrokerName> -e <IntegrationServerName>
-o ComIbmSocketConnectionManager -n tcpNoDelaySSL -v true|false
```

Embedded listener for HTTPReply, SOAPReply, and SCARReply nodes

```
mqschangeproperties <BrokerName> -e <IntegrationServerName>
-o HTTPConnector -n tcpNoDelay -v true
mqschangeproperties <BrokerName> -e <IntegrationServerName>
-o HTTPSConnector -n tcpNoDelay -v true
```

HTTP Listener for HTTPReply, SOAPReply, and SCARReply nodes

```
mqsichangeproperties <BrokerName> -b httplistener  
-o HTTPConnector -n tcpNoDelay -v true|false  
mqsichangeproperties <BrokerName> -b httplistener  
-o HTTPSConnector -n tcpNoDelay -v true|false
```

To determine the value set, take the following steps:

Report property values

Use the following command:

```
mqsireportproperties <BrokerName> -e <IntegrationServerName>  
-o ComIbmSocketConnectionManager -r  
mqsireportproperties <BrokerName> -e <IntegrationServerName>  
-o HTTPConnector -r  
mqsireportproperties <BrokerName> -e <IntegrationServerName>  
-o HTTPSConnector -r  
mqsireportproperties <BrokerName> -b httplistener  
-o HTTPConnector -r  
mqsireportproperties <BrokerName> -b httplistener  
-o HTTPSConnector -r
```

Related tasks:

“Performance” on page 450

You can use message flow design and coding techniques to optimize the performance of your message flows. You can then analyze statistical information for your message flows and modify aspects of your broker configuration to tune the performance of your brokers and message flows.

Chapter 3, “Performance, monitoring, and workload management,” on page 449

You can change various aspects of your broker configuration to tune brokers and message flows, and monitor message flows and publish/subscribe applications.

“Performance planning” on page 451

When you design your IBM Integration Bus environment and the associated resources, the decisions that you make can affect the performance of your applications.

“Tuning the broker” on page 543

You can complete several tasks that enable you to tune different aspects of the broker performance.

The PutTime reported by WebSphere MQ on z/OS, and other times or timestamps are inconsistent

Follow this guidance to resolve the problem of inconsistently reported times and timestamps.

Procedure

- **Scenario:** The PutTime reported by WebSphere MQ on z/OS, and other times or timestamps are inconsistent. A difference of approximately 20 seconds is detected in:
 - Traces (including those obtained from the Trace node)
 - The MQPUTTIME timestamp in the message MQMD header
 - Timestamps obtained from ESQL (for example, in a Compute node)
- **Explanation:** IBM Integration Bus reports the time using Coordinated Universal Time (UTC), which does not account for leap seconds. However, on z/OS, the message putTime that is reported by WebSphere MQ in the MQMD header of a message *does* account for leap seconds, using the value specified for the number of leap seconds in the CVT field.

This inconsistency can cause:

- Problems when debugging
- Problems with message flows if you use timestamps to control the flow of messages
- Misinformation
- **Solution:** Set the CVT field so that it agrees with the UTC leap seconds. Alternatively, add an offset to adjust a z/OS timestamp reading. For example, add 20 seconds when getting the CURRENT_TIME in ESQL.

Related tasks:

“Performance” on page 450

You can use message flow design and coding techniques to optimize the performance of your message flows. You can then analyze statistical information for your message flows and modify aspects of your broker configuration to tune the performance of your brokers and message flows.

Chapter 3, “Performance, monitoring, and workload management,” on page 449
 You can change various aspects of your broker configuration to tune brokers and message flows, and monitor message flows and publish/subscribe applications.

“Performance planning” on page 451

When you design your IBM Integration Bus environment and the associated resources, the decisions that you make can affect the performance of your applications.

“Tuning the broker” on page 543

You can complete several tasks that enable you to tune different aspects of the broker performance.

You have a component that is running slowly

Follow this guidance to resolve the problem of a component that is running slowly.

Procedure

- **Scenario:** A particular component, or the system in general, is running slowly.
- **Solution:** Consider the following actions:
 - Check whether tracing is on. You might have started IBM Integration Bus user tracing or service tracing, ODBC tracing, WebSphere MQ tracing, or native database tracing. If one or more of these traces are active, turn them off.
 - Clear out all old abend files from your errors directory. If you do not clear the directory of unwanted files, you might find that your system performance degrades because significant space is used up.
 - On Windows, use the workpath **-w** parameter of the **mqsicreatebroker** command to create the errors directory in a hard disk partition that does not contain IBM Integration Bus or Windows.
 - Increase your system memory.

Related concepts:



Logs

If an error is reported by an IBM Integration Bus component, start your investigations into its causes by looking at the product and systems logs to which information is written during component operation.

Related tasks:

“Performance” on page 450

You can use message flow design and coding techniques to optimize the performance of your message flows. You can then analyze statistical information for your message flows and modify aspects of your broker configuration to tune

the performance of your brokers and message flows.

Chapter 3, “Performance, monitoring, and workload management,” on page 449
You can change various aspects of your broker configuration to tune brokers and message flows, and monitor message flows and publish/subscribe applications.

“Performance planning” on page 451

When you design your IBM Integration Bus environment and the associated resources, the decisions that you make can affect the performance of your applications.

“Tuning the broker” on page 543

You can complete several tasks that enable you to tune different aspects of the broker performance.

Chapter 4, “Troubleshooting and support,” on page 647

If you are having problems with your message flow applications, use the techniques described in this section to help you to diagnose and solve the problems.

“Making initial checks” on page 650

Before you start problem determination in detail, consider whether there is an obvious cause of the problem, or an area of investigation that is likely to give useful results. This approach to diagnosis can often save a lot of work by highlighting a simple error, or by narrowing down the range of possibilities.

“Using trace” on page 845

You can use different types of trace to help you with problem determination and troubleshooting.

“Resolving problems with performance” on page 823

Use the advice given here to help you to resolve common problems with performance.

Related reference:

 **mqsicreatebroker** command

Use the **mqsicreatebroker** command to create a broker and its associated resources.

Business-level monitoring

You can configure your message flow to emit event messages that can be used to support transaction monitoring and auditing, and business process monitoring.

Before you begin

Before you start:

- To learn about the basic concepts of monitoring, see “Monitoring basics” on page 568.
- To decide whether to do transaction monitoring or business monitoring, see “Monitoring scenarios” on page 571.
- To decide whether to use monitoring properties or a monitoring profile, see “Deciding how to configure monitoring events for message flows” on page 573.

About this task

An event is a message that a message flow publishes when something interesting happens. The message contains information about the source of the event, the time of the event, and the reason for the event. The event can include the message bit stream, and can also include selected elements from the message body. These fields

can be used to correlate messages that belong to the same transaction, or to convey business data to a monitoring application.

Note: Monitoring messages are published using the persistent as topic option. By default, the publication resolves to be non persistent, but you can change a publication to be persistent by configuring named topics in WebSphere MQ. For more information, see the Subscriptions and message persistence topic in the WebSphere MQ information center.

To receive monitoring events, complete the following steps.

Procedure

1. Configure event sources on the flow.

You can configure event sources by using either monitoring properties or a monitoring profile. For instructions, see one of the following topics.

- “Configuring monitoring event sources using monitoring properties” on page 575
- “Configuring monitoring event sources using a monitoring profile” on page 580

2. Enable event sources.

You can enable event sources by using either the monitoring properties on the node or the **-i** parameter on the **mqsichangeflowmonitoring** command. For instructions, see “Enabling and disabling event sources” on page 585

3. Activate monitoring for the flow.

You can activate monitoring by setting the **-c** parameter on the **mqsichangeflowmonitoring** command. For instructions, see “Activating monitoring” on page 583.

4. Subscribe to the topic for the flow.

The form of the topic name is shown in the following example.

```
$SYS/Broker/brokerName/Monitoring/integrationServerName/flowName
```

What to do next

You can use WebSphere Business Monitor to monitor your message flows. For more information, see “Monitoring flows by using WebSphere Business Monitor” on page 587.

Related concepts:

“Monitoring basics” on page 568

Message flows can be configured to emit events. The events can be read and used by other applications for transaction monitoring, transaction auditing, and business process monitoring.

“Monitoring scenarios” on page 571

Events can be used to support transaction monitoring, transaction auditing and business process monitoring.

Related tasks:



Developing integration solutions

IBM Integration Bus provides a flexible environment in which you can develop integration solutions to transform, enrich, route, and process your business messages and data. You can integrate client applications that use different protocols and message formats.

“Configuring monitoring event sources using monitoring properties” on page 575
In the Message flow Editor, use the Monitoring tab on the properties of a node to add one or more monitoring events.

“Configuring monitoring event sources using a monitoring profile” on page 580
You can create a monitoring profile and use the **mqsichangeflowmonitoring** command to configure your message flows to emit monitoring events.

“Enabling and disabling event sources” on page 585

When events have been configured for a message flow and deployed to the broker, you can enable and disable individual events. You can do this from the command line, without having to redeploy the message flow, or you can do this from the Message Flow editor, in which case you must redeploy the flow.

“Resolving problems when you monitor message flows” on page 827

Follow the advice for dealing with common problems that can arise when you are monitoring IBM Integration Bus flows.

Related reference:

 **mqsireportflowmonitoring** command

Use the **mqsireportflowmonitoring** command to display the current options for monitoring that have been set using the **mqsichangeflowmonitoring** command.

 **mqsichangeflowmonitoring** command

Use the **mqsichangeflowmonitoring** command to enable monitoring of message flows.

Monitoring overview

To monitor message flows, learn about monitoring basics and scenarios, and decide how to configure event sources.

An event is a message that a message flow publishes when something interesting happens. The message contains information about the source of the event, the time of the event, and the reason for the event. The event can include the message bit stream, and can also include selected elements from the message body. These fields can be used to correlate messages that belong to the same transaction, or to convey business data to a monitoring application.

To learn more about the concepts of monitoring, see the following topics.

- “Monitoring basics” on page 568
- “Monitoring scenarios” on page 571
- “Deciding how to configure monitoring events for message flows” on page 573

To configure monitoring, complete the steps in “Business-level monitoring” on page 565.

You can also find reference information about monitoring in Monitoring message flows.

Related tasks:

“Resolving problems when you monitor message flows” on page 827

Follow the advice for dealing with common problems that can arise when you are monitoring IBM Integration Bus flows.

Monitoring basics

Message flows can be configured to emit events. The events can be read and used by other applications for transaction monitoring, transaction auditing, and business process monitoring.

Monitoring Events

A monitoring event is an XML document that conforms to the monitoring event schema. Each event contains the following information:

- Source of the event
- Name of the event
- Sequence number and creation time
- Correlation ID for events emitted by the same transaction or unit of work

Additionally, a monitoring event can contain the following items:

- Application data extracted from the message
- Part or all of the message bit stream

See The monitoring event for more details

Event Sources

A message flow can emit two kinds of events:

Transaction events

Transaction events are emitted only from input nodes.

Terminal events

Terminal events are emitted from any terminal of any node, including input nodes.

An individual message flow can choose to emit transaction events, terminal events, or both kinds of event. You can configure, enable, and disable, both types of events in either of the following ways:

- Using the monitoring properties of the message flow.
- Using a monitoring profile configurable service.

The use of a monitoring profile configurable service overrides the monitoring properties of a message flow.

An event source address identifies an event source in a message flow.

Because terminal events can be emitted from any node in a message flow, they can be used as an alternative to dedicated event-emitting nodes or subflows such as that supplied in SupportPac IA9V.

Event sources emit events only if monitoring is activated for the message flow.

Terminal events

Any terminal in a message flow can be an event source. If the event source is active, it emits an event each time a message passes through the terminal, subject to the evaluation of the eventFilter expression; see “Event output options” on page 569.

Transaction events

Each input node in a message flow contains three events sources, in addition to terminal events.

Event source	Event source address	Description
Transaction start	<i>Nodename.transaction.Start</i>	The event is emitted when the message is read from the transport.
Transaction end	<i>Nodename.transaction.End</i>	The event is emitted when IBM Integration Bus has completed all processing of the message.
Transaction rollback	<i>Nodename.transaction.Rollback</i>	The event is emitted instead of transaction end if the message flow throws an exception which is not caught and processed within the message flow.

Events are emitted subject to the evaluation of the `eventFilter` expression; see “Event output options.”

If a message flow handles its own exceptions, a transaction end event, rather than a transaction rollback event, is issued, because the flow has taken control of the error and terminated normally. In this case, if you need to distinguish errors, you can configure terminal events at appropriate nodes in the flow.

Event output options

When you configure an event source, you can define a filter to control whether the event is emitted. You can tailor event emission to your business requirements, by filtering out events that do not match a set of rules. For example, you might decide to emit events only for transactions above a minimum amount.

```
$Body/StockTrade/Details/Value > 10000
```

This can reduce the number of events that are emitted, and reduce the workload on your monitoring application.

You might filter out the `transaction.Start` and `transaction.End` events that are emitted by the `MQInput` node for WebSphere MQ Backout transaction events after a backout threshold is reached so that appropriate data can be collected by an event monitoring application.

```
3 >= $Root/MQMD/BackoutCount
```

Events are published to a topic, where they can be read by multiple subscribers. The topic name is of the form:

```
$SYS/Broker/brokerName/Monitoring/integrationServerName/flowName
```

The hierarchical structure allows subscribers to filter the events which they receive. One subscriber can receive events from all message flows in the broker, while another receives only the events from a single integration server.

You decide whether events participate in transactions when you configure a monitoring event source. In general:

- If you want an event to be emitted only if the message flow transaction commits, configure the event source to coordinate the events with the message flow transaction.

- If you want an event to be emitted regardless of whether the message flow transaction commits or rolls back, configure the event source to emit events out of sync point. Such events are available immediately.
- If you want a group of events to be emitted together regardless of whether the message flow transaction commits or rolls back, configure the event source to emit events in a second, independent, unit of work.

Default monitoring configuration

If monitoring is activated for a message flow, and neither monitoring properties nor a monitoring profile configurable service have been configured for the flow, the default behavior is for transaction events to be emitted from each input node of the message flow. The events contain the bit stream of the input message.

Related tasks:

“Business-level monitoring” on page 565

You can configure your message flow to emit event messages that can be used to support transaction monitoring and auditing, and business process monitoring.

Developing integration solutions

IBM Integration Bus provides a flexible environment in which you can develop integration solutions to transform, enrich, route, and process your business messages and data. You can integrate client applications that use different protocols and message formats.

“Configuring monitoring event sources using monitoring properties” on page 575

In the Message flow Editor, use the Monitoring tab on the properties of a node to add one or more monitoring events.

“Configuring monitoring event sources using a monitoring profile” on page 580

You can create a monitoring profile and use the **mqsichangeflowmonitoring** command to configure your message flows to emit monitoring events.

Related reference:

Monitoring profile

To customize events after a message flow has been deployed, but without redeploying the flow, use a monitoring profile configurable service. By using this service, you can apply a monitoring profile to one or more message flows.

mqsichangeflowmonitoring command

Use the **mqsichangeflowmonitoring** command to enable monitoring of message flows.

mqsicreateconfigurableservice command

Use the **mqsicreateconfigurableservice** command to create an object name for a broker external resource.

mqsichangeproperties command

Use the **mqsichangeproperties** command to modify broker properties and properties of broker resources.

Built-in nodes

IBM Integration Bus supplies built-in nodes that you can use to define your message flows.



The monitoring event

You can configure IBM Integration Bus to emit a monitoring event (an XML document) when something interesting happens. Events are typically emitted to support transaction monitoring, transaction auditing, and business process monitoring. The event XML conforms to the monitoring event schema `WMBEvent.xsd`.

Monitoring scenarios

Events can be used to support transaction monitoring, transaction auditing and business process monitoring.

Transaction monitoring and auditing

The events published by IBM Integration Bus can be written to a transaction repository, creating an audit trail of the transactions that are processed by a broker. A transaction repository can be used for monitoring, auditing and replay of transactions. Bitstream data can be included so that failed transactions can be resubmitted. You can perform the following tasks to set up transaction monitoring and auditing.

Configure events for your transactions

In most cases bitstream information is not sufficient to allow querying of the logged transactions. Key fields and other correlation data can be extracted from the message payload and placed into the `wmb:applicationData/wmb:simpleContent` or `wmb:applicationData/wmb:complexContent` element of the event. The logging application or message flow can extract these fields and log them with the message bit stream.

Subscribe to the event topic and write events to a repository

You can create a message flow, or any WebSphere MQ application, that subscribes to the event topic and writes events to a relational database. The details of the database schema depend on the requirements of your organization, for example the number of key fields and transaction IDs.

Business process monitoring

The events published by a broker can be monitored by WebSphere Business Monitor. Important fields in the message payload can be added to the events emitted by your message flows, allowing them to be monitored. You can use the following items to help you use WebSphere Business Monitor to monitor your message flows:

Message-driven bean

The events must be submitted to the CEI repository for WebSphere Business Monitor to monitor them. A message-driven bean is supplied for this purpose. The message-driven bean, which runs in WebSphere Application Server, subscribes to the event topic and writes events that match its subscription to the CEI repository as Common Base Event events.

A message-driven bean is supplied with the WebSphere Business Monitor sample. For instructions about how to install and configure the sample message-driven bean, see *Running the WebSphere Business Monitor sample*.

WebSphere Business Monitor Model

IBM Integration Bus includes an example monitor model for use with WebSphere Business Monitor. This model demonstrates how to monitor

transaction events and terminal events, including events that capture data from the input message and output message. Modify the model to match your actual events and message formats.

A WebSphere Business Monitor Model is supplied with the WebSphere Business Monitor sample. For instructions about how to install and configure the sample WebSphere Business Monitor Model, see [Running the WebSphere Business Monitor sample](#).

Related concepts:

“Monitoring basics” on page 568

Message flows can be configured to emit events. The events can be read and used by other applications for transaction monitoring, transaction auditing, and business process monitoring.

Related tasks:

“Business-level monitoring” on page 565

You can configure your message flow to emit event messages that can be used to support transaction monitoring and auditing, and business process monitoring.



Developing integration solutions

IBM Integration Bus provides a flexible environment in which you can develop integration solutions to transform, enrich, route, and process your business messages and data. You can integrate client applications that use different protocols and message formats.

“Configuring monitoring event sources using a monitoring profile” on page 580

You can create a monitoring profile and use the **mqsichangeflowmonitoring** command to configure your message flows to emit monitoring events.

“Resolving problems when you monitor message flows” on page 827

Follow the advice for dealing with common problems that can arise when you are monitoring IBM Integration Bus flows.

“Configuring monitoring event sources using monitoring properties” on page 575

In the Message flow Editor, use the Monitoring tab on the properties of a node to add one or more monitoring events.

Related reference:



Monitoring profile

To customize events after a message flow has been deployed, but without redeploying the flow, use a monitoring profile configurable service. By using this service, you can apply a monitoring profile to one or more message flows.



mqsichangeflowmonitoring command

Use the **mqsichangeflowmonitoring** command to enable monitoring of message flows.



mqsicreateconfigurableservice command

Use the **mqsicreateconfigurableservice** command to create an object name for a broker external resource.



mqsichangeproperties command

Use the **mqsichangeproperties** command to modify broker properties and properties of broker resources.



Built-in nodes

IBM Integration Bus supplies built-in nodes that you can use to define your message flows.

Deciding how to configure monitoring events for message flows

Decide whether to use monitoring properties, or a monitoring profile configurable service, to customize the events produced by a message flow.

If you want to customize events when you are designing the message flow, use monitoring properties. If you want to customize events after a message flow has been deployed, but without redeploying, use a monitoring profile. This topic gives information about both methods.

Using monitoring properties to configure events

Using the IBM Integration Toolkit Message Flow Editor, you can configure event sources on most nodes in a message flow. A node that supports the configuration of event sources has a Monitoring tab on its Properties view; use this tab to add events and to set their properties. When you deploy the message flow in a broker archive file, the monitoring properties are included as part of the message flow.

Key facts about monitoring properties:

- Use monitoring properties when you want to configure events at message flow design time.
- Monitoring properties apply only to the message flow in question. You can share monitoring properties between message flows only by creating reusable subflows.
- Monitoring properties are deployed in a BAR file as part of the message flow.
- You use the **mqsichangeflowmonitoring** command to activate the message flow to emit monitoring events.
- You can use the **mqsichangeflowmonitoring** command to enable or disable individual event sources in a message flow.
- To change any other properties of an event, you alter the monitoring properties in the flow, then redeploy the BAR file.

To configure monitoring events using monitoring properties, see this information: “Configuring monitoring event sources using monitoring properties” on page 575.

Tip: Use the **mqsireportflowmonitoring** command to report a list of the names of the event sources for a message flow. You can then use these event source names in a **mqsichangeflowmonitoring** command to enable or disable individual event sources from the command line.

Using a monitoring profile to configure events

Using operational commands, you can create a monitoring profile configurable service directly on the broker, and associate it with one or more message flows.

Key facts about monitoring profiles:

- Use a monitoring profile when you want to configure events for a message flow that has already been deployed and for which no events have been configured.
- Use a monitoring profile to override the monitoring properties of a message flow that has already been deployed, as an alternative to redeploying the BAR file. The monitoring profile completely replaces all monitoring properties.
- A single monitoring profile can be applied to many message flows, provided that the message flows contain the same event sources.

- Monitoring profiles are created directly on the broker using the **mqsicreateconfigurable-service** command and the **mqsichangeproperties** command. They are not deployed in a BAR file.
- You use the **mqsichangeflowmonitoring** command to associate the monitoring profile with a message flow.
- You use the **mqsichangeflowmonitoring** command to activate the message flow to emit monitoring events.
- You can use the **mqsichangeflowmonitoring** command to enable or disable individual event sources in a message flow.

To configure monitoring events using monitoring properties, see this information:

“Configuring monitoring event sources using a monitoring profile” on page 580

Tip: Use the **mqsireportflowmonitoring** command to report a list of event sources for a message flow. You can then use the names of these event source in a subsequent **mqsichangeflowmonitoring** command to enable and disable individual event sources from the command line.

Tip: Use the **mqsireportflowmonitoring** command to report the monitoring profile for a message flow. You can edit the profile, to add or change event sources, then update it using the **mqsichangeproperties** command.

Tip: Use the **mqsireportflowmonitoring** command to create the equivalent monitoring profile for a message flow for which monitoring properties are configured. You can edit the profile, then use it to create a new monitoring profile configurable service using the **mqsicreateconfigurable-service** and **mqsichangeproperties** commands. You can then associate the new profile with the original flow using the **mqsichangeflowmonitoring** command. You can use this technique to override the monitoring properties for a message flow, without having to edit the flow and redeploy the BAR file.

Related concepts:

“Monitoring basics” on page 568

Message flows can be configured to emit events. The events can be read and used by other applications for transaction monitoring, transaction auditing, and business process monitoring.

Related tasks:

“Configuring monitoring event sources using monitoring properties” on page 575

In the Message flow Editor, use the Monitoring tab on the properties of a node to add one or more monitoring events.

“Configuring monitoring event sources using a monitoring profile” on page 580

You can create a monitoring profile and use the **mqsichangeflowmonitoring** command to configure your message flows to emit monitoring events.

Related reference:

 Monitoring profile

To customize events after a message flow has been deployed, but without redeploying the flow, use a monitoring profile configurable service. By using this service, you can apply a monitoring profile to one or more message flows.

 **mqsichangeflowmonitoring** command

Use the **mqsichangeflowmonitoring** command to enable monitoring of message flows.

mqsireportflowmonitoring command

Use the **mqsireportflowmonitoring** command to display the current options for monitoring that have been set using the **mqsichangeflowmonitoring** command.

mqsichangeproperties command

Use the **mqsichangeproperties** command to modify broker properties and properties of broker resources.

mqsicreateconfigurable-service command

Use the **mqsicreateconfigurable-service** command to create an object name for a broker external resource.

Configuring monitoring event sources using monitoring properties

In the Message flow Editor, use the Monitoring tab on the properties of a node to add one or more monitoring events.

Before you begin

Before you start:

Read the following topics:

- “Business-level monitoring” on page 565
- “Monitoring basics” on page 568

You must have a message flow that contains a node to which you want to add a monitoring event.

You can use XPath 1.0 expressions to configure a monitoring event. Some XPath expressions, listed in XPath expressions that are not suitable for the export monitoring information option, are not suitable for use with the export monitoring information option (“Creating a monitor model for WebSphere Business Monitor V7 or later” on page 591).

Creating events

About this task

An event source is a point in a message flow from which a monitoring event can be emitted. Each event source has a set of properties that control the contents of the monitoring events that it emits.

Procedure

1. Display the properties for the node.
2. Select the Monitoring tab.
3. Click **Add**.

The Add entry window is displayed.

4. Complete the **Event Source** field.

The field has a drop-down list of all events that can be defined for this node. The event source information is used to populate the attributes of the `wmb:eventPointData/wmb:messageFlowData/wmb:node` element of the event.

When you have selected the event source, the corresponding value for Event Source Address is displayed as a read-only property.

Tip: If you later decide to enable or disable events using the **mqsichange flow monitoring** command, you must specify a value for Event Source Address, not Event Name.

- Complete the **Event Name** details; select either **Literal** or **Data location**.

Every monitoring event has a name that is placed in the `wmb:eventPointData/wmb:eventIdentity/@wmb:eventName` attribute of the event. The default names are shown in the following table:

Event source	Default event name	Example
Transaction start	<code>nodeLabel.TransactionStart</code>	<code>MQInput.TransactionStart</code>
Transaction end	<code>nodeLabel.TransactionEnd</code>	<code>MQInput.TransactionEnd</code>
Transaction rollback	<code>nodeLabel.TransactionRollback</code>	<code>MQInput.TransactionRollback</code>
Terminal	<code>nodeLabel.terminal_label.Terminal</code>	<code>MQInput.OutTerminal</code>

You can override the default in these ways:

- By specifying an alternative literal string.
- By specifying an XPath query; the query extracts the event name from a field in the input message. Click **Edit** to use the XPath Expression Builder.

You cannot use monitoring properties to configure transaction events on the following nodes:

- Collector node
- Resequencing node

Use a monitoring profile instead; see “Configuring monitoring event sources using a monitoring profile” on page 580.

- Optional: Complete the Event Filter section by providing an XPath expression to control whether the event is emitted. Take one of the following steps:
 - Type in the expression (for example, `$Body/StockTrade/Details/Value > 10000`); or
 - Click **Edit** to launch XPath Expression Builder.

The expression must evaluate to true or false, and can reference fields in the message tree, or elsewhere in the message assembly. The default value is `true()`, which means that the event is always produced.

Using this facility, you can tailor event emissions to your business requirements, by filtering out events that do not match a set of rules. This can reduce the number of events emitted, and reduce the workload on your monitoring application.

- Optional: Complete the **Event Payload** section if the event is to contain selected data fields extracted from the message. Click **Add** to launch the Add Data Location dialog box. Take one of the following steps:
 - Type in the location (for example, `$LocalEnvironment/File/Name`); or
 - Click **Edit** to launch XPath Expression Builder.

You can extract one or more fields from the message data and include it with the event. The fields can be simple or complex. Simple content is contained in the `wmb:applicationData/wmb:simpleContent` field of the event; complex data is contained in the `wmb:applicationData/wmb:complexContent` field.

This facility is commonly used for communicating significant business data in a business event. If the event contains the input bit stream, this facility can also be used to extract key fields, allowing another application to provide an audit trail or to resubmit failed messages.

- Optional: Select the **Include bitstream data in payload** field if the event is to capture message bitstream data.

Content

Select from Headers, Body, All.

Encoding

Select from base64, HexBinary and CData (the original text, without encoding).

9. Optional: Select the Correlation tab, to complete details for event correlation.
10. Complete the **Event Correlation** details; for information about correlation, see Correlation and monitoring events.

Every monitoring event must contain at least one correlation attribute, and can contain up to three. If you do not specify any correlation information, the first event source in the message flow allocates a unique identifier that all later event sources in the same transaction will use.

- a. Optional: Complete the **Local transaction correlator** details.

Automatic

The local correlator used by the most recent event for this invocation of the message flow will be used. If no local correlator exists yet, a new unique value will be generated.

Specify location of correlator

Enter a value, or click Edit to launch the XPath Expression Builder. The local correlator will be read from the specified location in the message tree. Ensure that the specified location contains a correlator value unique to this invocation of the message flow.

- b. Optional: Complete the **Parent transaction correlator** details to extract a correlation field from the parent transaction.

Automatic

The parent correlator used by the most recent event for this invocation of the message flow will be used. If no parent correlator exists yet, no parent correlator will be used.

Specify location of correlator

Enter a value, or click Edit to launch the XPath Expression Builder. The parent correlator will be read from the specified location in the message tree. Ensure that the specified location contains a suitable value for the parent correlator.

- c. Optional: Complete the **Global transaction correlator** details to extract a correlation field from a global transaction.

Automatic

The global correlator used by the most recent event for this invocation of the message flow will be used. If no global correlator exists yet, no global correlator will be used.

Specify location of correlator

Enter a value, or click Edit to launch the XPath Expression Builder. The global correlator will be read from the specified location in the message tree. Ensure that the specified location contains a suitable value for the global correlator.

11. Optional: Choose whether the emission of monitoring events by a message flow is coordinated with the message flow transaction, or is in an independent unit of work, or is not in a unit of work.

Click the Transaction tab and select the appropriate option for **Event Unit of Work**.

Message flow

The event, and all other events with this setting, are emitted only if the message flow commits its unit of work successfully.

If the transaction start event is specified to be included in the message flow unit of work, but the message processing fails and this unit of work is not published, the transaction start event will be included in an independent unit of work. This ensures that your monitoring application receives a pair of events (start and rollback), rather than receiving a rollback event in isolation.

Independent

The event is emitted in a second unit of work, independent of the main unit of work. The event, and all other events with this setting are emitted whether or not the main unit of work commits successfully.

An independent transaction can be started only if the main transaction has been either committed or rolled back. If the **Commit count** property of the flow is greater than one, (Configurable message flow properties), or the **Commit by message group** property is set (Receiving messages in a WebSphere MQ message group), the events targeted for the independent transaction are instead emitted out of sync point, and a message is output stating that this has been done.

None The event is emitted out of sync point (not in any unit of work.) The event is emitted when the message passes through the event source, and is available for reading immediately.

Not all these options are available on all event types. The defaults and allowed values are shown in the following table:

Event source	Allowed values	Default
Transaction start	Message flow Independent None	Message flow
Transaction end	Message flow None	Message flow
Transaction rollback	Independent None	Independent
Terminal	Message flow Independent None	Message flow

12. Click **Finish**.

The **Events** table in the Monitoring tab of the Properties view for the node is updated with details of the event that you just added; the event is enabled.

13. Optional: Disable the event.

14. Save the message flow.

Deploying monitoring properties Procedure

1. When you have added all events to the flow, add the message flow to the broker archive (BAR) file and deploy the BAR file.

Monitoring is inactive for the flow; deploying the BAR file does not activate it.

2. Activate monitoring for the flow by using the `mqsichangeflowmonitoring -c` command.

Updating monitoring properties

About this task

The monitoring properties for a node show all monitoring events that are defined for that node. Edit the monitoring properties of a node to do these tasks:

- Enable or disable a monitoring event.
- Add, delete, or change the monitoring events for the node.

Procedure

1. Right-click the node and select **Properties**.
2. Select the Monitoring tab.
The monitoring events that you have previously defined are displayed.
3. Select or clear the **Enabled** check box for each event as appropriate.
 - To disable an event, clear the check box.
 - To enable an event, select the check box.
4. To add an event, click **Add**.
5. To delete an event, select the event, then click **Delete**.
6. To edit an event, select the event, then click **Edit**.
The Add Event is displayed. For a description of options on this window, see “Creating events” on page 575.
7. Save the message flow.

Related concepts:

“Deciding how to configure monitoring events for message flows” on page 573
Decide whether to use monitoring properties, or a monitoring profile configurable service, to customize the events produced by a message flow.

“Monitoring scenarios” on page 571

Events can be used to support transaction monitoring, transaction auditing and business process monitoring.

Related tasks:



Adding files to a broker archive

To deploy files to an integration server, include them in a broker archive (BAR) file.

“Activating monitoring” on page 583

Use the **mqsichangeflowmonitoring** command to activate monitoring after you have configured monitoring event sources.

“Creating a monitor model for WebSphere Business Monitor V6.2” on page 589

Export monitoring information from IBM Integration Bus to create a monitoring model for WebSphere Business Monitor V6.2

“Creating a monitor model for WebSphere Business Monitor V7 or later” on page 591

To create a monitoring model for WebSphere Business Monitor V7 or later, export monitoring information from IBM Integration Bus.

Related reference:



mqsichangeflowmonitoring command

Use the **mqsichangeflowmonitoring** command to enable monitoring of message flows.



Example XPath expressions for event filtering

Use numeric, string, or Boolean expressions when configuring an event source, to

determine whether the event is emitted.

Related information:



Correlation and monitoring events

A monitoring application uses correlation attributes to identify events that belong to the same business transaction.

Configuring monitoring event sources using a monitoring profile

You can create a monitoring profile and use the `mqsichangeflowmonitoring` command to configure your message flows to emit monitoring events.

Before you begin

Before you start:

Read the following topics:

- “Business-level monitoring” on page 565
- “Monitoring basics” on page 568

You must have a message flow that contains a node to which you want to add a monitoring event.

You can use XPath 1.0 expressions to configure a monitoring event.

Creating a monitoring profile

About this task

First create a monitoring profile XML file. This is a file that lists the event sources in the message flow that will emit events, and defines the properties of each event.

Procedure

Follow the guidance at Monitoring profile to create your monitoring profile XML file.

Applying a monitoring profile

About this task

When you have created a monitoring profile XML file, follow these steps to apply it.

Procedure

1. Use the `mqsicreateconfigurableservice` command to create a configurable service for the monitoring profile. In the following command example, replace *myBroker* with the name of your broker, and *myMonitoringProfile* with the name of your monitoring profile.

```
mqsicreateconfigurableservice myBroker -c MonitoringProfiles  
-o myMonitoringProfile
```

2. Use the `mqsichangeproperties` command to associate your monitoring profile XML file with the configurable service. In the following command example, replace *myBroker* with the name of your broker, *myMonitoringProfile* with the name of your monitoring profile, and *myMonitoringProfile.xml* with the name of the monitoring profile XML file.


```
mqsichangeproperties myBroker -c MonitoringProfiles -o myMonitoringProfile
-n profileProperties -p myMonitoringProfile.xml
```

Set the `useParserNameInMonitoringPayload` property to `TRUE` to force the `wmb:applicationData/wmb:complexContent/wmb:elementName` attribute to hold the name of the input node parser, if present. See `MonitoringProfiles` configurable service for details.

```
mqsichangeproperties myBroker -c MonitoringProfiles -o myMonitoringProfile
-n useParserNameInMonitoringPayload -v TRUE
```

3. Use the `mqsichangeflowmonitoring` command to apply a monitoring profile configurable service to one or more message flows.

- Apply a monitoring profile to a single message flow `messageflow1` in integration server `IS1`:

```
mqsichangeflowmonitoring myBroker -e IS1
-f messageflow1 -m myMonitoringProfile
```

- Apply a monitoring profile to all message flows in all integration servers:

```
mqsichangeflowmonitoring myBroker -g -j -m myMonitoringProfile
```

Monitoring for the flow is inactive; applying the monitoring profile does not activate it.

4. Alternatively, use the broker archive editor to apply a monitoring profile configurable service to one or more message flows, by setting message flow property **Monitoring Profile Name**.

- a. In the IBM Integration Toolkit, switch to the Integration Development perspective.

- b. In the Application Development view, right-click the BAR file, then click **Open with > Broker Archive Editor**.

- c. Click the **Manage and Configure** tab.

- d. Click the message flow on which you want to set the monitoring profile configurable service. The properties that you can configure for the message flow are displayed in the **Properties** view.

- e. In the **Monitoring Profile Name** field, enter the name of a monitoring profile.

- f. Save the BAR file.

- g. Deploy the BAR file.

Monitoring for the flow is inactive; deploying the BAR file does not activate it.

5. Activate monitoring for the flow using the `mqsichangeflowmonitoring -c` command.

- Activate monitoring for a single message flow `messageflow1` in integration server `IS1`:

```
mqsichangeflowmonitoring myBroker -e IS1
-f messageflow1 -c active
```

- Activate monitoring for all message flows in all integration servers:

```
mqsichangeflowmonitoring myBroker -g -j -c active
```

Updating a monitoring profile Procedure

1. Follow the guidance at `Monitoring profile` to update your monitoring profile XML file.
2. Use the `mqsichangeproperties` command to update the configurable service to use the new XML file. For example:

```
mqschangeproperties myBroker -c MonitoringProfiles -o myMonitoringProfile  
-n profileProperties -p myMonitoringProfile.xml
```

3. You must stop and start the integration server for the changes of property values to take effect.

Related concepts:

“Monitoring basics” on page 568

Message flows can be configured to emit events. The events can be read and used by other applications for transaction monitoring, transaction auditing, and business process monitoring.



Configurable services

Configurable services are typically runtime properties. You can use them to define properties that are related to external services on which the broker relies; for example, an SMTP server or a JMS provider.

Related tasks:

“Business-level monitoring” on page 565

You can configure your message flow to emit event messages that can be used to support transaction monitoring and auditing, and business process monitoring.

“Activating monitoring” on page 583

Use the **mqschange`flow`monitoring** command to activate monitoring after you have configured monitoring event sources.

Related reference:



Monitoring profile

To customize events after a message flow has been deployed, but without redeploying the flow, use a monitoring profile configurable service. By using this service, you can apply a monitoring profile to one or more message flows.



mqschange`properties` command

Use the **mqschange`properties`** command to modify broker properties and properties of broker resources.



mqscreate`configurable`service command

Use the **mqscreate`configurable`service** command to create an object name for a broker external resource.



mqschange`flow`monitoring command

Use the **mqschange`flow`monitoring** command to enable monitoring of message flows.



mqsireport`flow`monitoring command

Use the **mqsireport`flow`monitoring** command to display the current options for monitoring that have been set using the **mqschange`flow`monitoring** command.



MonitoringProfiles configurable service

Select the objects and properties that you want to change for the MonitoringProfiles configurable service.



Example XPath expressions for event filtering

Use numeric, string, or Boolean expressions when configuring an event source, to determine whether the event is emitted.

Activating monitoring

Use the `mqsichangeflowmonitoring` command to activate monitoring after you have configured monitoring event sources.

Before you begin

Before you start:

You must have configured monitoring event sources by using either monitoring properties or a monitoring profile; see these topics:

- “Configuring monitoring event sources using monitoring properties” on page 575
- “Configuring monitoring event sources using a monitoring profile” on page 580

About this task

The use of a monitoring profile configurable service overrides the monitoring properties of a message flow.

If monitoring is activated for a message flow, and neither monitoring properties nor a monitoring profile configurable service have been configured for the flow, the default behavior is for transaction events to be emitted from each input node of the message flow. The events contain the bit stream of the input message.

Activating monitoring from the command line

About this task

Use the `-c active` parameter. You can activate monitoring for all message flows in all integration servers, or specify the integration servers and message flows for which monitoring is to be activated.

Procedure

- To activate monitoring for all message flows in all integration servers on Linux, UNIX, and Windows: `Linux` `UNIX` `Windows`

```
mqsichangeflowmonitoring myBroker -c active -g -j
```

To activate monitoring for all message flows in all integration servers on z/OS:

```
z/OS
```

```
F MI10BRK,cm c=active,g=yes,j=yes
```

- To activate monitoring for all message flows in the default integration server on Linux, UNIX, and Windows: `Linux` `UNIX` `Windows`

```
mqsichangeflowmonitoring myBroker -c active -e default -j
```

To activate monitoring for all message flows in the default integration server on z/OS: `z/OS`

```
F MI10BRK,cm c=active,g=default,j=yes
```

- To activate monitoring for the myFlow message flow in the default integration server on Linux, UNIX, and Windows: `Linux` `UNIX` `Windows`

```
mqsichangeflowmonitoring myBroker -c active -e default -f myFlow
```

To activate monitoring for the myFlow message flow in the default integration server on z/OS: `z/OS`

F MI10BRK,cm c=active,g=default,f=myFlow

- To activate monitoring for all message flows in application *application1* in integration server *default* on Linux, UNIX, and Windows: Linux UNIX

Windows

```
mqsichangeflowmonitoring IB9NODE -c active -e default -k application1 -j
```

To activate monitoring for all message flows in application *application1* in integration server *default* on z/OS: z/OS

F IB9NODE,cm c=active,e=default,k=application1,j=yes

- To activate monitoring for message flow *myflow1* in library *library1*, referenced by application *application1*, in integration server *default*, on Linux, UNIX, and Windows: Linux UNIX Windows

```
mqsichangeflowmonitoring IB9NODE -c active -e default -y library1 -k application1 -f myflow1
```

Activate monitoring for message flow *myflow1* in library *library1*, referenced by application *application1*, in integration server *default*, on z/OS: z/OS

F IB9NODE,cm c=active,e=default,y=library1,k=application1,f=myflow1

Deactivating monitoring from the command line

About this task

Use the **-c inactive** parameter. You can deactivate monitoring for all message flows in all integration servers, or specify the integration servers and message flows for which monitoring is to be activated.

Procedure

- To deactivate monitoring for all message flows in all integration servers on Linux, UNIX, and Windows: Linux UNIX Windows

```
mqsichangeflowmonitoring myBroker -c inactive -g -j
```

To deactivate monitoring for all message flows in all integration servers on z/OS: z/OS

F MI10BRK,cm c=inactive,g=yes,j=yes

- To deactivate monitoring for all message flows in the default integration server on Linux, UNIX, and Windows: Linux UNIX Windows

```
mqsichangeflowmonitoring myBroker -c inactive -e default -j
```

To deactivate monitoring for all message flows in the default integration server on z/OS: z/OS

F MI10BRK,cm c=inactive,g=default,j=yes

- To deactivate monitoring for the myFlow message flow in the default integration server on Linux, UNIX, and Windows: Linux UNIX Windows

```
mqsichangeflowmonitoring myBroker -c inactive -e default -f myFlow
```

To deactivate monitoring for the myFlow message flow in the default integration server on z/OS: z/OS

F MI10BRK,cm c=inactive,g=default,f=myFlow

- To deactivate monitoring for all message flows in application *application1* in integration server *default* on Linux, UNIX, and Windows: Linux UNIX

Windows

```
mqsichangeflowmonitoring IB9NODE -c inactive -e default -k application1 -j
```

To activate monitoring for all message flows in application *application1* in integration server *default* on z/OS: z/OS

```
F IB9NODE,cm c=inactive,e=default,k=application1,j=yes
```

- To deactivate monitoring for message flow *myflow1* in library *library1*, referenced by application *application1*, in integration server *default*, on Linux, UNIX, and Windows: Linux UNIX Windows

```
mqsichangeflowmonitoring IB9NODE -c inactive -e default -y library1 -k application1 -f myflow1
```

Activate monitoring for message flow *myflow1* in library *library1*, referenced by application *application1*, in integration server *default*, on z/OS: z/OS

```
F IB9NODE,cm c=inactive,e=default,y=library1,k=application1,f=myflow1
```

Related tasks:

“Configuring monitoring event sources using monitoring properties” on page 575
In the Message flow Editor, use the Monitoring tab on the properties of a node to add one or more monitoring events.

“Configuring monitoring event sources using a monitoring profile” on page 580
You can create a monitoring profile and use the **mqsichangeflowmonitoring** command to configure your message flows to emit monitoring events.

Related reference:



mqsichangeflowmonitoring command

Use the **mqsichangeflowmonitoring** command to enable monitoring of message flows.

Enabling and disabling event sources

When events have been configured for a message flow and deployed to the broker, you can enable and disable individual events. You can do this from the command line, without having to redeploy the message flow, or you can do this from the Message Flow editor, in which case you must redeploy the flow.

Enabling and disabling events from the command line

Before you begin

Before you start:

You must have configured events by using either monitoring properties in the Message Flow editor, or a monitoring profile configurable service. For more information, see “Deciding how to configure monitoring events for message flows” on page 573.

Procedure

To enable and disable event sources, use the **mqsichangeflowmonitoring** command, as shown in the following example.

```

mqsichangeflowmonitoring WBRK_BROKER
  -e default
  -f myMessageFlow
  -s "SOAP Input1.terminal.out,MQOutput1.terminal.in"
  -i enable

mqsichangeflowmonitoring WBRK_BROKER
  -e default
  -f myMessageFlow
  -s "SOAP Input1.terminal.catch"
  -i disabled

```

You can enable or disable multiple events at once; the change of state takes place immediately.

If you configured events by using monitoring properties, the change persists if the message flow is restarted, but is lost if the message flow is redeployed. To make the change permanent, you must also update the monitoring properties.

Tip: When specifying values for the **-s** parameter, use the **Event source address** property of the event, not the **Event name** property.

Tip: To find the list of configured event sources for a message flow, use the **mqsi**report**flow**monitoring command, as shown in the following example:

```

mqsireportflowmonitoring WBRK_BROKER
  -e default
  -f myMessageFlow
  -n

```

Tip: If you configured events by using monitoring properties, you can see a list of the configured event sources in the Message Flow editor by completing the following steps.

1. Open the message flow by using the Message Flow editor.
2. Click the canvas.
3. Select the **Monitoring** tab in the Properties view.

The monitoring events that you have defined previously are shown.

Enabling and disabling events from the Message Flow editor

Before you begin

Before you start:

You must have configured events by using the monitoring properties of the Message Flow editor.

About this task

To enable and disable event sources by using the Message Flow editor, complete the following steps.

Procedure

1. Open the message flow in the editor.
2. Click the canvas.
3. Select the **Monitoring** tab in the Properties view. The monitoring events that you have defined previously are shown.
4. Select **Enabled** for each event that you want to enable.

To disable a single event, clear the check box for that event. To disable all events, click **Uncheck All**.

5. Save the message flow.
6. Rebuild and redeploy the BAR file.

Related concepts:

“Monitoring basics” on page 568

Message flows can be configured to emit events. The events can be read and used by other applications for transaction monitoring, transaction auditing, and business process monitoring.

Related tasks:

“Business-level monitoring” on page 565

You can configure your message flow to emit event messages that can be used to support transaction monitoring and auditing, and business process monitoring.

Related reference:



mqsichangeflowmonitoring command

Use the **mqsichangeflowmonitoring** command to enable monitoring of message flows.



mqsireportflowmonitoring command

Use the **mqsireportflowmonitoring** command to display the current options for monitoring that have been set using the **mqsichangeflowmonitoring** command.

Monitoring flows by using WebSphere Business Monitor

You can monitor message flows by using WebSphere Business Monitor.

Before you begin

Before you start:

- Install WebSphere Business Monitor, and create and start a monitor server. To use the Monitor Model Editor (MME), install the WebSphere Business Monitor Development Toolkit. For full instructions, see IBM WebSphere Business Monitor Version 7.0 product documentation.

About this task

IBM WebSphere Business Monitor is business activity monitoring software that provides a real-time view of your business processes and operations. It contains personalized business dashboards that calculate and display key performance indicators (KPIs) and metrics derived from business processes, business activity data, and business events from a wide range of information sources, including IBM Integration Bus message flows. You can get immediate insight into your business operations so that you can mitigate problems or take immediate advantage of opportunities, resulting in cost savings and increased revenues.

Before you export a message flow, ensure that the message flow is not empty and has no error markers in the Problems view. Also, consider the following requirements of WebSphere Business Monitor:

- WebSphere Business Monitor must be able to identify the start and end of a monitoring context. Always define transaction events (`transaction.Start`, `transaction.End`, and `transaction.Rollback`) on message flows that are monitored by WebSphere Business Monitor.

- The WebSphere Business Monitor toolkit does not support local elements with anonymous types. The export monitoring information option therefore does not generate an event part for event payload XPath queries that resolve to an element of this type. You see a warning message in the report log *flowProjectName_batchgen_report.txt*.
- The WebSphere Business Monitor toolkit does not support creating metrics of type `xs:anyType`. If an XPath expression in your event payload resolves to an element of type `xs:anyType`, the export monitoring information option creates an event part of this type, but you cannot create a metric of this type in the WebSphere Business Monitor toolkit. Create an event part with a supported type; see *Defining Event Parts* in the WebSphere Business Monitor information center.
- If you include unmodeled data, the export monitoring information option cannot type the data. It assigns a type of `string` to the data.
- The option to export monitoring information does not support XPath queries that contain wildcards.

Tip: To identify transaction.Start and transaction.End events separately that are issued following message flow error handling, create an event source on the Failure terminal of the Input node.

A sample is provided with IBM Integration Bus that shows how you can configure WebSphere Business Monitor to receive events from IBM Integration Bus. The sample demonstrates the following steps.

- Installing a message-driven bean (MDB) into WebSphere Business Monitor
- Importing monitoring information about a message flow
- Deploying a monitor model to WebSphere Business Monitor
- Creating a WebSphere Business Monitor dashboard
- Sending messages through a flow and viewing the events that are generated in the WebSphere Business Monitor dashboard

For more information about the sample, including detailed instructions for these steps, see *WebSphere Business Monitor*. (You can run samples only when you use the information center that is integrated with the IBM Integration Toolkit.)

To monitor message flows from IBM Integration Bus, generate a monitor model in the WebSphere Business Monitor development toolkit. Follow the instructions in the appropriate topic.

Procedure

- “Creating a monitor model for WebSphere Business Monitor V6.2” on page 589
- “Creating a monitor model for WebSphere Business Monitor V7 or later” on page 591

Related tasks:



Message Sets: Generating XML Schemas

You can generate either a single XML Schema from a message definition file, or multiple XML Schemas from a message set.

“Business-level monitoring” on page 565

You can configure your message flow to emit event messages that can be used to support transaction monitoring and auditing, and business process monitoring.

“Resolving problems when you monitor message flows” on page 827

Follow the advice for dealing with common problems that can arise when you are

monitoring IBM Integration Bus flows.

Related reference:



XPath expressions that are not suitable for the export monitoring information option

Some XPath expressions produce a warning on the Monitoring tab.

Creating a monitor model for WebSphere Business Monitor V6.2

Export monitoring information from IBM Integration Bus to create a monitoring model for WebSphere Business Monitor V6.2

About this task

In IBM Integration Bus

Procedure

1. If an event contains complex data extracted from a message, supply details of the structure of the data to WebSphere Business Monitor:
 - a. Ensure that the complex data is modeled as a complex type in a message definition file within a message set in the IBM Integration Toolkit.
 - b. In the Application Development perspective, right-click the message set folder and click **Generate XML Schemas** to create a .zip file of XML schemas from the definitions in the message set.
 - c. Extract the file containing the XML schemas on the computer where you will be using the Monitor Model Editor.
2. Export the WMBEvent.xsd file from an existing message set (or create a message set, then export it from there):
 - a. In the Application Development perspective, right-click the message set and click **New > Message Definition File From > IBM Supplied Message**.
 - b. In the window that is displayed, scroll down and click **Message Broker Monitoring Event**.
 - c. Click **Finish**.
 - d. Right-click the message set again, and click **Generate > XML Schemas**.
 - e. In the window that is displayed, click **Export to an external directory**, then enter or browse to a directory.
 - f. Click **Finish**. A compressed file containing the WMBEvent.xsd file is created in the directory that you specified.

The WMBEvent.xsd file gives WebSphere Business Monitor details of the structure of the IBM Integration Bus event.

What to do next

In WebSphere Business Monitor V6.2

This section outlines the steps you need to take in WebSphere Business Monitor. See the documentation for WebSphere Business Monitor for full and up-to-date details.

1. In the WebSphere Business Monitor development toolkit, create a Monitor Model.
Import the WMBEvent.xsd schema and any schemas describing complex data that were exported from IBM Integration Bus, then create a monitor model. You

see errors, because the model is currently empty. You also see a key, which you can rename, for example to LocalTransactionID.

2. Create WebSphere Business Monitor inbound events.

A *monitoring context definition* is the term used by WebSphere Business Monitor to describe all the data that should be collected about an entity (such as a transaction or business process). Each runtime instance (referred to as a monitoring context instance) collects information from inbound events and stores this information in fields that represent the business measures that a monitoring context collects: metrics, counters, and stopwatches.

You need to define an inbound event to describe each event source defined in your message flow that contains information that you want to monitor. For example, if your message flow has Transaction start, Transaction end and Transaction rollback event sources, define an inbound event for each of these event sources.

You typically define inbound events for these three event sources because they contain information that tells WebSphere Business Monitor when the start and end of the monitoring context instance occurs. You also define inbound events to describe any event sources downstream in the flow that contain data that you want to monitor, for example in the In terminal of the MQOutput node. Creating inbound events typically involves the following actions:

a. Define event parts within the inbound event.

Event parts are XML Schema definition (XSD) types that provide information about the structure of part of an event. For an IBM Integration Bus event, define event parts to describe the different parts of the event that you want to monitor data from. For a description of the event structure, see The monitoring event.

As a minimum, define an event part for the event described by type wmb:event. To monitor data about the source of the event (information about the message flow name, broker name), also define an event part for the eventPointData section described by type wmb:eventPointData. You might also want to define event parts to describe message payload data from the applicationData section of the event.

b. Define a correlation expression.

You typically correlate events on fields from the eventCorrelation section in the IBM Integration Bus event. For example, you could correlate events using the localTransactionId field from the IBM Integration Busevent.

You must also define whether the event should create a monitoring context; create this for a Transaction start event.

c. Optional: Define a filter condition.

Set a filter condition. For example you might want to filter events for a specific broker, integration server, or message flow.

d. Optional: Define the event sequence path.

Select a field in the inbound event that can be used to set the order in which the inbound events are processed. For example, you could use creationTime from the IBM Integration Bus event.

e. Complete the key.

The key uniquely identifies a monitoring context instance. You can select any value for the key; for an IBM Integration Bus event, a typical value is the localTransactionId field from the IBM Integration Bus event.

3. Define the metrics.

Having defined inbound events you can now define your metrics. Metrics hold data from the event in a monitoring context.

You might want to define metrics that hold event source data from the `eventPointData` section of the IBM Integration Bus event, for example broker name or message flow name. You can also define metrics that hold event sequencing information, for example `TimeStarted` and `TimeEnded` metrics, which hold the `creationTime` for Transaction start and Transaction end or Transaction rollback events.

In addition, metrics can be defined to hold application data from the IBM Integration Bus event.

Creating a monitor model for WebSphere Business Monitor V7 or later

To create a monitoring model for WebSphere Business Monitor V7 or later, export monitoring information from IBM Integration Bus.

Before you begin

Before you start:

- For a list of the conditions that must be met before you can export monitoring information, see “Monitoring flows by using WebSphere Business Monitor” on page 587.
- For instructions about using earlier versions of WebSphere Business Monitor, see “Creating a monitor model for WebSphere Business Monitor V6.2” on page 589.
- A sample is provided with IBM Integration Bus that shows how you can configure WebSphere Business Monitor to receive events from IBM Integration Bus. For detailed instructions, see WebSphere Business Monitor. (You can run samples only when you use the information center that is integrated with the IBM Integration Toolkit.)

About this task

To create a monitoring model, you export your message flow as a `.zip` file from IBM Integration Bus, then import it into WebSphere Business Monitor.

Procedure

1. Export your message flow from IBM Integration Bus.
 - a. In the IBM Integration Toolkit, right-click the message flow or integration project, and click **Export**. The Export wizard starts.
 - b. Click **Business Monitoring > Application monitoring information**, then click **Next**.
 - c. Select the flows from which you want to export monitoring information, and specify a file name.
 - d. Optional: Specify that an existing file should be overwritten without warning.
 - e. Click **Finish**.

A `.zip` file is created that contains the following monitoring information about the message flow:

- All monitoring information that is defined in the message flow files
- The `WMEvent.xsd` file, which is the schema for the emitted event
- All `.xsd` files in the message sets that are referenced by the selected flows

A report of the export is written to file `flowProjectName_batchgen.report.txt` in the log directory of the integration project.

2. Import the exported .zip file into WebSphere Business Monitor V7 or later.
 - a. In the Business Monitoring perspective of WebSphere Business Monitor, right-click the Project Explorer view and click **Import**.
 - b. Complete the instructions in the Generate Monitor Model wizard.

Select one or more of the following templates, depending on the event sources in your message flow:

 - **Average Transaction Duration** This template is available if you have created `transaction.Start`, `transaction.End`, and `transaction.Rollback` event sources. If you select this template, a metric and stopwatch are created for you in the monitor model showing the Average Transaction Duration, and a key performance indicator (KPI) is created using the data from this stopwatch.
 - **Number of Failed Transactions** This template is available if you have created `transaction.Rollback` event sources. If you select this template, metrics are created for you in the monitor model showing the number of failed transactions and the failed transaction time (this has a default value of 1 January 9999 01:00:00.) A measure and dimension are also created for use in creating multidimensional reports. See *Defining Dimensions* in the WebSphere Business Monitor information center.
 - **Message Flow Correlation** This template provides information about the broker (such as broker name, integration server name, `parentTransactionID`, `globalTransactionID`). You can display this information in a Business Space Dashboard with other metrics and key performance indicators. Because these metrics are used in the correlation expression for the inbound events defined, select this template only if the events for a specified monitoring context are from the same integration server.

For detailed instructions for using WebSphere Business Monitor to monitor message flows, see *Generating monitor models based on application monitoring information from WebSphere Message Broker* in IBM WebSphere Business Monitor Version 7.0 product documentation.

3. Install and configure a message-driven bean in WebSphere Business Monitor.

For WebSphere Business Monitor to monitor events, they must be submitted to the CEI repository by using a message-driven bean. The message-driven bean, which runs in WebSphere Application Server, subscribes to the event topic and writes events that match its subscription to the CEI repository as Common Base Event events. A message-driven bean is supplied with the WebSphere Business Monitor sample. For instructions about how to install and configure the sample message-driven bean, see *Running the WebSphere Business Monitor sample*.
4. Ensure that the event topic has a subscription registered against it for the message flow.
5. Create a WebSphere Business Monitor dashboard.

For detailed instructions see IBM WebSphere Business Monitor Version 7.0 product documentation. This step is also demonstrated by the sample (Running the WebSphere Business Monitor sample).
6. In IBM Integration Bus, configure event sources by using either monitoring properties or a monitoring profile. For more information, see the appropriate topic:

- “Configuring monitoring event sources using monitoring properties” on page 575
 - “Configuring monitoring event sources using a monitoring profile” on page 580
7. Ensure that the appropriate events are enabled.
For more information, see “Enabling and disabling event sources” on page 585.
 8. In IBM Integration Bus, activate monitoring for the deployed message flow.
For more information, see “Activating monitoring” on page 583.

Results

When you have completed these steps, you can send messages through the deployed message flow and view the events in the WebSphere Business Monitor dashboard.

Related tasks:

“Resolving problems when you monitor message flows” on page 827

Follow the advice for dealing with common problems that can arise when you are monitoring IBM Integration Bus flows.

Related reference:



XPath expressions that are not suitable for the export monitoring information option

Some XPath expressions produce a warning on the Monitoring tab.

Reporting monitoring settings

Use the `mqsireportflowmonitoring` command to report monitoring settings for a flow.

About this task

The following examples show how to report monitoring options for a broker called `BROKER1`, integration server default, and message flow `PurchaseOrder`.

Report configured events for a message flow Procedure

Issue the following command on Linux, Unix, or Windows:

```
Linux      UNIX      Windows
mqsireportflowmonitoring BROKER1 -e default -f PurchaseOrder -n
```

Issue the following command on z/OS: `z/OS`

```
F MI10BRK,rm BROKER1,e='default',f='PurchaseOrder,n='yes''
```

Report all possible events for a message flow Procedure

Issue the following command on Linux, Unix, or Windows:

```
Linux      UNIX      Windows
mqsireportflowmonitoring BROKER1 -e default -f PurchaseOrder -a
```

Issue the following command on z/OS: `z/OS`

```
F MI10BRK,rm e='default',f='PurchaseOrder,a='yes'
```

Report specified events for a message flow Procedure

Issue the following command on Linux, Unix, or Windows:

Linux

UNIX

Windows

```
mqsi reportflowmonitoring BROKER1 -e default -f PurchaseOrder  
-s "MQInput1.transaction.Start,MQOutput1.terminal.catch"
```

Issue the following command on z/OS: z/OS

```
F MI10BRK,rm e='default',f='PurchaseOrder',  
s="MQInput1.transaction.Start,MQOutput1.terminal.catch"
```

Tip: When specifying values for the `-s` parameter, use the **Event source address** property of the event, not the **Event name** property.

Export a message flow's monitoring profile About this task

Use the following command to export a profile to file `myMonProf.xml`

Procedure

Issue the following command on Linux, Unix, or Windows:

Linux

UNIX

Windows

```
mqsi reportflowmonitoring BROKER1 -e default -f PurchaseOrder -x -p myMonProf.xml
```

Issue the following command on z/OS: z/OS

```
F MI10BRK,rm e='default',f='PurchaseOrder',x='yes',p='myMonProf.xml'
```

If monitoring for a message flow is configured using monitoring properties, rather than a monitoring profile configurable service, the command creates and returns the equivalent monitoring profile XML file.

Tip: This is an alternative to using an XML editor to create a monitoring profile XML file.

Related tasks:

“Business-level monitoring” on page 565

You can configure your message flow to emit event messages that can be used to support transaction monitoring and auditing, and business process monitoring.

“Configuring monitoring event sources using monitoring properties” on page 575

In the Message flow Editor, use the Monitoring tab on the properties of a node to add one or more monitoring events.

“Configuring monitoring event sources using a monitoring profile” on page 580

You can create a monitoring profile and use the `mqsi changeflowmonitoring` command to configure your message flows to emit monitoring events.

Related reference:



`mqsi changeflowmonitoring` command

Use the `mqsi changeflowmonitoring` command to enable monitoring of message flows.



`mqsi changeproperties` command

Use the `mqsi changeproperties` command to modify broker properties and

properties of broker resources.



mqsireportflowmonitoring command

Use the **mqsireportflowmonitoring** command to display the current options for monitoring that have been set using the **mqsichangefflowmonitoring** command.

Recording, viewing, and replaying data

Record data that is processed by a message flow, or emitted from WebSphere Application Server. View and replay the data.

About this task

If you configured your message flow to emit event messages, the monitoring events publish selected message data, which you can then view or replay (resubmit for processing). WebSphere Application Server events are typically published to a WebSphere MQ queue, configured by the WebSphere Application Server administrator. The record function subscribes to the published monitoring data, and stores the data in a database. You can then view the data through the web user interface, or replay it by resending the message to an MQ queue.

For more information about recording and replaying data, see “Record and replay” on page 597. For more information about monitoring events, see “Business-level monitoring” on page 565.

To record, view, and replay data, follow these steps:

Procedure

1. (Message flows only.) Configure the monitoring event points in the message flow. For information, see “Deciding how to configure monitoring events for message flows” on page 573.
2. (Message flows only.) Deploy your message flow. For more information, see [Deploying resources](#)
3. (Message flows only.) Configure IBM Integration Bus to record data. For more information, see “Recording data” on page 601.
4. Configure IBM Integration Bus to view recorded data through the web user interface or programmatically. For more information, see “Viewing recorded data” on page 616.
5. Replay data to a WebSphere MQ queue, by using the web user interface, the REST API, or the IBM Integration API. For more information, see “Replaying data” on page 618.
6. If you want to view or replay recorded data through the web user interface, you must create a web user account by using the **mqsiwebuseradmin** command. See “Managing web user accounts” on page 226 and **mqsiwebuseradmin** command.
7. If you want to control users' ability to record, view, and replay data, you must enable broker administration security and also configure security for data capture. For information, see “Enabling security for record and replay” on page 598.
8. If you want to use different brokers for deploying your message flows and for recording data captured from those message flows to a database, you must configure a publish/subscribe relationship between the brokers. For more information, see “Using multiple brokers for record and replay” on page 620.

Related concepts:

“Role-based security” on page 220

You can control access to broker resources by associating web users with roles.

“Record and replay” on page 597

For audit or problem determination purposes, you can record data to a database, then view it, and replay it.

“Deciding how to configure monitoring events for message flows” on page 573

Decide whether to use monitoring properties, or a monitoring profile configurable service, to customize the events produced by a message flow.

Related tasks:

“Recording and replaying data with a CMP application” on page 107

You can create CMP applications to examine and replay the data that you have recorded and stored by using record and replay.



Representational State Transfer (REST) API

IBM Integration Bus supports the REST management API, and you can use HTTP clients to administer broker resources.

“Administering brokers using the web user interface” on page 127

You can use the IBM Integration Bus web user interface to administer broker resources.

“Configuring the web user interface server” on page 128

To enable access to broker resources through the web user interface, configure the IBM Integration Bus web user interface server.

“Accessing the web user interface” on page 133

You can access broker resources by logging on to the web user interface.

“Managing web user accounts” on page 226

You can control a web user's access to broker resources by associating the web user ID with a role, which has security permissions assigned to it.

“Using multiple brokers for record and replay” on page 620

You can have multiple brokers in your record and replay topology. If you use different brokers for deploying your message flows and for recording data captured from those message flows to a database, then you must configure a publish/subscribe relationship between the brokers.

“Enabling security for record and replay” on page 598

You can restrict the users who can view and replay data for a broker by enabling security.

“Recording data” on page 601

Record data that is flowing through a message flow, or that is emitted by WebSphere Application Server.

“Viewing recorded data” on page 616

You can view a list of recorded messages, or details of individual messages.

“Replaying data” on page 618

When you have recorded data, you can replay it to a WebSphere MQ queue.

“Business-level monitoring” on page 565

You can configure your message flow to emit event messages that can be used to support transaction monitoring and auditing, and business process monitoring.



Deploying resources

Deploy message flow applications to integration servers by sending a broker archive (BAR) file to a broker, which unpacks and stores the contents ready for when your message flows are started.

Related reference:



Configurable services properties

The supplied configurable services, and the configurable services that you create, are defined by their names and properties. You can use the supplied services.



`mqsiwebuseradmin` command

Use the `mqsiwebuseradmin` command to administer user accounts for the web user interface.

Record and replay

For audit or problem determination purposes, you can record data to a database, then view it, and replay it.

If you need an audit record of messages that pass through the broker, you can record those messages. You might also want to record messages if you need to keep a history of messages for development and test purposes, or to help in problem determination.

To determine which data is recorded, you must configure monitoring on the message flow. Data is stored in a database, which can be an Oracle, Microsoft SQL Server, or DB2 database. You must create a data source, then use a configurable service to define the data source name to use when recording data.

After you have recorded data, you can view it in several ways. By using the web interface, you can view a list of recorded messages, or you can view details of a specific message. You can also view recorded data by using the IBM Integration API or the IBM Integration Bus Representational State Transfer (REST) API.

You can replay a message to a WebSphere MQ queue by using the web interface or IBM Integration API. You can also replay the message to another message flow for testing or problem determination.

Recording events from WebSphere Application Server

You can record events emitted by WebSphere Application Server when it is in a service mapping configuration. The WebSphere Application Server administrator must take the following actions:

1. Supply the topic for the `DataCaptureSource`; otherwise configuration in IBM Integration Bus is exactly the same as for other events.
2. Configure a JMS connection to the queue manager.

See WebSphere Application Server documentation for more information.

Related tasks:

“Recording data” on page 601

Record data that is flowing through a message flow, or that is emitted by WebSphere Application Server.

“Viewing recorded data” on page 616

You can view a list of recorded messages, or details of individual messages.

“Replaying data” on page 618

When you have recorded data, you can replay it to a WebSphere MQ queue.

“Recording and replaying data with a CMP application” on page 107

You can create CMP applications to examine and replay the data that you have recorded and stored by using record and replay.

“Creating and configuring a database for recording data” on page 606
To record data, create a database and configure an ODBC definition to it.
Configure your broker so that it can connect to the database.

“Enabling security for record and replay”

You can restrict the users who can view and replay data for a broker by enabling security.

Related reference:



Configurable services properties

The supplied configurable services, and the configurable services that you create, are defined by their names and properties. You can use the supplied services.

Enabling security for record and replay

You can restrict the users who can view and replay data for a broker by enabling security.

Before you begin

Before you start:

For information about how to configure your system to record data, see “Recording data” on page 601.

About this task

If you do not enable security, all users can complete all actions against a broker and all integration servers. To enable administrative security, you must set the **-s** parameter on the **mqsicreatebroker** or **mqsichangebroker** command. The **-s** parameter specifies the administrative security status for the broker; by default, this parameter is set to *inactive*. If you set the **-s** parameter to *active*, administrative security is enabled and only user IDs that you authorize are permitted to complete actions on the broker. When you create an integration server on a broker for which administrative security is enabled, the queue `SYSTEM.BROKER.AUTH.EG` is created. Populate the queue with the appropriate user authorization.

Procedure

To enable security for record and replay, complete the following steps.

1. To use a web administration server to view and replay data, you must create a system user account on your operating system, such as `ibmuser`. This system user account functions as a role, which you can associate with one or more web user accounts. You do not need to create this system user account if you are not enabling broker administration security. For more information about roles, see “Role-based security” on page 220.
2. Enable administrative security by setting the **-s** parameter to *active*.
 - To enable administrative security when you create the broker, run the **mqsicreatebroker**, as shown in the following example:

```
mqsicreatebroker brokerName -q brokerQueueManagerName -s active
```

(If you run this command on Windows, you must also set the **-i** parameter. For details, see **mqsicreatebroker** command.)

- To enable administrative security for a broker that you have already created, stop the broker, then run the `mqsichangebroker`, as shown in the following example:

```
mqsichangebroker brokerName -s active
```

For more information, see “Enabling administration security” on page 222.

3. To allow users with an assigned role to run record and replay queries on the integration server, ensure that the role (system user account) has inquire (inq) administration authority on the queues SYSTEM.BROKER.AUTH and SYSTEM.BROKER.AUTH.EG.

For more information, see “Authorizing users for administration” on page 228.

4. In addition to setting administrative security, you must also set security for data capture. The queue SYSTEM.BROKER.DC.AUTH controls the record and replay actions that users with a specified role (such as `ibmuser`) can complete on the broker. Ensure that the role has the appropriate authorization to complete the following actions on this queue:

Action	Authority required
To view data, bit streams, and exception lists	READ (+INQ)
To replay data	EXECUTE (+SET)

To change these authorizations, you can use WebSphere MQ commands or the IBM Integration Explorer.

To use WebSphere MQ commands, see “Granting and revoking authority on Linux, UNIX, and Windows systems” on page 232.

To use the IBM Integration Explorer, complete the following steps.

- a. In the MQ Explorer - Navigator view, navigate to **IBM WebSphere MQBrokers > Queene Managers**, expand your queue manager, and select **Queues**.
 - b. Right-click the queue SYSTEM.BROKER.DC.AUTH, then click **Object Authorities > Manage Authority Records**.
 - c. Expand **Specific Profiles** and click SYSTEM.BROKER.DC.AUTH.
 - d. On the **Users** tab, select `ibmuser` and click **Edit**.
 - e. Set the appropriate authorizations and click **OK**, then close the Manage Authority Records dialog box.
5. If you change authorizations, restart your broker and broker queue manager to ensure that changes take effect.
 6. Create a web user account by using the `mqswebuseradmin` command. This web user account is the one that you will use to login to the web user interface for viewing and replaying data. If you have broker administration enabled, you must specify a role for the web user account when you create it. The role is the system user account that you created in Step 1, which has security permissions assigned. For more information, see `mqswebuseradmin` command and “Managing web user accounts” on page 226.

What to do next

To disable security, set the `-s` parameter to `inactive` on the `mqsichangebroker` command.

Next:

To view data that has been recorded, see “Viewing recorded data” on page 616.

To replay data that has been recorded, see “Replaying data” on page 618.

Related concepts:

“Administration security overview” on page 214

Broker administration security controls the rights of users to complete administrative tasks for a broker and its resources.

“Record and replay” on page 597

For audit or problem determination purposes, you can record data to a database, then view it, and replay it.

“Role-based security” on page 220

You can control access to broker resources by associating web users with roles.

“Authorization queues for broker administration security” on page 218

If you have enabled broker administration security, the broker examines specific queues to determine if a user has the authority to complete a particular task against a broker or its resources.

“IBM Integration Bus permissions and equivalent WebSphere MQ permissions” on page 217

If you have enabled broker administration security, you can give different permissions to user IDs to allow them to complete various actions against a broker or its resources.

Related tasks:

“Authorizing users for administration” on page 228

Grant authority to one or more groups or users to authorize them to complete specific tasks against a broker and its resources.

“Recording data” on page 601

Record data that is flowing through a message flow, or that is emitted by WebSphere Application Server.

“Viewing recorded data” on page 616

You can view a list of recorded messages, or details of individual messages.

“Replaying data” on page 618

When you have recorded data, you can replay it to a WebSphere MQ queue.

“Managing web user accounts” on page 226

You can control a web user's access to broker resources by associating the web user ID with a role, which has security permissions assigned to it.

Related reference:



mqsicreatebroker command

Use the **mqsicreatebroker** command to create a broker and its associated resources.



mqsichangebroker command

Use the **mqsichangebroker** command to change one or more of the configuration parameters of the broker.



mqswebuseradmin command

Use the **mqswebuseradmin** command to administer user accounts for the web user interface.



Tasks and authorizations for administration security

If you have enabled broker administration security, users require specific authority so that they can complete administration tasks.

Recording data

Record data that is flowing through a message flow, or that is emitted by WebSphere Application Server.

- Create and configure a database.
- Configure security settings.
- Create appropriate configurable services.
- Configure monitoring for the message flow.

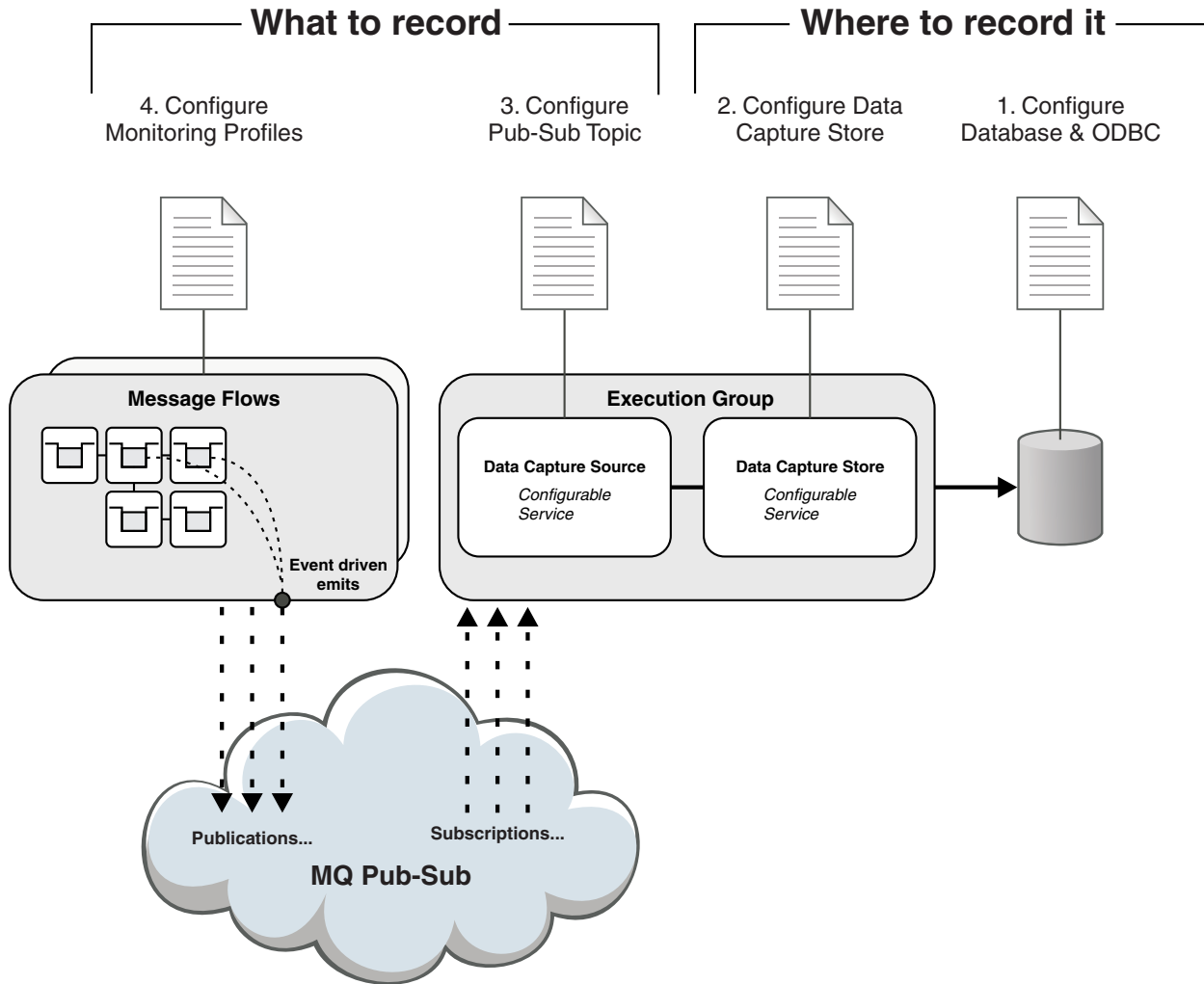
Before you begin

Before you start:

Ensure that the message flow for which you want to record data has been deployed. For more information, see [Deploying resources](#).

About this task

You can record data to a database for audit purposes, or to help with problem determination. To record data, you must identify the source of the data that you want to record and the place that you want to record it to. The steps that you take to record data are shown in the following diagram:



Procedure

To configure IBM Integration Bus to record data, complete the following steps.

1. Create and configure your database, and define an ODBC definition for the data source name (DSN). Specify an ID and password for your broker to use when connecting to the database. See "Creating and configuring a database for recording data" on page 606.
2. Configure your data capture store.

To define how and where data is stored, create a `DataCaptureStore` configurable service. This configurable service specifies the broker runtime properties for data processing and for connecting to the database.

Your record and replay topology can include more than one broker. If you deploy the message flows for which you want to capture data to one broker, and use a different broker to record the data, you must connect the two brokers. For more information about how you can configure your broker topology, see "Using multiple brokers for record and replay" on page 620.

You can use the provided `DefaultCaptureStore` configurable service or create your own configurable service of type `DataCaptureStore`. You can use the IBM Integration Explorer to create the configurable service; for more information, see Using the IBM Integration Explorer to work with configurable services.

Alternatively, use the **mqsicreateconfigurable-service** command; for more information, see **mqsicreateconfigurable-service** command. For descriptions of properties of this configurable service, see DataCaptureStore configurable service.

For example, enter the following command on a command line:

```
mqsicreateconfigurable-service brokerName -c DataCaptureStore -o dataCaptureStoreName  
-n dataSourceName, egForRecord -v dataSource, integrationServer
```

- *brokerName* is the name of your broker. You configured this broker to connect to the database when you completed the steps in the topic “Creating and configuring a database for recording data” on page 606.
 - *dataCaptureStoreName* is the name of your configurable service object.
 - *dataSource* is the name of your data source.
 - *integrationServer* is the name of the integration server that processes data for recording.
3. Specify a publish/subscribe topic that identifies the source of the data that you want to capture.

To identify the source of the data, create a DataCaptureSource configurable service. You use this configurable service to specify the monitoring topic that identifies the messages flows from which your data comes, and the data capture store to use for storing this data. Multiple instances of the DataCaptureSource configurable service can use the same DataCaptureStore configurable service.

You can use the IBM Integration Explorer or the **mqsicreateconfigurable-service** command to create the configurable service. If you use the DataCaptureSourceTemplate in IBM Integration Explorer, you must create a new configurable service based on the template. If you edit the template without creating a new configurable service, an error is issued at run time. For information about the properties of this configurable service, see DataCaptureSource configurable service. For example, on UNIX systems, enter the following command on a command line:

```
mqsicreateconfigurable-service brokerName -c DataCaptureSource -o dataCaptureSourceName  
-n dataCaptureStore, topic  
-v dataCaptureStoreName, '$SYS/Broker/myBroker/Monitoring/integrationServerName/msgFlowName'
```

- *brokerName* is the name of your broker.
- *dataCaptureSourceName* is the name of the configurable service object.
- *dataCaptureStoreName* is the name of the DataCaptureStore configurable service that you want to use for this subscription. You must use *brokerName* to create this DataCaptureStore configurable service.
- *myBroker*, *integrationServerName*, and *msgFlowName* are the names of the broker, integration server, and message flow from which you want to capture data. These values are part of a topic string, which is used to subscribe to events that you set up by using business monitoring. You can use topic wildcards in this topic string. On UNIX systems, enclose the topic string in single quotation marks when you enter it on a command line. On Windows systems, use double quotation marks. No quotation marks are required if you create the configurable service by using the IBM Integration Explorer.

For events that are emitted by WebSphere Application Server, the WebSphere Application Server administrator must supply details of the topic, for example:

```
$SYS/AppServer/exampleCell02Cell/exampleNode03.server1/#
```

For more information about how monitoring is used for capturing data, see “Configuring monitoring for data capture” on page 613.

Test that your subscription to the topic specified in the topic property was successful by retrieving the subscriptions on the queue manager for *brokerName*. Use IBM Integration Explorer or the **runmqsc** command.

To check the subscriptions by using the IBM Integration Explorer, complete the following steps:

- a. Expand the queue manager under the **Queue Managers** folder
- b. To open the Subscriptions pane, click **Subscriptions**.
- c. Click **Refresh** and check that a subscription with a topic string of `$SYS/Broker/myBroker/Monitoring/integrationServerName/msgFlowName` exists

To check the subscription by using **runmqsc**, complete the following steps:

- a. At a command prompt, type `runmqsc qmName`, where *qmName* is your queue manager name.
 - b. To display all the queue manager subscriptions, type `dis sub(*)`
 - c. Check that the topic name is returned in the list of subscription topics, for example `SUB(myBroker:myTopic)`
 - d. To exit the **runmqsc** environment, type `end`
4. To generate the data that you want to record, configure monitoring on your message flows. See “Configuring monitoring for data capture” on page 613.

What to do next

Next:

- You can enable security for recording data by following the instructions in “Enabling security for record and replay” on page 598.
- After you have recorded data, you can view it. For more information, see “Viewing recorded data” on page 616.
- You might want to turn off recording for a particular integration server temporarily (for performance reasons, for example). For more information, see “Disabling and enabling data capture” on page 605.
- For more information about tuning data capture, see “Tuning data capture” on page 615.
- If the broker that you specified in your `DataCaptureStore` is different to the broker that you specified in your `DataCaptureSource`, you must configure a publish/subscribe relationship between the brokers. For more information, see “Using multiple brokers for record and replay” on page 620.

Related concepts:

“Record and replay” on page 597

For audit or problem determination purposes, you can record data to a database, then view it, and replay it.

Related tasks:

“Viewing recorded data” on page 616

You can view a list of recorded messages, or details of individual messages.

“Replaying data” on page 618

When you have recorded data, you can replay it to a WebSphere MQ queue.

“Recording and replaying data with a CMP application” on page 107

You can create CMP applications to examine and replay the data that you have recorded and stored by using record and replay.

Related reference:



Configurable services properties

The supplied configurable services, and the configurable services that you create, are defined by their names and properties. You can use the supplied services.



DataCaptureSource configurable service

Select the objects and properties that you want to change for the DataCaptureSource configurable service.



DataCaptureStore configurable service

Select the objects and properties that you want to change for the DataCaptureStore configurable service.

Disabling and enabling data capture

Recording of data is enabled by default, but takes effect only if you define a DataCaptureSource configurable service. You can disable recording for all of the data capture sources on an integration server by modifying the broker runtime properties for that integration server.

About this task

You might want to temporarily stop recording for an integration server for performance reasons, or to modify the integration server. You can also use these instructions to start recording again.

Procedure

You can use the IBM Integration Explorer or IBM Integration Bus runtime commands to disable recording for your integration server. Complete the following steps.

Review and edit the current setting on the ComIbmDataCaptureManager resource manager.

- If you are using the IBM Integration Explorer, complete the following steps:
 1. In the MQ Explorer - Navigator view, navigate to **IBM WebSphere MQ > Brokers > brokerName**, where *brokerName* is the name of your broker. Right-click your integration server name, and select **Properties**.
 2. Select the **DataCapture** tab.
 3. Review and set the **DataCapture Enabled** property by selecting either true or false from the menu.
 4. Click **Apply** to review your changes and click **OK** to close the Properties window.

Your change takes effect immediately.

- If you are using the runtime commands, complete the following steps:
 1. Enter the following command on an IBM Integration Bus command line, where *brokerName* is the name of your broker and *integrationServer* is the name of the integration server:

```
mqsireportproperties brokerName
-e integrationServer -o ComIbmDataCaptureManager -a
```

The command produces a response similar to this example:

```
ComIbmDataCaptureManager
uid='ComIbmDataCaptureManager'
userTraceLevel='none'
```

```
traceLevel='none'  
userTraceFilter='none'  
traceFilter='none'  
enabled='true'
```

2. Change the enabled property to false, by entering the following command:

```
mqsichangeproperties brokerName  
-e integrationServer -o ComIbmDataCaptureManager -n enabled -v false
```

Your change takes effect immediately.

What to do next

Next:

When you are ready to start recording again for your integration server, use the **mqsichangeproperties** command to change the enabled property to true, and restart the broker.

Related concepts:

“Record and replay” on page 597

For audit or problem determination purposes, you can record data to a database, then view it, and replay it.

Related tasks:

“Starting and stopping a broker” on page 19

Run the appropriate command to start or stop a broker.

Related reference:



mqsireportproperties command

Use the **mqsireportproperties** command to display properties that relate to a broker, an integration server, or a configurable service.



mqsichangeproperties command

Use the **mqsichangeproperties** command to modify broker properties and properties of broker resources.

Creating and configuring a database for recording data

To record data, create a database and configure an ODBC definition to it. Configure your broker so that it can connect to the database.

Before you begin

Before you start:

Read the concept topic “Record and replay” on page 597. For an overview of tasks you must complete to record data, see “Recording data” on page 601.

About this task

You can record data to a database for audit purposes, or to help with problem determination.

A script is provided with IBM Integration Bus that you can use to create the database and database tables. You can run this script unmodified, or you can customize it. The script creates a database called MBRECORD with a default schema. After creating the database, you create an ODBC definition to identify the

data source name (DSN) for the database. You use the `mqsisetdbparms` command to set a user identifier and password for the broker to use when connecting to the database.

You can record data to DB2, Microsoft SQL Server, and Oracle databases. For information about how to create and configure your database, see the following topics:

- “Creating and configuring a DB2 database for recording data”
- “Creating and configuring an Oracle database for recording data” on page 609
- “Creating and configuring a Microsoft SQL Server database for recording data” on page 611

What to do next

Next:

Continue to follow the steps for recording data; see “Recording data” on page 601.

Related concepts:

“Record and replay” on page 597

For audit or problem determination purposes, you can record data to a database, then view it, and replay it.

Related tasks:



Configuring databases

Configure databases to hold application or business data that you can access from your message flows.

“Recording data” on page 601

Record data that is flowing through a message flow, or that is emitted by WebSphere Application Server.

Creating and configuring a DB2 database for recording data:

To record data to a DB2 database, create the database and configure an ODBC definition to it. Configure your broker so that it can connect to the database.

Before you begin

Before you start:

Read the following topics:

- “Record and replay” on page 597
- “Recording, viewing, and replaying data” on page 595
- “Creating and configuring a database for recording data” on page 606

About this task

Procedure

1. Use the script that is provided with IBM Integration Bus to create and configure a DB2 database to store your recorded data.
 - a. Locate the script for your operating system:
 - Windows: `install_dir\ddl\db2\DataCaptureSchema.sql`
 - UNIX: `install_dir/ddl/db2/DataCaptureSchema.sql`

- z/OS : the JCL script is in the data set and member SBIPPROC(BIPRRDB)
install_dir is the location of your IBM Integration Bus installation.
- b. Optional: To specify your own database or schema, customize the provided DataCaptureSchema script, and save your changes.
If you modify the SQL to specify a particular schema, you must also set the same schema name in the DataCaptureStore configurable service.
You might also want to edit the script for the following reasons:
 - If you ran the script, and want to run it again, you must drop the database MBRECORD first. Insert the command drop database MBRECORD before the line that reads create database MBRECORD.
 - The maximum message body size that you can record (after encoding has taken place) is 5 MB. The default size is 5 MB, but you can increase this size by editing the script to make the value in the WMB_BINARY_DATA.DATA column larger.
- c. At a command line, navigate to the script location and run it.
On Windows, use a DB2 Command Window to ensure that the command environment is set up correctly. Click **Start > IBM DB2 > databaseInstance > Command Line Tools**, and select **Command Window**, where *databaseInstance* is the DB2 installation name.
On UNIX, a script called db2profile is provided for setting up the environment; for more information, see Command environment: Linux and UNIX systems.
When the command environment is set up, you can run the script.
For example, on Windows or UNIX, enter the following command:
db2 -tvf DataCaptureSchema.sql

2. Create an ODBC definition for the database.
If you used the supplied script to create your database without modifications, create an ODBC definition for the database called MBRECORD, with MBRECORD as the data source name (DSN). For more information, see Enabling ODBC connections to the databases.
3. Use the **mqsisetdbparms** command to set a user identifier and password for the broker to use when connecting to the database; for example:
mqsisetdbparms brokerName -n dataSourceName -u userID -p password
 - *brokerName* is the name of your broker.
 - *dataSourceName* identifies the database to which you want to record data.
 - *userID* and *password* specify the user identifier and the password that the broker uses to connect to the database.
4. To ensure that the changes to the **mqsisetdbparms** command take effect, restart the broker. For more information, see “Starting and stopping a broker” on page 19.
5. Test the connection to your database by using the **mqsicvp** command. For more information, see **mqsicvp** command.

What to do next

Next:

Continue to follow the steps for recording data; for more information, see “Recording data” on page 601.

Related tasks:

Configuring databases

Configure databases to hold application or business data that you can access from your message flows.

“Recording data” on page 601

Record data that is flowing through a message flow, or that is emitted by WebSphere Application Server.

“Creating and configuring a database for recording data” on page 606

To record data, create a database and configure an ODBC definition to it. Configure your broker so that it can connect to the database.

“Creating and configuring an Oracle database for recording data”

To record data to an Oracle database, create the database tables and configure an ODBC definition. Configure your broker so that it can connect to the database.

Command environment: Linux and UNIX systems

Set up the Linux or UNIX environment to run IBM Integration Bus commands.

“Starting and stopping a broker” on page 19

Run the appropriate command to start or stop a broker.

Enabling ODBC connections to the databases

Set up the resources and environment that the broker requires for Open Database Connectivity (ODBC) connections to databases on distributed systems.

Related reference:

mqsicvp command

Use the **mqsicvp** command to perform verification tests on a broker, or to verify ODBC connections.

mqsisetdbparms command

Use the **mqsisetdbparms** command to associate a specific user ID and password (or SSH identity file) with one or more resources that are accessed by the broker.

Configurable services properties

The supplied configurable services, and the configurable services that you create, are defined by their names and properties. You can use the supplied services.

Creating and configuring an Oracle database for recording data:

To record data to an Oracle database, create the database tables and configure an ODBC definition. Configure your broker so that it can connect to the database.

Before you begin

Before you start:

Read the following topics:

- “Record and replay” on page 597
- “Recording, viewing, and replaying data” on page 595
- “Creating and configuring a database for recording data” on page 606

About this task

Procedure

1. Use the script that is provided with IBM Integration Bus to create and configure the Oracle database tables in the default table space, which is where your recorded data will be stored.
 - a. Locate the script for your operating system:
 - Windows: `install_dir\ddl\oracle\DataCaptureSchema.sql`
 - UNIX: `install_dir/ddl/oracle/DataCaptureSchema.sql`

`install_dir` is the location of your IBM Integration Bus installation.
 - b. On Oracle, enter the following command to run the script:
`sqlplus user/password @DataCaptureSchema.sql`
2. Create an ODBC definition for the database.

If you used the supplied script to create your database without modifications, create an ODBC definition for the database called MBRECORD, with MBRECORD as the data source name (DSN). For more information, see Enabling ODBC connections to the databases.
3. Use the **mqsisetdbparms** command to set a user identifier and password for the broker to use when connecting to the database; for example:
mqsisetdbparms *brokerName* -n *dataSourceName* -u *userID* -p *password*
 - *brokerName* is the name of your broker
 - *dataSourceName* identifies the database to which you want to record data
 - *userID* and *password* specify the user identifier and the password that the broker uses to connect to the database
4. To ensure that the changes to the **mqsisetdbparms** command take effect, restart the broker. For more information, see “Starting and stopping a broker” on page 19.
5. Test the connection to your database by using the **mqsicvp** command. For more information, see **mqsicvp** command.

What to do next

Next:

Continue to follow the steps for recording data; for more information, see “Recording data” on page 601.

Related tasks:



Configuring databases

Configure databases to hold application or business data that you can access from your message flows.

“Recording data” on page 601

Record data that is flowing through a message flow, or that is emitted by WebSphere Application Server.

“Creating and configuring a database for recording data” on page 606

To record data, create a database and configure an ODBC definition to it. Configure your broker so that it can connect to the database.

“Creating and configuring a DB2 database for recording data” on page 607

To record data to a DB2 database, create the database and configure an ODBC definition to it. Configure your broker so that it can connect to the database.



Command environment: Linux and UNIX systems

Set up the Linux or UNIX environment to run IBM Integration Bus commands.

“Starting and stopping a broker” on page 19

Run the appropriate command to start or stop a broker.



Enabling ODBC connections to the databases

Set up the resources and environment that the broker requires for Open Database Connectivity (ODBC) connections to databases on distributed systems.

Related reference:



mqsicvp command

Use the **mqsicvp** command to perform verification tests on a broker, or to verify ODBC connections.



mqsisetdbparms command

Use the **mqsisetdbparms** command to associate a specific user ID and password (or SSH identity file) with one or more resources that are accessed by the broker.



Configurable services properties

The supplied configurable services, and the configurable services that you create, are defined by their names and properties. You can use the supplied services.

Creating and configuring a Microsoft SQL Server database for recording data:

To record data to a Microsoft SQL Server database, create the database, and configure an ODBC definition to it. Configure your broker so that it can connect to the database.

Before you begin

Before you start:

Read the following topics:

- “Record and replay” on page 597
- “Recording, viewing, and replaying data” on page 595
- “Creating and configuring a database for recording data” on page 606

Procedure

1. Use the script that is provided with IBM Integration Bus to create and configure an SQL Server database to store your recorded data.
 - a. Locate the script at *install_dir*\ddl\sqlServer\DataCaptureSchema.sql, where *install_dir* is the location of your IBM Integration Bus installation.
 - b. Optional: Customize the provided DataCaptureSchema script.
If you modify the SQL to specify a particular schema, you must also set the same schema name in the DataCaptureStore configurable service.
 - c. To run the script, at a command line, navigate to the script location and enter the following command:

```
sqlcmd -d databaseName -i DataCaptureSchema.sql
```

2. Create an ODBC definition for the database.

If you used the supplied script to create your database without modifications, create an ODBC definition for the database that is called MBRECORD, with

MBRECORD as the data source name (DSN). For more information, see Enabling ODBC connections to the databases.

3. Use the **mqsisetdbparms** command to set a user identifier and password for the broker to use when it connects to the database. For example:

```
mqsisetdbparms brokerName -n dataSourceName -u userID -p password
```

- *brokerName* is the name of your broker.
- *dataSourceName* identifies the database to which you want to record data.
- *userID* and *password* specify the user identifier and the password that the broker uses to connect to the database.

4. To ensure that the changes to the **mqsisetdbparms** command take effect, restart the broker. For more information, see “Starting and stopping a broker” on page 19.
5. Test the connection to your database by using the **mqsicvp** command. For more information, see **mqsicvp** command.

What to do next

Next:

Follow the steps for recording data (see “Recording data” on page 601).

Related tasks:



Configuring databases

Configure databases to hold application or business data that you can access from your message flows.

“Recording data” on page 601

Record data that is flowing through a message flow, or that is emitted by WebSphere Application Server.

“Creating and configuring a database for recording data” on page 606

To record data, create a database and configure an ODBC definition to it.

Configure your broker so that it can connect to the database.

“Creating and configuring an Oracle database for recording data” on page 609

To record data to an Oracle database, create the database tables and configure an ODBC definition. Configure your broker so that it can connect to the database.



Command environment: Linux and UNIX systems

Set up the Linux or UNIX environment to run IBM Integration Bus commands.

“Starting and stopping a broker” on page 19

Run the appropriate command to start or stop a broker.



Enabling ODBC connections to the databases

Set up the resources and environment that the broker requires for Open Database Connectivity (ODBC) connections to databases on distributed systems.

Related reference:



mqsicvp command

Use the **mqsicvp** command to perform verification tests on a broker, or to verify ODBC connections.



mqsisetdbparms command

Use the **mqsisetdbparms** command to associate a specific user ID and password (or SSH identity file) with one or more resources that are accessed by the broker.



Configurable services properties

The supplied configurable services, and the configurable services that you create, are defined by their names and properties. You can use the supplied services.

Configuring monitoring for data capture

Determine which data you want to record, and configure your message flows to emit this data, by using monitoring events. Learn about the differences between how you configure event sources for data capture and for business-level monitoring.

Before you begin

Before you start:

Create and configure a database, and specify the broker runtime properties that are required for recording data. See “Recording data” on page 601.

About this task

Data capture is based on a publish/subscribe model. To specify the data that you want to record to a database, you configure your message flows to emit event messages. These messages contain the data that you want to record. The messages are published to a topic that you specify in a `DataCaptureSource` configurable service. This configurable service also contains the name of the `DataCaptureStore` configurable service that represents the database to which the data is to be recorded. The broker subscribes to the topic and routes the published messages to the database.

The mechanism that is used for configuring message flows to emit event messages for data capture is also used for business-level monitoring. This mechanism is summarized in the concept topic “Business-level monitoring” on page 565, and is described further on in this topic. Because data capture and business-level monitoring can share event sources and monitoring events, you must ensure that the configuration for one does not inadvertently affect the other. Before you begin to configure monitoring for data capture, you need to be aware of some specific considerations that relate to the use of the monitoring mechanism for capturing data for record and replay.

- Typically, the type of data that you want to capture for record and replay differs from the type of data that is useful for business monitoring. When you are capturing data for record and replay, you probably want to capture part or all of the message bitstream data. However, if you are monitoring transactions or business processes, you might want to include only particular message fields that represent significant thresholds. See “Monitoring scenarios” on page 571. The requirements for data capture and for business-level monitoring might therefore conflict with one another.
- When you view your recorded data in the web administration **Data Viewer**, you might see one or more errors in the **Exception** column. You can retrieve the exception data for these errors only if you included the exception list (`$ExceptionList`) when you created your monitoring events. However, if the monitoring events are exported in order to create a monitoring model for WebSphere Business Monitor, the inclusion of the exception list results in a warning. See “Creating a monitor model for WebSphere Business Monitor V7 or later” on page 591.
- You can record messages from many different providers and sources. However, you can replay messages to WebSphere MQ destinations only.

When you configure a monitoring event, you can choose to include all of the message bitstream data, the headers only, or the message body only. When a broker replays a message, it needs to include a WebSphere MQ message header. It handles this requirement in one of the following ways:

- If the captured data includes a WebSphere MQ header, the broker uses the existing WebSphere MQ message header.
- If the captured data does not include a WebSphere MQ header, the broker generates a WebSphere MQ header. All other headers are appended to the message payload.

Procedure

To configure monitoring on a message flow to emit events for capture, use one of the following methods.

- Configure and enable event sources, and activate monitoring for the message flow, by completing the steps in “Business-level monitoring” on page 565.
You do not need to subscribe to the monitoring topic; the broker manages the subscription to the topic specified in the DataCaptureSource configurable service.
- Configure monitoring event sources by using the sample monitoring profile that is provided at *install_dir/sample/RecordReplay/basicMonitoringProfile.xml*.
The sample monitoring profile works for a single input node. You must update the profile to replace *NODENAME* with the name of your input node. To enable this monitoring profile, run the **mqsicreateconfigurableservice**, **mqsichangeproperties**, and **mqsichangeflowmonitoring** commands against the broker to which you deployed the flow. For details about running these commands, see “Configuring monitoring event sources using a monitoring profile” on page 580.

The subscription to the monitoring events is durable, which results in messages being put to the queue specified in the *queueName* property of the DataCaptureStore configurable service. Messages can be put to this queue even when the broker is not running. For more information about how you can monitor the queue depth and tune data capture, see “Tuning data capture” on page 615.

What to do next

Next:

Review the steps that you completed for recording data; see “Recording data” on page 601. Now you are ready to view the data that you recorded; see “Viewing recorded data” on page 616.

Related tasks:

“Business-level monitoring” on page 565

You can configure your message flow to emit event messages that can be used to support transaction monitoring and auditing, and business process monitoring.

“Recording, viewing, and replaying data” on page 595

Record data that is processed by a message flow, or emitted from WebSphere Application Server. View and replay the data.

“Viewing recorded data” on page 616

You can view a list of recorded messages, or details of individual messages.

“Recording data” on page 601

Record data that is flowing through a message flow, or that is emitted by WebSphere Application Server.

Tuning data capture

You can change the broker configuration to improve performance and message throughput when you are recording data. You can configure the number of integration servers and queues used to process the data, the size of the thread pool that is used by each integration server, and how often data is committed. You can also use more than one database for storing data.

About this task

Consider each of the following factors when designing your broker configuration for record and replay. For further information about the `DataCaptureStore` configurable service properties, see `DataCaptureStore` configurable service.

`threadPoolSize`

This property is set on the `DataCaptureStore` configurable service. The property determines the number of threads that are used by the integration server specified in the `egForRecord` property to process subscriptions to the monitoring topic.

`commitCount` and `commitIntervalSecs`

These properties are set on the `DataCaptureStore` configurable service. The `commitCount` property identifies the number of input messages that are processed on each thread before a sync point is taken. Decreasing the value of `commitCount` can improve performance. The `commitIntervalSecs` identifies the time interval at which a commit is taken when the `commitCount` property is greater than 1 but the number of messages processed has not reached the value of the `commitCount` property. Increasing the value of `commitIntervalSecs` can improve performance.

`egForRecord` and `egForView`

These properties are set on the `DataCaptureStore` configurable service. You can spread workload by specifying multiple data capture stores with different integration servers for recording and viewing data specified for each one.

`DataCaptureStore` and `DataCaptureSource` configurable service instances

The relationship of the `DataCaptureStore` configurable service to the `DataCaptureSource` configurable service is many to one. To use multiple data sources for storing recorded data, you can define several `DataCaptureSource` configurable services. To increase the numbers of execution groups and queues used to process data for recording and viewing, you can change the ratio of `DataCaptureStore` to `DataCaptureSource` configurable services.

`queueName` and `backoutQueue`

These properties are set on the `DataCaptureStore` configurable service. The `queueName` property specifies the queue that is used for holding data before it is recorded. The `backoutQueue` property specifies the queue that is used for backing out messages that cannot be processed. You can use the default queues or you can create new queues so that the data is spread across multiple queues.

You can configure WebSphere MQ to warn when queue depths are close to their maximum size. For more information, see the WebSphere MQ documentation about “Event monitoring” on the WebSphere MQ Library web page. You can also increase the maximum queue depth and the maximum message length for queues by using either the `runmqsc` command or the IBM Integration Explorer.

To use the **runmqsc** command, complete the following steps.

1. At a command prompt, enter the following command, where *qmName* is the name of your queue manager:
`runmqsc qmName`
2. Confirm the current settings for your queue by running the following command, where *qName* is the name of the relevant queue; for example, SYSTEM.BROKER.DC.RECORD. If your queue name contains lowercase characters, you must enclose the queue name in single quotation marks.
`dis qlocal(qName) maxdepth,maxmsgl`
3. Enter the following command, where *qName* is the name of your queue, *newMaxDepth* is the new setting for maximum queue depth, and *newMaxMsgL* is the new setting for maximum message length:
`alter qlocal(qName) maxdepth(newMaxDepth) maxmsgl(newMaxMsgL)`
4. To exit the **runmqsc** environment, type end.

To use the IBM Integration Explorer, complete the following steps.

1. Under the **Queue Managers** folder, right-click the queue manager and select **Properties**.
2. In the menu tree, select **Extended**.
3. Edit the values for the **Max uncommitted messages** or **Max message length** properties.

Related tasks:

“Performance” on page 450

You can use message flow design and coding techniques to optimize the performance of your message flows. You can then analyze statistical information for your message flows and modify aspects of your broker configuration to tune the performance of your brokers and message flows.

“Recording, viewing, and replaying data” on page 595

Record data that is processed by a message flow, or emitted from WebSphere Application Server. View and replay the data.

Related reference:



Configurable message flow properties

When you add a message flow to a broker archive (BAR) file in preparation for deploying it to a broker, you can set additional properties that influence its run time operation. These properties are available for review and update when you select the **Manage and Configure** tab for the broker archive file.



Configurable services properties

The supplied configurable services, and the configurable services that you create, are defined by their names and properties. You can use the supplied services.



DataCaptureStore configurable service

Select the objects and properties that you want to change for the DataCaptureStore configurable service.

Viewing recorded data

You can view a list of recorded messages, or details of individual messages.

Before you begin

Before you start:

Ensure that you have completed all the steps in “Recording data” on page 601.

About this task

You can view recorded data from the web user interface or programmatically. Broker administrators can customize the way in which the data in each data capture store is displayed to web users. For information about how to customize the appearance of the data, see “Customizing the data viewer” on page 135.

Procedure

To configure IBM Integration Bus to view recorded data, complete the following steps.

1. Specify the integration server that is to be used for processing data for viewing, by setting a value for the `egForView` property of your `DataCaptureStore` configurable service.

If you are using the configurable services that you used for recording data, use the `mqsichangeproperties` command to modify the `egForView` property, as shown in the following example:

```
mqsichangeproperties brokerName -c DataCaptureStore -o dataCaptureStoreName  
-n egForView -v integrationServerName
```

- `brokerName` is the name of your broker.
- `dataCaptureStoreName` is the name of your `DataCaptureStore` configurable object.
- `integrationServerName` is the name of the integration server that processes the data for viewing.

If you do not want to use the `DataCaptureStore` and `DataCaptureSource` configurable services that you set up for recording data, create new ones as described in “Recording data” on page 601. Specify a value for the `egForView` property. You can specify the same integration server in the `egForRecord` and the `egForView` properties, or you might want to spread the workload across two integration servers.

For more information about the `egForRecord` and `egForView` properties, see `DataCaptureStore` configurable service.

2. Use one of the following methods to view your data.
 - Use the IBM Integration API. For more information, see “Recording and replaying data with a CMP application” on page 107.
 - View recorded data and administer your brokers from a web interface:
 - a. Configure a web administration server, as described in “Configuring the web user interface server” on page 128.
 - b. Log on to the web user interface (see “Accessing the web user interface” on page 133).
 - c. When you have logged on to the web user interface, the Navigator view is displayed on the left side. In the Navigator view, select the Data node. The data viewer is displayed on the right side of the window.
 - d. Select the data capture store containing the data that you want to view.

- Use the IBM Integration Bus Representational State Transfer (REST) application programming interface. For more information, see Representational State Transfer (REST) API.

Related concepts:

“Record and replay” on page 597

For audit or problem determination purposes, you can record data to a database, then view it, and replay it.

Related tasks:

“Recording data” on page 601

Record data that is flowing through a message flow, or that is emitted by WebSphere Application Server.

“Replaying data”

When you have recorded data, you can replay it to a WebSphere MQ queue.

“Recording and replaying data with a CMP application” on page 107

You can create CMP applications to examine and replay the data that you have recorded and stored by using record and replay.

“Customizing the data viewer” on page 135

Broker administrators can customize the appearance of the data shown in the data capture store in the web user interface.

Related reference:



Configurable services properties

The supplied configurable services, and the configurable services that you create, are defined by their names and properties. You can use the supplied services.



DataCaptureStore configurable service

Select the objects and properties that you want to change for the DataCaptureStore configurable service.

Replaying data

When you have recorded data, you can replay it to a WebSphere MQ queue.

Before you begin

Before you start:

To replay data, you must have configured your system to record data and to view that recorded data. For instructions, see “Recording data” on page 601 and “Viewing recorded data” on page 616.

About this task

You can replay data that you have recorded by using the IBM Integration API or the web user interface.

Replaying data by using the IBM Integration API Procedure

To replay data to a WebSphere MQ queue by using the IBM Integration API, complete the following steps.

1. Use the DataDestination configurable service to define the WebSphere MQ queues to which you can replay data.

Create the configurable service by using the IBM Integration Explorer or the **mqsicreateconfigurableservice** command. You can set one destination for each instance of the DataDestination configurable service. This configurable service is dynamic, therefore you do not need to restart the broker for changes to take effect.

For a description of the properties that you set on this configurable service, see DataDestination configurable service.

2. For an example of how to replay data with the IBM Integration API, see “Recording and replaying data with a CMP application” on page 107.

Replaying data by using a web browser Procedure

To replay data to a WebSphere MQ queue from a web browser, complete the following steps.

1. Use the DataDestination configurable service to define the WebSphere MQ queues to which you can replay data.

Create the configurable service by using the IBM Integration Explorer or the **mqsicreateconfigurableservice** command. You can set one destination for each instance of the DataDestination configurable service. This configurable service is dynamic, therefore you do not need to restart the broker for changes to take effect.

For a description of the properties that you set on this configurable service, see DataDestination configurable service.

2. View your recorded data through the web user interface:
 - a. Configure a web user interface server, as described in “Configuring the web user interface server” on page 128.
 - b. Use the **mqswebuseradmin** command to create your web user accounts. For information on how to do this, see “Managing web user accounts” on page 226.
 - c. Log on to the web user interface (see “Accessing the web user interface” on page 133).
 - d. When you have logged on to the web user interface, the Navigator view is displayed on the left side of the window. In the Navigator view, select the Data node. The data viewer is displayed on the right side of the window.
3. Select the data capture store that contains the data that you want to replay.
Only data stores for which the egForView property is set in the corresponding DataCaptureStore configurable service are available.
4. Each row that is displayed represents a recorded message. To sort these rows into a particular order, click a column heading. You can also use a filter to help find rows in which you are interested.
Filtering is case sensitive and allows the use of wildcards. To search for an exact match, enclose the search string in quotation marks. You can also search for a substring, and you can use an asterisk (*) to match zero or more characters.
5. Optional: You can download data by clicking the arrow in the Data column.
6. Select the messages that you want to replay by clicking the check box next to each required message. You can replay only messages that have data.
7. Click **Mark for replay**. The marked messages are added to the replay list - they are not replayed at this stage.
8. Click the **Replay list** tab.

9. Select the destination for your data.
The list of available destinations corresponds to the destinations that you defined in instances of the DataDestination configurable service.
10. You can now replay all messages or a single message. To replay all messages, click the **Replay all** button. To replay a single message, click the replay icon in the relevant row.

Results

Results:

The selected data is sent to the specified destination. Messages that are not delivered are highlighted.

Related concepts:

“Record and replay” on page 597

For audit or problem determination purposes, you can record data to a database, then view it, and replay it.



Using the IBM Integration Explorer to work with configurable services
Configurable services are used to define properties that are related to external services on which the integration node (broker) relies. Use the IBM Integration Explorer to view, add, modify and delete configurable services.

Related tasks:

“Recording data” on page 601

Record data that is flowing through a message flow, or that is emitted by WebSphere Application Server.

“Viewing recorded data” on page 616

You can view a list of recorded messages, or details of individual messages.



Creating a message flow

Create a message flow to specify how to process messages in the broker. You can create one or more message flows and deploy them to one or more brokers.

Related reference:



mqsicreateconfigurable-service command

Use the **mqsicreateconfigurable-service** command to create an object name for a broker external resource.



DataDestination configurable service

Select the objects and properties that you want to change for the DataDestination configurable service.



MQInput node

Use the MQInput node to receive messages from clients that connect to the broker by using the WebSphere MQ Enterprise Transport, and that use the MQI and AMI application programming interfaces.

Using multiple brokers for record and replay

You can have multiple brokers in your record and replay topology. If you use different brokers for deploying your message flows and for recording data captured from those message flows to a database, then you must configure a publish/subscribe relationship between the brokers.

Before you begin

Before you start:

Read the concept topic “Record and replay” on page 597.

About this task

You can use separate brokers for deploying the message flows from which you want to capture data, for recording the data to a database, and for viewing the data. The broker used for viewing and replaying data does not interact with the broker used to record data to a database. However, if the broker to which you deploy your message flows is not the recording broker, then you must set up a publish/subscribe relationship between these brokers.

Data capture is based on a publish/subscribe model. You configure monitoring on message flows that you have deployed to a broker, for example, to broker *MONBKR*. *MONBKR* publishes to the topic that you specify when you configure your monitoring events. The topic identifies the source of the data that you want to capture. You specify this topic in your *DataCaptureSource* configurable service.

You define a *DataCaptureStore* configurable service where you specify the integration server to use for processing the captured data, *egForRecord*. The broker that you use to create the *DataCaptureStore* configurable service, *RECBKR*, subscribes to the monitoring topic. The subscription messages that this broker receives are processed by *egForRecord* and recorded to the data source specified in the *DataCaptureStore* configurable service.

You must use the same broker to create the *DataCaptureStore* and the *DataCaptureSource* configurable services.

In this scenario, *MONBKR* publishes on the monitoring topic and *RECBKR* subscribes to the topic. If *MONBKR* runs on queue manager *MONQM* and *RECBKR* runs on queue manager *RECRM*, then you need to configure a publish/subscribe relationship between *MONQM* and *RECRM*.

You can choose to create either a cluster or a hierarchical publish/subscribe relationship between the two queue managers. If you plan to add queue managers to your topology frequently, then a cluster relationship is more appropriate. See the topics on “Publish/subscribe topologies” in the WebSphere MQ Version 7 Information Center online. This example uses a hierarchical relationship. In the example, values enclosed in single quotation marks can be replaced with your preferred values, but keep the quotation marks if you use lowercase characters. Complete the following steps:

Procedure

1. Establish a point-to-point channel connection between *MONQM* and *RECRM*.
 - a. On a command line on the server that hosts *MONQM*, enter `runmqsc MONQM`
 - b. Define a transmission queue for *MONQM* to use when forwarding messages to *RECRM*:

```
DEFINE QLOCAL('RECRM') USAGE(XMITQ) TRIGGER TRIGDATA('MONQM.TO.RECRM') INITQ(SYSTEM.CHANNEL.INITQ)
```
 - c. Define a sender channel:

```
DEFINE CHANNEL('MONQM.TO.RECRM') CHLTYPE(SDR) TRPTYPE(TCP) CONNAME('RECRM.SERVER') XMITQ(XMIT)  
DESCR('Sender channel from MONQM to RECRM')
```

RECQM.SERVER is the name of the server that hosts *RECQM*.

- d. Define a receiver channel:

```
DEFINE CHANNEL('RECQM.TO.MONQM') CHLTYPE(RCVR) TRPTYPE(TCP)
DESCR('Receiver channel from RECQM to MONQM')
```

- e. Configure a listener on the queue manager:

```
DEFINE LISTENER ('LISTENER.TCP') TRPTYPE(TCP) PORT(PORT) CONTROL(QMGR)
```

PORT is the port number on which the listener listens.

- f. Start the listener. Enter `START LISTENER('LISTENER.TCP')`

- g. Enter `END` to finish the **RUNMQSC** session.

- h. Now configure *RECQM*. On the server that hosts *RECQM*, enter the following commands at a command line:

```
runmqsc RECQM

DEFINE QLOCAL('MONQM') USAGE(XMITQ) TRIGGER TRIGDATA('RECQM.TO.MONQM') INITQ(SYSTEM.CHANNEL.INITQ)
DEFINE CHANNEL('RECQM.TO.MONQM') CHLTYPE(SDR) TRPTYPE(TCP) CONNAME('MONQM.SERVER') XMITQ(XMIT)
DESCR('Sender channel from RECQM to MONQM')
DEFINE CHANNEL('MONQM.TO.RECQM') CHLTYPE(RCVR) TRPTYPE(TCP)
DESCR('Receiver channel from MONQM to RECQM')
DEFINE LISTENER ('LISTENER.TCP') TRPTYPE(TCP) PORT(PORT) CONTROL(QMGR)
START LISTENER ('LISTENER.TCP')
END
```

MONQM.SERVER is the name of the server that hosts *MONQM*, and *PORT* is the port number on which the listener listens.

You can also configure the queue manager channels by using IBM Integration Explorer. For more information about configuring channels for publish/subscribe, see the topics on “Publish/subscribe topologies” in the WebSphere MQ Version 7.1 Information Center online .

2. Configure the queue manager publish/subscribe hierarchy. Make *RECQM* a child of *MONQM* by completing the following steps:

- a. Open a command line on the server that hosts *RECQM*, and enter: `runmqsc RECQM`

- b. Make *RECQM* a child of *MONQM*:

```
ALTER QMGR PARENT('RECQM')
```

Confirm that the operation worked by listing all publish/subscribe relationships for this queue manager:

```
DIS PUBSUB TYPE(ALL)
```

The output shows that *MONQM* is now the parent of *RECQM*:

```
AMQ8723: Display pub/sub status details.
QMNAME(RECQM) TYPE(LOCAL)
AMQ8723: Display pub/sub status details.
QMNAME(MONQM) TYPE(PARENT)
```

- c. Type `END` to end the **RUNMQSC** session for *RECQM*.

- d. Start a **RUNMQSC** session for *MONQM* and display the publish/subscribe types, as you did for *RECQM*. The output shows that *RECQM* is a child of *MONQM*.

What to do next

Next:

Consider the performance implications of your record and replay topology; see “Tuning data capture” on page 615.

Complete the steps for recording data; see “Recording data” on page 601.

Related tasks:

“Recording, viewing, and replaying data” on page 595

Record data that is processed by a message flow, or emitted from WebSphere Application Server. View and replay the data.

Related reference:



Designing the WebSphere MQ infrastructure

You must create and manage the WebSphere MQ resources that are required to support your brokers, and the applications that connect to them to supply or receive messages.

Using Activity logs

Use Activity logs to get an overview of recent activities in your message flows and associated external resources.

About this task

Activity logs provide immediate, basic information about what is happening in your message flows, and how they are interacting with external resources. As a systems administrator, you can use Activity logs to understand recent activities affecting your systems. The logs are also useful for diagnosing problems that would not be easily resolved by using lower-level trace. For example, they can help you to understand why your message flow is not processing any messages from a remote resource.

Use Activity logs as the first point of investigation when something unexpected happens in a message flow. Since you do not need to turn on activity logging, it can provide an early indication of changed behavior, such as broken connections to external resources.

Find out more about Activity logs by reading the concept topic “Activity log overview” on page 624.

Use the Activity Log view in IBM Integration Explorer to view and work with recent Activity log information for individual message flows and resource types. See “Working with Activity logs in IBM Integration Explorer” on page 625.

To view activities over a longer period, use the Activity log configurable service to write Activity logs to file. See “Writing Activity logs to files” on page 632.

You can also view and configure the runtime properties for Activity logs. See “Viewing and setting runtime properties for Activity logs” on page 630.

Related concepts:

“Resource statistics” on page 530

Resource statistics are collected by a broker to record performance and operating details of resources that are used by integration servers.

Related reference:

IBM Integration Bus logs

IBM Integration Bus writes information to a number of product-specific logs to report the results of actions that you take.

Activity logs

Activity logs contain information about message flows and how they interact with external resources. They can be generated for a message flow or a resource type. Learn about the structure and content of Activity logs, and about the *tags* that enrich the log data and can be used to filter its content.

Activity log overview

Activity logs help users to understand what their message flows are doing by providing a high-level overview of how IBM Integration Bus interacts with external resources. They do not require detailed product knowledge, and can be used to enhance understanding of the overall system.

Activity log messages are concise and avoid technical complexity, although more information is provided in the message detail. Because the log entries are short, uncomplicated, and focused on single activities, they can be quickly scanned and understood. Patterns of behavior and changes to such patterns are easier to identify than in more extensive product trace. The qualitative data about resources provided in Activity logs complements the quantitative resource data provided by resource statistics.

The Activity Log view in IBM Integration Explorer provides a graphical view of recent Activity log events, as well as extensive facilities for organizing and filtering data. Logs can be generated for individual message flows or resource types. You can have multiple Activity Log views for flows and resource types open simultaneously. You can also save a view to a file in Comma Separated Value (CSV) format or copy and paste a selection of the data.

Activity logs can also be written to a circular file system. Use the ActivityLog configurable service to set up file logging if you want continuous logging of activities over a long period.

The configurable service and the IBM Integration Explorer Activity Log view both have extensive filtering capabilities, based on *tags*. *Tags* allow you to search or filter on various attributes, such as resource types or message flow names. All Activity log messages have several tags which are displayed in the log entries and can be used as search and filter keys. Individual resource types also have resource-specific tags that you can use to retrieve the data that you are interested in.

Activity logs can be retrieved programmatically by using the IBM Integration API (CMP).

Because Activity logs are enabled by default, they can be useful for troubleshooting. For example, if the Activity log for a message flow shows a prolonged period of inactivity, review the Activity logs for associated resource types. You might discover that a problem with an external resource is the reason why messages are no longer being processed by your message flow.

Related concepts:

“Resource statistics” on page 530

Resource statistics are collected by a broker to record performance and operating details of resources that are used by integration servers.

Configurable services

Configurable services are typically runtime properties. You can use them to define properties that are related to external services on which the broker relies; for example, an SMTP server or a JMS provider.

Related tasks:

“Using Activity logs” on page 623

Use Activity logs to get an overview of recent activities in your message flows and associated external resources.

“Navigating brokers and broker resources in a CMP application” on page 80

Explore the status and attributes of the broker that your CMP application is connected to, and discover information about its resources.

Related reference:

IBM Integration API

Use the IBM Integration API (CMP) Java classes and methods to develop CMP applications.

Configurable services properties

The supplied configurable services, and the configurable services that you create, are defined by their names and properties. You can use the supplied services.

Working with Activity logs in IBM Integration Explorer

You can view and work with Activity log information by using the IBM Integration Explorer.

Before you begin

Read the concept topic “Activity log overview” on page 624.

About this task

IBM Integration Explorer provides a graphical user interface for viewing and working with Activity logs. You can open multiple logs for message flows and resource types simultaneously. For each individual view, you can save or copy the log information. You can also customize the views by using the search, filter, ordering, and column selection features.

Read the following topics to understand more about what you can do with Activity logs in IBM Integration Explorer:

- “Viewing Activity logs for message flows and resource types” on page 626
- “Saving Activity log information” on page 627
- “Copying Activity log information” on page 628
- “Working with Activity log content” on page 629

For information about how to configure properties that affect the general behavior of Activity logging within a single integration server, such as whether logging is enabled, see “Viewing and setting runtime properties for Activity logs” on page 630.

For a description and explanation of the columns displayed in the Activity Log view, see Activity logs.

Related tasks:

“Using Activity logs” on page 623

Use Activity logs to get an overview of recent activities in your message flows and associated external resources.

Related reference:

 IBM Integration Explorer views

The IBM Integration Explorer can be used to manage and administer your integration nodes (brokers) and deployed resources.

 Activity logs

Activity logs contain information about message flows and how they interact with external resources. They can be generated for a message flow or a resource type. Learn about the structure and content of Activity logs, and about the *tags* that enrich the log data and can be used to filter its content.

Viewing Activity logs for message flows and resource types

You can view Activity log information for a resource type or a message flow by using the IBM Integration Explorer.

Before you begin

Read the concept topic “Activity log overview” on page 624.

About this task

You can open an Activity Log view for one or more resource types or message flows; there is a separate pane for each log opened. You can have Activity Log views for message flows and resource types open at the same time.

The names of the views follow the format *integrationNode\integrationServer\resource* - Activity Log where:

- *integrationNode* is the name of the integration node (broker).
- *integrationServer* is the name of the integration server.
- *resource* is the name of the message flow or resource type (for example, JMS).

Each row within a log represents an activity. For more information about what is displayed for each activity, see Activity logs.

You can view Activity log information for a message flow only if the message flow has been deployed to an integration node that is running and connected.

Procedure

To open the Activity Log view for a message flow or resource type, complete the following steps:

1. In the **WebSphere MQ Explorer - Navigator** view, expand the Integration Nodes folder and then the folder for the integration server that you want to work with.
2. Choose one of the following options to view an Activity log for either a message flow or resource type:
 - To open the Activity log for a message flow, right-click the message flow and select **Open Activity Log**.
 - To open the Activity log for a resource type:
 - a. Expand the **Resource Managers** folder.

- b. Right-click a resource type and select **Open Activity Log**.

What to do next

To find out more about what you can do with Activity log content, see “Working with Activity log content” on page 629.

Related concepts:

“Activity log overview” on page 624

Activity logs help users to understand what their message flows are doing by providing a high-level overview of how IBM Integration Bus interacts with external resources. They do not require detailed product knowledge, and can be used to enhance understanding of the overall system.

Related tasks:

“Working with Activity logs in IBM Integration Explorer” on page 625

You can view and work with Activity log information by using the IBM Integration Explorer.

Saving Activity log information

Save the Activity log information that is written to the Activity Log view in the IBM Integration Explorer.

Before you begin

Open the Activity Log view that you want to save in the IBM Integration Explorer. See “Viewing Activity logs for message flows and resource types” on page 626.

About this task

Activity log information is deleted automatically from the integration node (broker) when the integration node restarts; and log entries are lost when the in-memory log cache is overwritten. You can save the log contents to file if you want to retain them.

Procedure

1. Right-click in the Activity Log view and select **Save Activity Log**.
2. Navigate to a directory into which to save the log information and enter a file name. The file is saved as a comma-delimited .csv file.
3. Click **Save**.

Results

Each message displayed in the Activity Log view is written to a comma-delimited .csv file. Only displayed activities and fields are written; hidden fields or activities that were removed from the view by using a filter search string are not saved. For more information about how you can organize the data in the Activity Log view, see “Working with Activity log content” on page 629.

You can open the saved log in a text editor or in a spreadsheet application such as Microsoft Excel.

Related concepts:

“Activity log overview” on page 624

Activity logs help users to understand what their message flows are doing by providing a high-level overview of how IBM Integration Bus interacts with external resources. They do not require detailed product knowledge, and can be

used to enhance understanding of the overall system.

Related tasks:

“Viewing Activity logs for message flows and resource types” on page 626
You can view Activity log information for a resource type or a message flow by using the IBM Integration Explorer.

“Copying Activity log information”

Copy a selection from the Activity log information that is written to the Activity Log view in the IBM Integration Explorer.

“Working with Activity log content” on page 629

Manipulate the information that is displayed in the Activity Log view in the IBM Integration Explorer. Display the information that you are interested in, by using the refresh, search, sort, and filter features available in this view.

Copying Activity log information

Copy a selection from the Activity log information that is written to the Activity Log view in the IBM Integration Explorer.

Before you begin

Open the Activity Log view for the log that you want to copy in the IBM Integration Explorer. See “Viewing Activity logs for message flows and resource types” on page 626.

About this task

Procedure

1. Use the Shift or Ctrl keys on your keyboard to select part of the Activity log. If you do not make a selection, all of the visible data is copied.
2. Right-click in the Activity Log view and select **Copy Selection**.
3. Paste the data into an appropriate editor.

Results

The data is written in comma -delimited format. It can be imported into a spreadsheet application such as Microsoft Excel.

Related concepts:

“Activity log overview” on page 624

Activity logs help users to understand what their message flows are doing by providing a high-level overview of how IBM Integration Bus interacts with external resources. They do not require detailed product knowledge, and can be used to enhance understanding of the overall system.

Related tasks:

“Viewing Activity logs for message flows and resource types” on page 626
You can view Activity log information for a resource type or a message flow by using the IBM Integration Explorer.

“Saving Activity log information” on page 627

Save the Activity log information that is written to the Activity Log view in the IBM Integration Explorer.

“Working with Activity log content” on page 629

Manipulate the information that is displayed in the Activity Log view in the IBM Integration Explorer. Display the information that you are interested in, by using the refresh, search, sort, and filter features available in this view.

Working with Activity log content

Manipulate the information that is displayed in the Activity Log view in the IBM Integration Explorer. Display the information that you are interested in, by using the refresh, search, sort, and filter features available in this view.

Before you begin

Open the Activity Log view that you want to work with. See “Viewing Activity logs for message flows and resource types” on page 626.

About this task

Use the data mining features of the Activity Log view to identify and organize the Activity log information that you are interested in. You can then save the customized information to a file or copy it into an editor of your choice.

Procedure

Use the toolbar in the Activity Log view to customize the information that is displayed.

- To replace the current log entries with the most recent activities, click the **Refresh** icon. Display options that were in force at the time of the refresh are maintained. For example, if a filter was applied before the refresh using the **Apply filter** button, this filter will still be active after the refresh operation completes.
- Search the log entries for a specified string.
 1. Optional: Select an individual column from the **Filter** menu. The default behavior if a column is not selected is to search on **All Columns**.
 2. Enter a string into the text box next to the **Filter** menu. The search is case-sensitive. You can also substitute the wildcard "." for any single character and the wildcard "*" for any sequence of zero or more characters.
 3. Click **Apply filter** or press **Enter**. The filter applies only to the specified subset of visible Activity log entries and columns. Activities that match the search string in one or more of the selected and displayed fields are shown. Activities that do not contain a matching string disappear from view.
- Click **Clear** to clear the search and filter criteria. This button is activated only if a search string has been entered. The Activity log entries which had previously been removed by filtering are restored.
- To view the log entries for a single thread, perform one of the following actions:
 - Select a thread identifier from the thread menu on the toolbar.
 - Right-click any row in the log with that thread identifier in its *ThreadID* field and select **Focus on this thread**. If a thread has already been selected using the thread menu on the toolbar, right-clicking a row no longer provides the menu option **Focus on this thread**. Instead, you get the menu option **Show all threads**.

The default behavior is to display **All Threads**.

- Add columns to, or remove columns from, the view:
 1. Click **Select columns** to display the Activity Log Columns window. This window lists all of the available Activity log fields. Fields that are currently displayed are selected.
 2. Select or clear the check boxes to determine which columns are displayed in the view.

When you open a view for the first time, only the *Severity Level*, *Message Number*, *Timestamp*, *Resource Type (RM)*, *Message Flow (MSGFLOW)*, and *Message Summary* columns are displayed by default. If you change the displayed columns, your preferences will be preserved next time you open an Activity Log view.

If you filter on an Activity log, your search string is only matched against the data in columns that have been selected for display.

- For more detailed message information, double-click any row in the Activity log pane.
- To order on a column, click the column title bar. The data under the column is sorted into ascending order. If you click a second time, the original order is restored. You can order on these columns: *Thread ID*, *Timestamp*, and all *Tag* fields. For more information about *Tags*, see Activity logs.
- Click **Previous** or **Next** to display more log entries. These buttons are disabled if there are no further log entries to display.

Related tasks:

“Saving Activity log information” on page 627

Save the Activity log information that is written to the Activity Log view in the IBM Integration Explorer.

“Copying Activity log information” on page 628

Copy a selection from the Activity log information that is written to the Activity Log view in the IBM Integration Explorer.

Related reference:



Activity logs

Activity logs contain information about message flows and how they interact with external resources. They can be generated for a message flow or a resource type. Learn about the structure and content of Activity logs, and about the *tags* that enrich the log data and can be used to filter its content.

Viewing and setting runtime properties for Activity logs

Display and modify the runtime properties that govern the behavior of Activity logs within the scope of a single integration server.

About this task

Use the **mqsireportproperties** command to display the current configuration parameters for activity logging within the scope of a single integration server. Use the **mqsichangeproperties** command to modify these configuration parameters, in order, for example, to change the number of Activity log entries held in memory, or to limit log entries to error messages.

The properties that can be set are:

- activityLogEnabled

This property takes a Boolean value that indicates whether Activity log entries are kept in memory. The default value for this property is true.

- defaultLogSizePerThread

This property takes an integer value that determines the maximum number of Activity log entries written to memory before the log is recycled. The default value is 1000. The property can have any positive integer value.

If the thread on which the log is processed terminates (for example, because you redeploy or restart the integration server), the log entries that are held in

memory are lost. When diagnosing a problem, save the current activity log before redeploying or restarting your integration server.

- **minSeverityLevel**

This property takes an integer value that is used to specify the severity of messages written to the Activity log.

Only messages with a severity level higher than or equal to the specified value for this property are logged. The initial value of the property is INFO, which means that messages of all severities are written to the log.

Valid values are INFO, WARN, and ERROR, where ERROR is the highest severity level and INFO the lowest.

Specify a value of WARN to log messages of severity levels WARN and ERROR.

To display the current configuration properties and then modify them, complete these steps:

Procedure

1. Ensure that the broker is running. If it is not, use the **mqsistart** command to start it.
2. Display the current Activity log configuration properties. Enter the following command:

```
mqsireportproperties myBroker -e myIntegrationServer -o  
ComIbmActivityLogManager -a
```

where:

- *myBroker* is the name of your broker
- *myIntegrationServer* identifies the integration server by name
- *ComIbmActivityLogManager* is the name of the Activity log resource manager

The **-a** option indicates that all top-level property values are displayed. An example of the output produced follows:

```
ComIbmActivityLogManager  
uuid='ComIbmActivityLogManager'  
userTraceLevel='none'  
traceLevel='none'  
userTraceFilter='none'  
traceFilter='none'  
activityLogEnabled='true'  
defaultLogSizePerThread='1000'  
minSeverityLevel='INFO'
```

The response confirms that activity logging is enabled. There are 1,000 log entries held in memory for each active thread in an integration server. The **minSeverityLevel** value of INFO means that Activity log messages of all severities are displayed in the log.

3. Change the Activity log configuration parameters. Enter this command:

```
mqsichangeproperties myBroker -e myIntegrationServer -o  
ComIbmActivityLogManager -n PropertyNames -v PropertyValue
```

where:

- *myBroker* is the name of your broker
- *myIntegrationServer* identifies the integration server by name
- *ComIbmActivityLogManager* is the Activity log resource manager
- *PropertyNames* is a comma-delimited list of the properties to add or change; for example, defaultLogSizePerThread,minSeverityLevel

- *PropertyValues* is a comma-delimited list of the proposed values for the properties to add or change; for example, 500,ERROR

Related reference:



mqsichangeproperties command

Use the **mqsichangeproperties** command to modify broker properties and properties of broker resources.



mqsireportproperties command

Use the **mqsireportproperties** command to display properties that relate to a broker, an integration server, or a configurable service.



Activity logs

Activity logs contain information about message flows and how they interact with external resources. They can be generated for a message flow or a resource type. Learn about the structure and content of Activity logs, and about the *tags* that enrich the log data and can be used to filter its content.

Writing Activity logs to files

You can define a configurable service of type ActivityLog to write your Activity logs to a file system. Configure the service to capture the Activity log data that you are interested in.

About this task

Use the filtering capabilities of this feature to customize the log content and make it relevant to your needs. You can configure the size and number of log files and the criteria by which the log rotates from one file to the next. You can also identify the location and file name of the log files.

You can use the ActivityLog configurable service to configure the following Activity log file characteristics:

- Whether file logging is enabled
- The maximum number of files used for each log
- The maximum size of the log files (before the log rotates to the next file)
- The maximum age of the log files (before the log rotates to the next file)
- Filters that determine the content of the logs
- The severity levels of messages logged
- The log path and file name
- Whether messages are formatted with inserts included in the message text
- The scope of the configurable service (which integration server it applies to)

For more information about the properties that can be set and their possible values, see Activity log configurable service. The scope of the settings is the integration server. For example, if you set the **numberOfLogs** property to 5, each integration server has a maximum of five files for its Activity log.

The files are in UTF-8 format.

Procedure

To define and customize the configurable service, perform the following steps:

1. Define a configurable service of type ActivityLog. You can define the configurable service by using IBM Integration Explorer. See Using the IBM Integration Explorer to work with configurable services. Alternatively, you can use the **mqsicreateconfigurable-service** command.
2. Optional: Use the **mqsireportproperties** and the **mqsichangeproperties** commands to view and modify the Activity log file properties.

Related tasks:

“Using Activity logs” on page 623

Use Activity logs to get an overview of recent activities in your message flows and associated external resources.

Related reference:

 Activity logs

Activity logs contain information about message flows and how they interact with external resources. They can be generated for a message flow or a resource type. Learn about the structure and content of Activity logs, and about the *tags* that enrich the log data and can be used to filter its content.

 **mqsicreateconfigurable-service** command

Use the **mqsicreateconfigurable-service** command to create an object name for a broker external resource.

 **mqsireportproperties** command

Use the **mqsireportproperties** command to display properties that relate to a broker, an integration server, or a configurable service.

 **mqsichangeproperties** command

Use the **mqsichangeproperties** command to modify broker properties and properties of broker resources.

 Configurable services properties

The supplied configurable services, and the configurable services that you create, are defined by their names and properties. You can use the supplied services.

 Activity log configurable service

Select the objects and properties that you want to change for the Activity log configurable service.

Workload management

Allows system administrators to monitor and adjust the speed that messages are processed, as well as controlling the actions taken on unresponsive flows and threads.

The following topics explain the various options available under workload management:

Message flow notification

A common requirement is to be able to monitor the speed at which IBM Integration Bus processes messages. Workload management allows the system administrator to express a notification threshold for individual message flows deployed. An out of range notification message is produced if the notification threshold is exceeded. A back in range notification message is produced if the notification threshold later drops back into

range. For more information, see “Configure a message flow to cause notification threshold messages to be published.”

Setting the maximum rate for a message flow

The system administrator can set the maximum rate that an individual message flow can run at. The maximum rate is specified as the total number of input messages processed every second. When set, the number of input messages that are processed across the flow is measured. This measure is irrespective of the number of additional instances in use, the number of input nodes in the message flow, or the number of errors that occur. If necessary, a processing delay is introduced to keep the input message processing rate under the maximum flow rate setting. For more information, see “Setting the maximum rate for a message flow” on page 637.

Unresponsive message flows

Allows you to specify and monitor the maximum amount of time that any message flow is allowed to process a message for, and to specify an action to be taken if the timeout is exceeded. Additionally, manual requests can be made to stop a message flow by restarting the integration server. For more information, see “Unresponsive message flows” on page 639.

Configure a policy

The system administrator can specify a workload management policy within Integration Registry for a message flow. The workload management policy has the advantage in that it encompasses all of the properties available under workload management in one place and therefore allows for easier tuning of message flow performance. For more information, see Integration Registry.

Configure a message flow to cause notification threshold messages to be published

How to configure and monitor the message flow notification threshold.

About this task

Details how a notification threshold can be set up on a message flow to cause a notification message to be sent if the message count for messages arriving in the flow exceeds the notification threshold. A further notification message is sent if later the message count drops back below the notification threshold.

Note: Unless otherwise stated, the notification threshold is a measure of the total messages every second.

The following sections provide further information about how you can configure the message flow notification threshold and monitor message flow performance:

- “Setting the notification threshold for a message flow”
- “Message publication when the message rate for a message flow is out of range” on page 636
- “Message publication when the message rate for a message flow goes back into range” on page 636

Setting the notification threshold for a message flow

Describes how to calculate and set the message flow notification threshold.

About this task

Before activating the message flow notification threshold, the system administrator must first establish what message rate is required for the message flow in question. The message rate is a measure of the total number of messages arriving each second at the message flow from all input nodes of any type. It is the total rate and not the average rate. For example: A message flow contains two input nodes that are called A and B. If input node A received messages at the rate of 50 messages a second, and input node B received messages at the rate of 70 messages a second, then the total message rate for the message flow would be 120 messages per second.

There are two ways the notification threshold can be set for a message flow:

- Directly within a BAR file.
- As one of the attributes within a workload management policy that is defined within Integration Registry.

BAR file

The notification threshold is set within the BAR file under a property called *notificationThresholdMsgsPerSec*.

The property can be set in the following ways:

- Within the BAR file through the IBM Integration Toolkit editor.
- Within the BAR file through the IBM Integration Explorer.
- Within the BAR file through the **mqsiapplybaroverride** command line.

Additionally, once the BAR file is deployed, the property can be set dynamically within the flow through the IBM Integration API. Any change to the property is picked up immediately and does not require the flow to be restarted.

Workload management policy

Details of how to create and configure a workload management policy are described in “Configure a workload management policy within Integration Registry” on page 644.

A notification threshold value of zero, or not set, will cause the message flow notification threshold to be turned off. The default state is off.

Within the same integration server, a mixture of message flows can run along side each other, some with the notification threshold set, others with the notification threshold turned off.

Related concepts:



IBM Integration Explorer

The IBM Integration Explorer is a graphical user interface based on the Eclipse platform for administering your integration nodes (brokers).

Related reference:



IBM Integration Toolkit

The IBM Integration Toolkit provides your application development environment on Windows and Linux on x86.



mqsiapplybaroverride command

Use the **mqsiapplybaroverride** command to replace configurable values in the broker archive (BAR) deployment descriptor with new values that you specify in a

properties file.



Broker properties that are accessible from ESQL, the Mapping node, and Java. You can access broker, message flow, and node properties from ESQL, the Mapping node, and Java.

Message publication when the message rate for a message flow is out of range

Lists the conditions that control the publishing of the message rate is out of range message.

About this task

The message rate statistics are collected at a checkpoint that occurs every 20 seconds. The total message rate is calculated at this checkpoint, and if the total message rate exceeds the notification threshold, the out of range XML message is published. If the total message rate continues to stay above the notification threshold, then no further out of range messages are published.

If you enable the notification threshold you can subscribe to the following topic:

```
$/SYS/Broker/<brokerName>/WorkloadManagement/AboveThreshold/<integrationServerLabel>/<applicationName>/<libraryName>/<messageFlowLabel>
```

where *brokerName* is the name of the broker, *integrationServerLabel* is the name of the integration server on that broker, *applicationName* is the name of the application on that integration server, *libraryName* is the name of the library on that application, and *messageFlowLabel* is the name of the message flow that is deployed to the library.

In the situation where the message flow is not contained in either an application or a library, the *applicationName* or *libraryName* parameters must be omitted along with their enclosing forward slash (/). For example:

- If the message flow is not contained in an application and a library:

```
$/SYS/Broker/<brokerName>/WorkloadManagement/AboveThreshold/<integrationServerLabel>/<messageFlowLabel>
```
- If the message flow is contained in an application and not in a library:

```
$/SYS/Broker/<brokerName>/WorkloadManagement/AboveThreshold/<integrationServerLabel>/<applicationName>/<messageFlowLabel>
```

Related tasks:

“Message publication when the message rate for a message flow goes back into range”

Lists the conditions that control the publishing of the message rate is back in range message.

Message publication when the message rate for a message flow goes back into range

Lists the conditions that control the publishing of the message rate is back in range message.

About this task

The message rate statistics are collected at a checkpoint that occurs every 20 seconds and the total message rate is calculated at this checkpoint. If the total

message rate previously exceeded the notification threshold, and then later the total message rate drops back into range, the back in range XML message is published.

No state is stored when the message flow is stopped, restarted, or redeployed.

When a flow is terminated, the flow termination process checks to see whether the last message published was to report that the message rate exceeded the notification threshold. In this situation, the flow termination process automatically publishes a message to report that the message flow is now back in range.

If you enable the notification threshold you can subscribe to the following topic:

```
$$SYS/Broker/<brokerName>/WorkloadManagement/BelowThreshold/<integrationServerLabel>/<applicationName>/<libraryName>/<messageFlowLabel>
```

where *brokerName* is the name of the broker, *integrationServerLabel* is the name of the integration server on that broker, *applicationName* is the name of the application on that integration server, *libraryName* is the name of the library on that application, and *messageFlowLabel* is the name of the message flow that is deployed to the library.

In the situation where the message flow is not contained in either an application or a library, the *applicationName* or *libraryName* parameters must be omitted along with their enclosing forward slash (/). For example:

- If the message flow is not contained in an application and a library:

```
$$SYS/Broker/<brokerName>/WorkloadManagement/BelowThreshold/<integrationServerLabel>/<messageFlowLabel>
```

where *brokerName* is the name of the broker, *integrationServerLabel* is the name of the integration server on that broker, and *messageFlowLabel* is the name of the message flow that is deployed to the integration server.

- If the message flow is contained in an application and not in a library:

```
$$SYS/Broker/<brokerName>/WorkloadManagement/BelowThreshold/<integrationServerLabel>/<applicationName>/<messageFlowLabel>
```

where *brokerName* is the name of the broker, *integrationServerLabel* is the name of the integration server on that broker, *applicationName* is the name of the application on that integration server, and *messageFlowLabel* is the name of the message flow that is deployed to the application.

Related tasks:

“Message publication when the message rate for a message flow is out of range” on page 636

Lists the conditions that control the publishing of the message rate is out of range message.

Setting the maximum rate for a message flow

Allows system administrators to restrict the maximum rate at which a message flow can run at by setting the maximum rate property.

About this task

The system administrator is able to restrict the rate that an individual message flow can run at by setting the maximum rate property. The maximum rate is specified as the total number of input messages processed every second. The maximum rate value is divided equally among all threads that are running in the message flow irrespective of the number of input nodes within the flow.

To calculate the number of threads within a specific message flow:

1. Count the number of input nodes within the flow.
2. Add on the number of additional instances that are specified for each input node.

The following three examples assume the maximum rate is set to 50, and helps to highlight some of the variations that can occur from basic to more complex scenarios:

- A message flow has one input node with no additional instances:
 - Only a single thread in operation.
 - The full maximum rate allocation of 50 messages every second is applied to that one thread.
- A message flow has one input node with one additional instance:
 - Two threads in operation. One for the node and one for the additional instance.
 - The maximum rate is split equally across the two threads. Each thread has a maximum rate allocation of 25 messages every second.
- A message flow has three input nodes with four additional instances that are specified on the first node and three additional instances that are specified on the second node:
 - 10 threads in operation. Three input node threads, four additional instance threads for the first node, and three additional instance threads for the second node.
 - The maximum rate is split equally across the 10 threads. Each thread has a maximum rate allocation of five messages every second.

If any individual thread exceeds their maximum rate allocation, a processing delay is introduced on that thread to keep the processing rate under the assigned maximum rate allocation.

There are two ways the maximum rate can be set for a message flow:

- Directly within a BAR file.
- As one of the attributes within a workload management policy that is defined within Integration Registry.

BAR file

The maximum rate is set within the BAR file under a property called *maximumRateMsgsPerSec*.

The property can be set in the following ways:

- Within the BAR file through the IBM Integration Toolkit editor.

Note: You must refresh the content of a migrated BAR file before you can see and configure the *maximumRateMsgsPerSec* property. For more information, see Refreshing the contents of a broker archive.

- Within the BAR file through the IBM Integration Explorer.
- Within the BAR file through the **mqsipplybaroverride** command line.

Note: For example, to set the *maximumRateMsgsPerSec* property for a message flow included in an application, you can use the following sample code:

```
mqsipplybaroverride -b BARfile -k applicationName -m sampleFlow#maximumRateMsgsPerSec=100
```

For more information, see **mqsiapplybaroverride** command.

Additionally, once the BAR file is deployed, the property can be set dynamically within the flow through the IBM Integration API. Any change to the property is picked up immediately and does not require the flow to be restarted.

Workload management policy

Details of how to create and configure a workload management policy are described in “Configure a workload management policy within Integration Registry” on page 644.

A maximum rate value of zero, or not set, causes the message flow maximum rate to be turned off. The default state is off.

Within the same integration server, a mixture of message flows can run along side each other, some with the maximum rate set, others with the maximum rate turned off.

Related concepts:

 IBM Integration Explorer

The IBM Integration Explorer is a graphical user interface based on the Eclipse platform for administering your integration nodes (brokers).

Related tasks:

 Refreshing the contents of a broker archive

Refresh the contents of a broker archive (BAR) file by rebuilding it in the Broker Archive editor. Alternatively, remove resources from your broker archive and, having made the required changes, add them back again.


Related reference:

 IBM Integration Toolkit

The IBM Integration Toolkit provides your application development environment on Windows and Linux on x86.

 **mqsiapplybaroverride** command

Use the **mqsiapplybaroverride** command to replace configurable values in the broker archive (BAR) deployment descriptor with new values that you specify in a properties file.

 Broker properties that are accessible from ESQL, the Mapping node, and Java
You can access broker, message flow, and node properties from ESQL, the Mapping node, and Java.

Unresponsive message flows

Allows system administrators to monitor a message flow to see if it has been requested to stop and to take the required action.

Under certain conditions message flow processing can become unresponsive, for instance when:

- Waiting for a response from an external system.
- Processing an infinite loop or a calculation that takes a very long time.
- Deadlocked between two resources.

In previous releases of WebSphere Message Broker the only way to resolve this situation is by terminating the integration server process. As part of IBM

Integration Bus Version 9.0 new functions are introduced to allow termination of unresponsive message flows in a controlled manner.

The following topics explain the functions available:

Programmatically check from within a message flow if it has been requested to stop

By using the programming APIs (ESQL, Java, and .NET), you are able to see if a message flow has been requested to stop. For more information, see “Check if a message flow has been requested to stop.”

Manually force a message flow to stop

A force option *-f* has been added to the `mqsistopmsgflow` command to allow you to specify how the message flow can be forced to stop. For more information, see “Manually force a message flow to stop” on page 641.

Automatically force a message flow to stop

Two message flow properties are provided which can be set within the BAR file and the workload management policy:

- *processingTimeoutSec* – maximum time a message flow can process a message before taking an action.
- *processingTimeoutAction* – the action to take. Currently, this is restricted to *none* or *restartExecutionGroup*.

Additionally, an event message is published on a WebSphere MQ topic once the *processingTimeoutSec* is exceeded and again when the message flow processing has finished.

For more information, see “Automatically force a message flow to stop” on page 642.

Check if a message flow has been requested to stop

By using programming APIs, check to see if a request has been made to stop processing within a message flow.

About this task

The three programming APIs (ESQL, Java, and .NET) have new functions added to them that allows you to check from within a message flow if the flow has been requested to stop. The function in all three APIs returns a Boolean value of true if a request has been made to stop the message flow.

The functions are described as follows:

ESQL

```
INSTANDESTOPPING();
```

Returns true if a request has been made to stop the message flow.

For more information, see INSTANDESTOPPING function.

Java

```
Static Boolean MbMessageFlow.isInstanceStopping();
```

Returns true if a request has been made to stop the message flow.

For more information, see **MbMessageFlow** Class under Java user-defined extensions API.

.NET

```
Static Boolean NbMessageFlow.InstanceStopping();
```

Returns true if a request has been made to stop the message flow.

For more information, see **NbMessageFlow** Class under .NET reference.

Related reference:



INSTANDESTOPPING function

The INSTANDESTOPPING checks to see if a request has been made to stop a message flow, and returns a Boolean value of true if a request has been made.



.NET reference

You can call .NET assemblies from .NETCompute nodes, and from nodes that support ESQL.

Related information:



Java user-defined extensions API

Manually force a message flow to stop

Using the **mqsistopmsgflow** command to force a message flow to stop.

About this task

The **mqsistopmsgflow** command has been enhanced with a force option to escalate the mechanisms that are used to stop a message flow.

Without the force option, the **mqsistopmsgflow** command sends a request to a message flow to stop it by first waiting for all threads that are used by the message flow to finish. If one of the threads is stuck in an operation, then the message flow stop will never complete.

The force option *-f* has been added to allow system administrators to specify how the message flow should be stopped. Currently, *restartExecutionGroup* is the only valid option available, that causes the message flow to be flagged as stopped and then to restart the integration server. When the integration server restarts the message flow will be in the stop state.

You can combine using the **mqsistopmsgflow** command without, and then with, the force option to escalate the mechanisms that are used to stop the message flow. For example:

```
mqsistopmsgflow IIB9BROKER -e eg1 -m mf1 -w 30  
mqsistopmsgflow IIB9BROKER -e eg1 -m mf1 -w 30 -f restartExecutionGroup
```

The first command would first try to stop the message flow normally and wait 30 seconds for a response. The second command would cause the integration server to restart, but only if the flow was not already stopped.

The non-force version of the **mqsistopmsgflow** command has the potential to hang the entire integration server deployment mechanism, but the force version of the **mqsistopmsgflow** command will not suffer from the same issue and avoids taking any locks that might cause hanging or deadlocks. The force version of the **mqsistopmsgflow** command will still work even if the standard deployment mechanisms are hung.

The same force option is available in the IBM Integration API and REST API, and within the web user interface.

Related tasks:

“Developing applications that use the IBM Integration API” on page 58
Develop Java applications that use the IBM Integration API (also known as the CMP) to communicate with, deploy to, and manage brokers and their associated resources.



Representational State Transfer (REST) API

IBM Integration Bus supports the REST management API, and you can use HTTP clients to administer broker resources.

“Administering brokers using the web user interface” on page 127
You can use the IBM Integration Bus web user interface to administer broker resources.

Related reference:



mqsisistopmsgflow command

Use the **mqsisistopmsgflow** command to stop integration servers, applications, and message flows.

Automatically force a message flow to stop

Monitors message flow processing time and automatically takes a specified action if the timeout is exceeded.

About this task

Two message flow properties are provided to specify the maximum amount of time that any message flow can be allowed to process a message and an action to be taken if the timeout is exceeded:

- *processingTimeoutSec* – maximum time a message flow can process a message before taking a specified action. The time is measured in seconds and is taken from the point a message is received on an input node.
- *processingTimeoutAction* – the action to take. Currently, this action is restricted to *none* or *restartExecutionGroup*.

There are two ways that both of these properties can be set for a message flow:

- Directly within a BAR file.
- As one of the attributes within a workload management policy that is defined within Integration Registry.

BAR file

The two properties can be set in the following ways:

- Within the BAR file through the IBM Integration Toolkit editor.
- Within the BAR file through the IBM Integration Explorer.
- Within the BAR file through the **mqsiapplybaroverride** command line.

Workload management policy

Details of how to create and configure a workload management policy are described in “Configure a workload management policy within Integration Registry” on page 644.

Once the *processingTimeoutSec* timeout period is exceeded an event message is published. For more information, see “Message flow timeout exceeded event message” on page 643. Thereafter, if the *processingTimeoutAction* option has been set

to *none* and processing of the message flow continues to completion, another event message is published. For more information, see “Message flow processing finished event message” on page 644. However, if the *processingTimeoutAction* option has been set to *restartExecutionGroup* the integration server is restarted and no further event messages are published from the message flow.

Related concepts:

IBM Integration Explorer

The IBM Integration Explorer is a graphical user interface based on the Eclipse platform for administering your integration nodes (brokers).

Related reference:

IBM Integration Toolkit

The IBM Integration Toolkit provides your application development environment on Windows and Linux on x86.

`mqsipplybaroverride` command

Use the `mqsipplybaroverride` command to replace configurable values in the broker archive (BAR) deployment descriptor with new values that you specify in a properties file.

Message flow timeout exceeded event message:

Details the event message that is published when the *processingTimeoutSec* timeout period is exceeded.

About this task

Once the *processingTimeoutSec* timeout period is exceeded for the message flow, a timeout exceeded XML message is published.

You can view the message by subscribing to the following topic:

```
$$SYS/Broker/<brokerName>/WorkloadManagement/ProcessingTimeout/<integrationServerLabel>/<applicationName>/<libraryName>/<messageFlowLabel>
```

where *brokerName* is the name of the broker, *integrationServerLabel* is the name of the integration server on that broker, *applicationName* is the name of the application on that integration server, *libraryName* is the name of the library on that application, and *messageFlowLabel* is the name of the message flow that is deployed to the library.

In the situation where the message flow is not contained in either an application or a library, the *applicationName* or *libraryName* parameters must be omitted along with their enclosing forward slash (/). For example:

- If the message flow is not contained in an application and a library:

```
$$SYS/Broker/<brokerName>/WorkloadManagement/ProcessingTimeout/<integrationServerLabel>/<messageFlowLabel>
```
- If the message flow is contained in an application and not in a library:

```
$$SYS/Broker/<brokerName>/WorkloadManagement/ProcessingTimeout/<integrationServerLabel>/<applicationName>/<messageFlowLabel>
```

Related tasks:

“Message flow processing finished event message” on page 644

Details the event message that is published if a timeout action of *none* is specified and when the message flow processing finishes.

Message flow processing finished event message:

Details the event message that is published if a timeout action of *none* is specified and when the message flow processing finishes.

About this task

If the message flow exceeds the *processingTimeoutSec* timeout period, the action that is defined by the property *processingTimeoutAction* is taken. If the action defined is *none*, processing of the message flow is allowed to continue and a processing finished XML message is published once all processing has completed.

You can view the message by subscribing to the following topic:

```
$/SYS/Broker/<brokerName>/WorkloadManagement/ProcessingFinished/<integrationServerLabel>/<applicationName>/<libraryName>/<messageFlowLabel>
```

where *brokerName* is the name of the broker, *integrationServerLabel* is the name of the integration server on that broker, *applicationName* is the name of the application on that integration server, *libraryName* is the name of the library on that application, and *messageFlowLabel* is the name of the message flow that is deployed to the library.

In the situation where the message flow is not contained in either an application or a library, the *applicationName* or *libraryName* parameters must be omitted along with their enclosing forward slash (/). For example:

- If the message flow is not contained in an application and a library:

```
$/SYS/Broker/<brokerName>/WorkloadManagement/ProcessingFinished/<integrationServerLabel>/<messageFlowLabel>
```

- If the message flow is contained in an application and not in a library:

```
$/SYS/Broker/<brokerName>/WorkloadManagement/ProcessingFinished/<integrationServerLabel>/<applicationName>/<messageFlowLabel>
```

Related tasks:

“Message flow timeout exceeded event message” on page 643

Details the event message that is published when the *processingTimeoutSec* timeout period is exceeded.

Configure a workload management policy within Integration Registry

Describes how to create and configure a policy, and defines the precedent rules.

About this task

Instead of defining properties on the message flow you can create policies within Integration Registry, so that message flows can refer to them to find properties at run time. If you use this method, you can change the values of attributes for a policy on the broker, which then affects the behavior of a message flow without the need for redeployment.

The workload management policy has the advantage in that it encompasses all of the properties available under workload management in one place, including the notification threshold and maximum rate, and therefore allows for easier tuning of message flow performance.

A policy can be set up and administered within Integration Registry in the following ways:

- By the IBM Integration Web Administration Interface.
- By the IBM Integration API.
- By the various command line options.

For more information about how the policy can be set up and administered within Integration Registry, see Integration Registry.

When a policy is created and deployed, it adheres to the following precedence rules:

- Properties that are set in the policy take precedence over properties that are set as BAR file properties.
- If a particular property is not set in the policy, the equivalent property that is set on the BAR file takes effect.
- If no policy is associated with a BAR file, the BAR file properties take effect.
- If the policy is removed from a BAR file, the BAR file properties take effect.

Related tasks:



Configuring workload management by using the web user interface
Create, edit, and delete workload management policies by using the web user interface.



Configuring workload management policies by using the IBM Integration API
The IBM Integration API provides methods for creating, retrieving, updating and deleting workload management policies.



Configuring workload management policies by using a command console
Command-line tools to create and administer workload management policies.

Chapter 4. Troubleshooting and support

If you are having problems with your message flow applications, use the techniques described in this section to help you to diagnose and solve the problems.

About this task

This section contains information about the various techniques that you can use to diagnose problems with IBM Integration Bus.

Procedure

- “Making initial checks” on page 650
- “Dealing with problems” on page 666
- “Using logs” on page 839
- “Using trace” on page 845
- “Using dumps and abend files” on page 871
- “Recovering after failure” on page 887
- “Contacting your IBM Support Center” on page 876

What to do next

You can also read the general troubleshooting guidance in the following topics:

- “Troubleshooting overview”
- “Searching knowledge bases” on page 882
- “Getting product fixes” on page 883
- “Contacting IBM Software Support” on page 885
- “IBM Support Assistant Data Collector” on page 879

If an IBM Integration Bus component or command has returned an error, and you want further information about a message written to the screen or the log, you can browse for details of the message in “Diagnostic messages” on page 891; the lists of messages grouped in numeric order.

For information that is specific to debugging message flows, see Testing and debugging message flows.

Related reference:



Troubleshooting

Use the reference information in this section to help you diagnose errors in IBM Integration Bus.

Troubleshooting overview

Troubleshooting is the process of finding and eliminating the cause of a problem. Whenever you have a problem with your IBM software, the troubleshooting process begins as soon as you ask yourself “what happened?”

A basic troubleshooting strategy at a high level involves:

1. "Recording the symptoms of the problem"
2. "Re-creating the problem"
3. "Eliminating possible causes"

Recording the symptoms of the problem

Depending on the type of problem that you have, whether it be with your application, your server, or your tools, you might receive a message that indicates that something is wrong. Always record the error message that you see. As simple as this sounds, error messages sometimes contain codes that might make more sense as you investigate your problem further. You might also receive multiple error messages that look similar but have subtle differences. By recording the details of each one, you can learn more about where your problem exists.

Sources of error messages:

- Problems view
- Local error log
- Eclipse log
- Activity log
- User trace
- Service trace
- Error dialog boxes
- `mqsixplain` command

Re-creating the problem

Think back to what steps you were doing that led to the problem. Try those steps again to see if you can easily re-create the problem. If you have a consistently repeatable test case, it is easier to determine what solutions are necessary.

- How did you first notice the problem?
- Did you do anything different that made you notice the problem?
- Is the process that is causing the problem a new procedure, or has it worked successfully before?
- If this process worked before, what has changed? (The change can refer to any type of change that is made to the system, ranging from adding new hardware or software, to reconfiguring existing software.)
- What was the first symptom of the problem that you witnessed? Were there other symptoms occurring around the same time?
- Does the same problem occur elsewhere? Is only one machine experiencing the problem or are multiple machines experiencing the same problem?
- What messages are being generated that could indicate what the problem is?

You can find more information about these types of question in "Making initial checks" on page 650.

Eliminating possible causes

Narrow the scope of your problem by eliminating components that are not causing the problem. By using a process of elimination, you can simplify your problem and avoid wasting time in areas that are not responsible. Consult the information in this product and other available resources to help you with your elimination process.

- Has anyone else experienced this problem? See: “Searching knowledge bases” on page 882.
- Is there a fix you can download? See: “Getting product fixes” on page 883.

Related tasks:

Chapter 4, “Troubleshooting and support,” on page 647

If you are having problems with your message flow applications, use the techniques described in this section to help you to diagnose and solve the problems.

“Making initial checks” on page 650

Before you start problem determination in detail, consider whether there is an obvious cause of the problem, or an area of investigation that is likely to give useful results. This approach to diagnosis can often save a lot of work by highlighting a simple error, or by narrowing down the range of possibilities.

“Dealing with problems” on page 666

Learn how to resolve some of the typical problems that can occur.

“Using logs” on page 839

There are a variety of logs that you can use to help with problem determination and troubleshooting.

“Using trace” on page 845

You can use different types of trace to help you with problem determination and troubleshooting.

“Using Activity logs” on page 623

Use Activity logs to get an overview of recent activities in your message flows and associated external resources.

“Using dumps and abend files” on page 871

A dump or an abend file might be produced when a problem occurs. Dumps and abend files can be used by IBM to resolve the problem.

“Searching knowledge bases” on page 882

If you have a problem with your IBM software, you want it resolved quickly. Begin by searching the available knowledge bases to determine whether the resolution to your problem is already documented.

“Getting product fixes” on page 883

A product fix might be available to resolve your problem. You can determine what fixes are available from the IBM support site.

“Contacting IBM Software Support” on page 885

IBM Software Support provides assistance with product defects.

“Recovering after failure” on page 887

Follow a set of procedures to recover after a serious problem.

Related reference:

 **mqsixplain** command

Use the **mqsixplain** command to display the contents of an IBM Integration Bus BIP message.

 Troubleshooting

Use the reference information in this section to help you diagnose errors in IBM Integration Bus.

Making initial checks

Before you start problem determination in detail, consider whether there is an obvious cause of the problem, or an area of investigation that is likely to give useful results. This approach to diagnosis can often save a lot of work by highlighting a simple error, or by narrowing down the range of possibilities.

About this task

This section contains a list of questions to consider. As you go through the list, make a note of anything that might be relevant to the problem. Even if your observations do not suggest a cause straight away, they might be useful later if you have to carry out a systematic problem determination exercise.

Procedure

- “Has IBM Integration Bus run successfully before?”
- “Did you log off Windows while IBM Integration Bus components were active?” on page 651
- “Are the Linux and UNIX environment variables set correctly?” on page 652
- “Are there any error messages or return codes that explain the problem?” on page 653
- “Can you reproduce the problem?” on page 654
- “Has the message flow run successfully before?” on page 656
- “Have you made any changes since the last successful run?” on page 657
- “Is there a problem with descriptive text for a command?” on page 658
- “Is there a problem with a database?” on page 659
- “Is there a problem with the network?” on page 660
- “Does the problem affect all users?” on page 661
- “Have you recently changed a password?” on page 661
- “Have you applied any service updates?” on page 662
- “Do you have a component that is running slowly?” on page 663
- “Additional checks for z/OS users” on page 664

Related tasks:

Chapter 4, “Troubleshooting and support,” on page 647

If you are having problems with your message flow applications, use the techniques described in this section to help you to diagnose and solve the problems.

Related reference:



Troubleshooting

Use the reference information in this section to help you diagnose errors in IBM Integration Bus.

Has IBM Integration Bus run successfully before?

If you have successfully run an IBM Integration Bus component, determine whether your problem is caused by incorrect installation and setup, or a different cause. If it has *not* run successfully before, work through the information to determine what is preventing successful operation.

About this task

Complete the following steps:

Procedure

1. Check your setup

IBM Integration Bus might not be set up correctly.

Linux **UNIX** On Linux and UNIX operating systems, check that you have set up the command environment correctly; see [Setting up a command environment](#) for more information.

2. Verify the installation

Check [Verifying your IBM Integration Bus installation](#) for information about how you can verify a basic configuration on your system. This topic describes how to verify your installation on Linux on x86, Linux on x86-64, or Windows by using either the IBM Integration Toolkit or the IBM Integration Explorer. On Linux on x86 and Windows, you can use the Default Configuration wizard and the Samples Preparation wizard to help you with basic configuration and verification.

For more detailed configuration information, refer to [Configuring brokers for test and production environments](#).

Related tasks:



Installing complementary products

IBM Integration Bus works with several other products to provide complementary services.



Configuring brokers for test and production environments

Create one or more brokers on one or more computers, and configure them on your test and production systems to process messages that contain your business data.



Creating a default configuration

Use the Default Configuration wizard to create and test a basic broker configuration.

Chapter 4, “Troubleshooting and support,” on page 647

If you are having problems with your message flow applications, use the techniques described in this section to help you to diagnose and solve the problems.

“Making initial checks” on page 650

Before you start problem determination in detail, consider whether there is an obvious cause of the problem, or an area of investigation that is likely to give useful results. This approach to diagnosis can often save a lot of work by highlighting a simple error, or by narrowing down the range of possibilities.

Did you log off Windows while IBM Integration Bus components were active?

How to avoid problems caused by logging off while components are active.

About this task

Windows On Windows, logging off when an IBM Integration Bus component (a broker) is active can cause a problem.

You might see messages, including BIP2070, BIP2642, BIP1102, and BIP1103, in the Windows Event log.

When you log off, any queue manager that supports the IBM Integration Bus component is stopped unless it is defined to run as a Windows service. The component runs Windows services and remains active, but it finds that the queue manager and queue manager objects that are associated with it are no longer available.

Procedure

To avoid this problem, set the queue manager that is used by the broker to run as a Windows service, so that logging off Windows from does not cause this problem.

Related tasks:

Chapter 4, “Troubleshooting and support,” on page 647

If you are having problems with your message flow applications, use the techniques described in this section to help you to diagnose and solve the problems.

“Making initial checks” on page 650

Before you start problem determination in detail, consider whether there is an obvious cause of the problem, or an area of investigation that is likely to give useful results. This approach to diagnosis can often save a lot of work by highlighting a simple error, or by narrowing down the range of possibilities.

“Windows: Viewing the local error log” on page 840

The Windows Event Viewer is where IBM Integration Bus writes records to the local system. Use Windows system facilities to view this log.

Are the Linux and UNIX environment variables set correctly?

Use the `mqsiprofile` command to set a command environment.

About this task

UNIX On Linux and UNIX systems, the basic settings are made by the `mqsiprofile` command, which is located in the following directory:

`install_dir/bin`

What to do next

See Setting up a command environment for information about setting up the command environment; see Creating a broker on Linux and UNIX systems for instructions about creating a broker on your operating system.

Related tasks:

Chapter 4, “Troubleshooting and support,” on page 647

If you are having problems with your message flow applications, use the techniques described in this section to help you to diagnose and solve the problems.

“Making initial checks” on page 650

Before you start problem determination in detail, consider whether there is an obvious cause of the problem, or an area of investigation that is likely to give useful results. This approach to diagnosis can often save a lot of work by highlighting a simple error, or by narrowing down the range of possibilities.



Installing complementary products

IBM Integration Bus works with several other products to provide complementary services.



Setting up a command environment

After you have installed the product on one of the distributed systems, you must initialize the environment before you can use a runtime component or command.

Related reference:



Environment variables after installation

On distributed systems, ensure that your environment is set up correctly.

Are there any error messages or return codes that explain the problem?

You can find details of error messages and return codes in several places.

Procedure

• BIP messages and reason codes

IBM Integration Bus error messages have the prefix BIP. If you receive any messages with this prefix (for example, in the UNIX, Linux, or z/OS syslog), you can search for them in the information center for an explanation. You can also view the full content of a BIP message by using the **mqsixplain** command. For more information, see **mqsixplain** command.

Windows

In the Windows Event log, references to BIP messages are identified by the source name "WebSphere Broker v6000", where v6000 can be replaced by a number representing the exact service level of your installation, for example 6001.

IBM Integration Bus messages that have a mixture of identifiers such as BIPmsgs, BIPv700, BIPv610, BIPv500, WMQIv210, MQSIV202, and MQSIV201 indicate a mixed installation, which does not work properly.

• Other messages

For messages with a different prefix, such as AMQ or CSQ for WebSphere MQ, or SQL for DB2, see the appropriate messages and codes documentation for a suggested course of action to help resolve the problem.

Messages that are associated with the startup of IBM Integration Bus, or were issued while the system was running before the error occurred, might indicate a system problem that prevented your application from running successfully.

A large or complex IBM Integration Bus broker environment might require some additional configuration of the environment beyond what is recommended in *Installing complementary products*. The need for such configuration changes is typically indicated by warning or error messages that are logged by the various components, including WebSphere MQ, the databases, and the operating system. These messages are normally accompanied by a suggested user response.

Related tasks:

Chapter 4, "Troubleshooting and support," on page 647

If you are having problems with your message flow applications, use the techniques described in this section to help you to diagnose and solve the problems.

“Making initial checks” on page 650

Before you start problem determination in detail, consider whether there is an obvious cause of the problem, or an area of investigation that is likely to give useful results. This approach to diagnosis can often save a lot of work by highlighting a simple error, or by narrowing down the range of possibilities.

“Windows: Viewing the local error log” on page 840

The Windows Event Viewer is where IBM Integration Bus writes records to the local system. Use Windows system facilities to view this log.

Related reference:

“Diagnostic messages” on page 891

Diagnostic messages are listed in this section in numeric order, grouped according to the component to which they relate.

Can you see all of your files and folders?

How to show all files in Windows Explorer:

About this task

If you are using Windows Explorer to view your files and you cannot see all of your files and folders, such as the broker workpath directory, this is because Windows Explorer, by default, hides some files and folders.

Procedure

1. Click **Tools > Folder options**. The Folder Options dialog box opens.
2. Click the **View** tab and select **Show hidden files and folders**.

Related tasks:

Chapter 4, “Troubleshooting and support,” on page 647

If you are having problems with your message flow applications, use the techniques described in this section to help you to diagnose and solve the problems.

“Making initial checks” on page 650

Before you start problem determination in detail, consider whether there is an obvious cause of the problem, or an area of investigation that is likely to give useful results. This approach to diagnosis can often save a lot of work by highlighting a simple error, or by narrowing down the range of possibilities.

“Changing the location of the work path” on page 194

The work path directory is the location where a component stores internal data, such as installation logs, component details, and trace output. The shared-classes directory is also located in the work path directory and is used for deployed Java code. If the work path directory does not have enough capacity, redirect the directory to another file system that has enough capacity.

Can you reproduce the problem?

If you can reproduce the problem, consider the conditions under which you can do so.

Procedure

- **Is the problem caused by a particular message flow?** If so, use the debugging facility of the IBM Integration Toolkit and user tracing to identify the problem.
- **Is the problem caused by a command?** On distributed operating systems, you issue commands at the system command line.

z/OS On z/OS, you can issue commands from the console, the syslog, or by submitting a batch job. You enter customization commands from an OMVS session. Console commands that you enter from the console or syslog might be converted to uppercase, depending on the system configuration. This conversion can cause some commands, such as **mqsichangetrace**, to fail, especially if these commands contain parameters that must be lowercase. An error message indicating that the integration server is not available might be caused by the integration server name being in the wrong case. The same thing can happen on message flows.

- **Does a problem command work if it is entered by another user ID?**

If the command works when it is entered by another user ID, check the environment of each user. Paths, especially shared library paths, might be different. On Windows, UNIX systems, and Linux verify that all users have set up their command environment correctly; refer to sample profile for more information.

Windows On Windows, the environment for the broker is determined by the system settings, not by a particular user's variables. However, the user's variables affect non-broker commands.

UNIX On LinuxUNIX systems, only the service ID that is specified when the broker was created can start a broker.

Windows On Windows, any authorized user can start a broker.

Related concepts:



Flow debugger overview

Use the flow debugger in the IBM Integration Toolkit to track messages through your message flows.



Trace

If you cannot get enough information about a particular problem from the entries that are available in the various logs, the next troubleshooting method to consider is using trace. Trace provides more details about what is happening while code runs. The information produced from trace is sent to a specified trace record, so that you or IBM support personnel can analyze it to discover the cause of your problem.

Related tasks:

Chapter 4, “Troubleshooting and support,” on page 647

If you are having problems with your message flow applications, use the techniques described in this section to help you to diagnose and solve the problems.

“Making initial checks” on page 650

Before you start problem determination in detail, consider whether there is an obvious cause of the problem, or an area of investigation that is likely to give useful results. This approach to diagnosis can often save a lot of work by highlighting a simple error, or by narrowing down the range of possibilities.

“Are the Linux and UNIX environment variables set correctly?” on page 652

Use the **mqsiprofile** command to set a command environment.

“Using trace” on page 845

You can use different types of trace to help you with problem determination and troubleshooting.

“Resolving problems when developing message flows” on page 708

Use the advice given here to help you to resolve common problems that can arise when developing message flows.

Related reference:



Commands

All IBM Integration Toolkit and runtime commands that are provided on distributed systems are listed, grouped by function, with references to command details.

Has the message flow run successfully before?

Sometimes a problem appears in a message flow that has previously run successfully.

About this task

To identify the cause of the problem, answer the following questions:

Procedure

- **Have you made any changes to the message flow since it last ran successfully?**

If so, it is likely that the error exists somewhere in the new or modified part of the flow. Examine the changes and see if you can find an obvious reason for the problem.

- **Have you used all the functions of the message flow before?**

Did the problem occur when you used part of the message flow that had never been invoked before? If so, it is likely that the error exists in that part. Try to find out what the message flow was doing when it failed by using user tracing, trace nodes, and the IBM Integration Toolkit's debugger function.

If you have run a message flow successfully on many previous occasions, check the current queue status and the files that were being processed when the error occurred. It is possible that they contain some unusual data value that invokes a rarely-used path in the message flow.

- **Does the message flow check all return codes?**

Has your system been changed, perhaps in a minor way, but your message flow does not check the return codes it receives as a result of the change? For example:

- Does your message flow assume that the queues that it accesses can be shared? If a queue has been redefined as exclusive, can your message flow deal with return codes indicating that it can no longer access that queue?
- Have any security profiles been changed? A message flow could fail because of a security violation.

- **Does the message flow expect particular message formats?**

If a message with an unexpected message format has been put onto a queue (for example, a message from a queue manager on a different operating system) it might require data conversion or a different form of processing. Also, check whether you have changed any of the message formats that are used.

- **Does the message flow run on other IBM Integration Bus systems?**

Is there something different about the way that your system is set up that is causing the problem? For example, have the queues been defined with the same maximum message length or priority? Are there differences in the databases used, or their setup?

- **Are you using any user-defined extensions?**

There might be translation or compilation problems with loadable implementation library (LIL) files. Before you look at the code, examine the

output from the translator, the compiler or assembler, and the linkage editor, to see if any errors have been reported. Fix any errors to make the user-defined extension work.

If the documentation shows that each of these steps was completed without error, consider the coding logic of the message flow, message set, or user-defined extension. Do the symptoms of the problem indicate which function is failing and, therefore, which piece of code is in error? See [User-defined extensions overview](#) for more information.

- **Can you see errors from IBM Integration Bus or external resources, such as databases?**

Your message flow might be losing errors because of incorrect use of the failure terminals on built-in nodes. If you use the failure terminals, make sure that you handle errors adequately. See [Handling errors in message flows](#) for more information about failure terminals.

Related concepts:



Trace

If you cannot get enough information about a particular problem from the entries that are available in the various logs, the next troubleshooting method to consider is using trace. Trace provides more details about what is happening while code runs. The information produced from trace is sent to a specified trace record, so that you or IBM support personnel can analyze it to discover the cause of your problem.

Related tasks:

Chapter 4, “[Troubleshooting and support](#),” on page 647

If you are having problems with your message flow applications, use the techniques described in this section to help you to diagnose and solve the problems.

“[Making initial checks](#)” on page 650

Before you start problem determination in detail, consider whether there is an obvious cause of the problem, or an area of investigation that is likely to give useful results. This approach to diagnosis can often save a lot of work by highlighting a simple error, or by narrowing down the range of possibilities.

“[Resolving problems when developing message flows](#)” on page 708

Use the advice given here to help you to resolve common problems that can arise when developing message flows.



Handling errors in message flows

The broker provides basic error handling for all your message flows. If basic processing is not sufficient, and you want to take specific action in response to certain error conditions and situations, you can enhance your message flows to provide your own error handling.

Related reference:



User trace

User trace is one of two types of optional trace that are available in IBM Integration Bus and provides more information than that provided by the entries that are written to the Administration log. User trace is inactive by default; you must activate it explicitly by using a command, or by selecting options in the IBM Integration Toolkit.

Have you made any changes since the last successful run?

Have you changed IBM Integration Bus, any related software, or any hardware?

About this task

When you are considering changes that might have been made recently, think not only about IBM Integration Bus, but also about the other programs with which it interfaces, the hardware, device drivers, and any new applications.

Procedure

- **Have you changed your initialization procedure?**

On z/OS, have you changed any data sets or library definitions? Has the operating system been initialized with different parameters? Check for error messages generated during initialization.

- **Has the profile of the user who is running the commands on Linux or UNIX systems been changed?**

If so, this might mean that the user no longer has access to the required objects and commands.

- **Has any of the software on your system been upgraded to a later release?**

Check that the upgrade completed successfully, and whether the new software is compatible with IBM Integration Bus (check the product readme.html file).

- **Do your message flows deal with the errors and return codes that they might get as a result of any changes that you have made?**

Check that your message flow can handle all possible error situations.

Related tasks:

Chapter 4, "Troubleshooting and support," on page 647

If you are having problems with your message flow applications, use the techniques described in this section to help you to diagnose and solve the problems.

"Making initial checks" on page 650

Before you start problem determination in detail, consider whether there is an obvious cause of the problem, or an area of investigation that is likely to give useful results. This approach to diagnosis can often save a lot of work by highlighting a simple error, or by narrowing down the range of possibilities.



Handling errors in message flows

The broker provides basic error handling for all your message flows. If basic processing is not sufficient, and you want to take specific action in response to certain error conditions and situations, you can enhance your message flows to provide your own error handling.

Is there a problem with descriptive text for a command?

On Linux and UNIX systems, be careful when including special characters (backslash (\) and double quotation mark (") characters) in descriptive text for some commands.

About this task

If you use either of these characters in descriptive text, precede them with the escape character backslash (\) ; that is, enter \\ or \" if you want \ or " in your text.

Related tasks:

Chapter 4, "Troubleshooting and support," on page 647

If you are having problems with your message flow applications, use the techniques described in this section to help you to diagnose and solve the

problems.

“Making initial checks” on page 650

Before you start problem determination in detail, consider whether there is an obvious cause of the problem, or an area of investigation that is likely to give useful results. This approach to diagnosis can often save a lot of work by highlighting a simple error, or by narrowing down the range of possibilities.

Related reference:



Commands

All IBM Integration Toolkit and runtime commands that are provided on distributed systems are listed, grouped by function, with references to command details.



Characters allowed in commands

You must adhere to a few rules when you provide names or identifiers for the components and resources in your broker environment.

Is there a problem with a database?

If you have database problems, complete a set of initial checks to identify errors.

Procedure

- Check that the database is started.
- Check that you have correctly completed ODBC configuration:
 - **Linux** **UNIX** On Linux and UNIX systems, check that you have created a copy of the sample ODBC configuration file (`odbc.ini`), and have modified them for your environment, and that you have not added any extra unsupported parameters.
 - **Windows** On Windows systems, click **Start > Control Panel > Administrative Tools > Data Sources (ODBC)** to configure the connections you require.
Detailed instructions for setting up ODBC connections on distributed systems are provided in Enabling ODBC connections to the databases.
- Use the `mqsicvp` command to help you with ODBC database diagnostics.
- Check that you have correctly completed JDBC configuration. Detailed instructions for setting up JDBC connections are provided in Enabling JDBC connections to the databases.
- Check the number of database connections that are in use on DB2 for AIX. If you use local mode connections, a maximum of 10 is supported.
- If messages that indicate that `imbfdb2v6.lil` failed to load, check that you have installed a supported database. Details of database managers and versions are given in Supported databases.

Related tasks:

Chapter 4, “Troubleshooting and support,” on page 647

If you are having problems with your message flow applications, use the techniques described in this section to help you to diagnose and solve the problems.

“Making initial checks” on page 650

Before you start problem determination in detail, consider whether there is an obvious cause of the problem, or an area of investigation that is likely to give useful results. This approach to diagnosis can often save a lot of work by highlighting a simple error, or by narrowing down the range of possibilities.

“ODBC trace” on page 864

You can use various methods to trace for ODBC activity, depending on the operating system that you are using.

“Resolving problems when using databases” on page 809

Use the advice given here to help you to resolve problems that can arise when using databases.

Capturing database state

If an error occurs when the broker accesses an external database, you can either let the broker throw an exception during node processing or use ESQL statements to process the exception within the node itself.

Related reference:

Supported databases

You can optionally configure databases to contain data that is accessed by your message flows. Databases from IBM and other suppliers are supported at specific versions on supported operating systems.

`mqsicvp` command

Use the `mqsicvp` command to perform verification tests on a broker, or to verify ODBC connections.

Is there a problem with the network?

IBM Integration Bus uses WebSphere MQ for inter-component communication. If components are on separate queue managers, they are connected by message channels.

About this task

There can be communication problems between any of these:

Procedure

- Brokers
- The IBM Integration Explorer
- The IBM Integration Toolkit

What to do next

If any two components are on different queue managers, make sure that the channels between them are working. Use the WebSphere MQ `display chstatus` command to see if messages are flowing.

Use the `ping` command to check that the remote computers are connected to the network, or if you suspect that the problem might be with the network itself. For example, use the command `ping brokername`, where *brokername* is a computer name. If you get a reply, the computer is connected. If you don't get a reply, ask your network administrator to investigate the problem. Further evidence of network problems might be messages building up on the transmission queues.

Related tasks:

Chapter 4, “Troubleshooting and support,” on page 647

If you are having problems with your message flow applications, use the techniques described in this section to help you to diagnose and solve the problems.

“Making initial checks” on page 650

Before you start problem determination in detail, consider whether there is an obvious cause of the problem, or an area of investigation that is likely to give useful results. This approach to diagnosis can often save a lot of work by highlighting a simple error, or by narrowing down the range of possibilities.

Does the problem affect all users?

On distributed systems, problems can be caused by different users having different environments.

About this task

Check whether there is a different user ID in an incoming message, or a different user ID issuing a command (or acting as the broker's service ID).

Related tasks:

Chapter 4, “Troubleshooting and support,” on page 647

If you are having problems with your message flow applications, use the techniques described in this section to help you to diagnose and solve the problems.

“Making initial checks” on page 650

Before you start problem determination in detail, consider whether there is an obvious cause of the problem, or an area of investigation that is likely to give useful results. This approach to diagnosis can often save a lot of work by highlighting a simple error, or by narrowing down the range of possibilities.



Securing WebSphere MQ resources

Secure the WebSphere MQ resources that your broker configuration requires.

Have you recently changed a password?

Check that you have updated passwords correctly.

About this task

If you have changed the operating system password for one of the following user IDs, the broker or the deployed message flows might have access problems:

- The ID that you have specified for the broker's **serviceUserId** on Windows
- The ID that you have specified for access to a database

If these problems occur, complete one or more of the following steps:

Procedure

- Change the properties of the broker to reflect the password change. Use the **mqsichangebroker** command to change the appropriate parameters.
- Run the **mqsisetdbparms** command to redefine the correct values for the user ID and password.
- Run the **mqsireportdbparms** command to see the user ID that is defined. You can also run **mqsireportdbparms** using the **-n** and **-p** parameters and a possible password. If the entered password is correct, then the command returns information, including if the password was changed since the broker was last started.
- Run **mqsireportdbparms** and check what user IDs are defined for the systems that require one. It might be that a security identity was not defined for the ID.

- Ensure that your passwords do not contain reserved keywords. For example, "IBM" is a reserved keyword in DB2.

Related tasks:

Chapter 4, "Troubleshooting and support," on page 647

If you are having problems with your message flow applications, use the techniques described in this section to help you to diagnose and solve the problems.

"Making initial checks" on page 650

Before you start problem determination in detail, consider whether there is an obvious cause of the problem, or an area of investigation that is likely to give useful results. This approach to diagnosis can often save a lot of work by highlighting a simple error, or by narrowing down the range of possibilities.

 **Securing WebSphere MQ resources**

Secure the WebSphere MQ resources that your broker configuration requires.

"Checking the security credentials that are used by an integration node" on page 392

To check what security credentials are set on an integration node that is connected to a remote system or database, use the `mqsireportdbparms` command.

Related reference:

 **mqsichangebroker** command

Use the `mqsichangebroker` command to change one or more of the configuration parameters of the broker.

 **mqsisetdbparms** command

Use the `mqsisetdbparms` command to associate a specific user ID and password (or SSH identity file) with one or more resources that are accessed by the broker.

 **mqsireportdbparms** command

Use the `mqsireportdbparms` command to list all parameters that are set for a specific broker.

Have you applied any service updates?

Check what service updates you have applied to your software.

About this task

If you have applied an APAR or a PTF to WebSphere MQ for z/OS, or a fix pack or interim fix has been applied to a distributed operating system, check that no error message was produced. If the installation was successful, check with the IBM Support Center for any known error with the service update.

Procedure

- If a service update has been applied to any other product, consider the effect that it might have on IBM Integration Bus.
- Ensure that you have followed any instructions in the service update that affect your system. For example, you might need to redefine a resource, or stop and restart a component.
- If you are not sure whether a service update has been applied to your system, search for and view the release notes stored in the product installation directory. These notes include the service level and details of the maintenance that you have applied.

Related tasks:



Finding the latest information

Access the latest information for IBM Integration Bus.



Applying service to the Integration Bus component

Apply maintenance updates and program fixes to the Integration Bus component.



Applying service to the IBM Integration Toolkit

Apply maintenance updates and program fixes to the IBM Integration Toolkit.

Chapter 4, “Troubleshooting and support,” on page 647

If you are having problems with your message flow applications, use the techniques described in this section to help you to diagnose and solve the problems.

“Making initial checks” on page 650

Before you start problem determination in detail, consider whether there is an obvious cause of the problem, or an area of investigation that is likely to give useful results. This approach to diagnosis can often save a lot of work by highlighting a simple error, or by narrowing down the range of possibilities.

Do you have a component that is running slowly?

If a particular component, or the system in general, is running slowly, you can take some actions to improve the performance.

About this task

Consider the following actions:

Procedure

- Check whether tracing is on. You might have started IBM Integration Bus user tracing or service tracing, ODBC tracing, WebSphere MQ tracing, or native database tracing. If one or more of these traces are active, turn them off.
- Clear out all old abend files from your errors directory. If you do not clear the directory of unwanted files, you might find that your system performance degrades because significant space is used up.
- On Windows, use the workpath `-w` parameter of the `mqsicreatebroker` command to create the errors directory in a hard disk partition that does not contain IBM Integration Bus or Windows.
- Increase your system memory.

Related concepts:



Logs

If an error is reported by an IBM Integration Bus component, start your investigations into its causes by looking at the product and systems logs to which information is written during component operation.

Related tasks:

Chapter 4, “Troubleshooting and support,” on page 647

If you are having problems with your message flow applications, use the techniques described in this section to help you to diagnose and solve the problems.

“Making initial checks” on page 650

Before you start problem determination in detail, consider whether there is an obvious cause of the problem, or an area of investigation that is likely to give useful results. This approach to diagnosis can often save a lot of work by highlighting a simple error, or by narrowing down the range of possibilities.

“Using trace” on page 845

You can use different types of trace to help you with problem determination and troubleshooting.

“Resolving problems with performance” on page 823

Use the advice given here to help you to resolve common problems with performance.

Related reference:

 **mqsicreatebroker** command

Use the **mqsicreatebroker** command to create a broker and its associated resources.

Additional checks for z/OS users

Problem determination for z/OS users.

About this task

On IBM Integration Bus for z/OS, you might find that:

- The console has unexpected messages; look for an explanation by searching for the message number in the information center, or referring to other relevant messages and codes documentation.
- RRS is not started; issue the command **D A,RRS**.
- The queue manager is not available; issue the WebSphere MQ command **DISPLAY THREAD(*)**.
- DB2 is not working; issue the DB2 command **DISPLAY THREAD**.

The cause of your problem on z/OS could be in any of the following areas. Check each of these, starting with whichever seems the most likely, based on the nature of your problem.

Procedure

- The queue manager address space
- A queue manager in your queue-sharing group
- The channel initiator address space
- Batch or TSO address space
- The z/OS system (including ARM, RRS, or the Coupling Facility)
- The network (including APPC or TCP/IP)
- Another system, for example a queue manager on another operating system or a WebSphere MQ client
- The external security manager product, for example RACF or ACF2
- DB2

What to do next

Understanding interactions between the runtime components

There are two runtime components: the broker and the integration server. These components communicate with each other by exchanging command requests inside WebSphere MQ messages to perform actions, such as deploying a message flow.

After completion of the command request, responses are sent back to the originating component, indicating whether the request was successful.

On IBM Integration Bus for z/OS, you see extra information messages issued by z/OS runtime components that allow you and IBM Service personnel to determine the interactions between the various runtime components, including the Configuration Manager, broker and integration server.

When the broker receives command requests the broker issues a message that identifies the command request. When this request completes, a message is issued that indicates whether the command is successful. Each command request that the broker receives results in at least one matching command request being sent to an integration server, and a corresponding message being issued. Every response from an integration server that results from these command requests result in a message being issued. These messages can be turned on and off and are to be used by the IBM Service team.

When an integration server receives command requests from the broker, the integration server issues information messages that identify the command request. For actions that are contained within a command request, an information message is issued that identifies the action to be performed and the resource upon which the action is to take effect. When the request completes, the integration server issues a message that indicates that the message has been processed.

Loading IBM and user-defined nodes and parsers

When an integration server starts, it loads all available IBM and user-defined nodes and parsers. For each library that is loaded, two messages are issued. One is issued before the library is loaded and one is issued after the library has been loaded.

Related tasks:

Chapter 4, “Troubleshooting and support,” on page 647

If you are having problems with your message flow applications, use the techniques described in this section to help you to diagnose and solve the problems.

“Making initial checks” on page 650

Before you start problem determination in detail, consider whether there is an obvious cause of the problem, or an area of investigation that is likely to give useful results. This approach to diagnosis can often save a lot of work by highlighting a simple error, or by narrowing down the range of possibilities.



Using WebSphere MQ shared queues for input and output (z/OS)

On z/OS systems, you can define WebSphere MQ shared queues as input and output queues for message flows. You might need to serialize access to those messages.

Related reference:



Other sources of diagnostic information on z/OS

Other files that you might find useful for problem determination.

Dealing with problems

Learn how to resolve some of the typical problems that can occur.

Before you begin

Before you start:

Make initial checks, see “Making initial checks” on page 650, and check the following locations to see if you can rectify the problem:

- Logs, see “Using logs” on page 839
- Trace, see “Using trace” on page 845
- Dumps, see “Using dumps and abend files” on page 871

About this task

This section contains the following topics:

Procedure

- “Resolving problems when installing” on page 667
- “Resolving problems when uninstalling” on page 675
- “Resolving problems that occur when migrating or importing resources” on page 676
- “Resolving problems when running commands” on page 679
- “Resolving problems when running samples” on page 681
- “Resolving problems when creating resources” on page 685
- “Resolving problems that occur when you start resources” on page 688
- “Resolving problems when stopping resources” on page 705
- “Resolving problems when deleting resources” on page 707
- “Resolving problems when developing message flows” on page 708
- “Resolving problems when deploying message flows or message sets” on page 752
- “Resolving problems that occur when debugging message flows” on page 768
- “Resolving problems when developing message models” on page 775
- “Resolving problems when using messages” on page 782
- “Resolving problems when you are writing business rules” on page 796
- “Resolving problems when you use the IBM Integration Toolkit” on page 797
- “Resolving problems when using the IBM Integration Explorer” on page 806
- “Resolving problems when using databases” on page 809
- “Resolving problems when using publish/subscribe” on page 819
- “Resolving problems with performance” on page 823
- “Resolving problems when developing CMP applications” on page 828
- “Resolving problems with user-defined extensions” on page 829

Related tasks:

Chapter 4, “Troubleshooting and support,” on page 647

If you are having problems with your message flow applications, use the techniques described in this section to help you to diagnose and solve the problems.

“Making initial checks” on page 650

Before you start problem determination in detail, consider whether there is an obvious cause of the problem, or an area of investigation that is likely to give useful results. This approach to diagnosis can often save a lot of work by highlighting a simple error, or by narrowing down the range of possibilities.

“Contacting your IBM Support Center” on page 876

If you cannot resolve problems that you find when you use IBM Integration Bus, or if you are directed to do so by an error message generated by IBM Integration Bus, you can request assistance from your IBM Support Center.

Resolving problems when installing

Work through the advice provided to help you to deal with problems that can arise when the product is installed.

About this task

The following list describes typical problems when installing, with a corresponding solution or workaround:

Note: When dealing with the Integration Bus component, the installation log files are only available after you have exited the installer.

- Integration Bus component:
 - “Dealing with problems during installation”
 - “Installation process is interrupted” on page 668
 - “Installation process is interrupted in console mode” on page 669
 - “java.lang.UnsatisfiedLinkError” on page 669
 - “RPM query fails” on page 669
 - “Display problems” on page 670
 - “Insufficient temporary space” on page 670
 - “Firewall software reports attempted internet access from an unexpected program” on page 671
 - “WebSphere MQ service fails to install in a Windows domain environment” on page 671
- IBM Integration Toolkit:
 - “Dealing with problems during installation” on page 672
- IBM Integration Explorer:
 - “Dealing with problems during installation” on page 673
- Installation Manager:
 - “Installation Manager hangs” on page 673
 - “Installation Manager does not show the IBM Integration Toolkit components” on page 673
- IBM Integration Bus Launchpad:
 - “IBM Integration Bus Launchpad return code indicates that an installation has failed” on page 674

Dealing with problems during installation Before you begin

Integration Bus component on all operating systems:

Procedure

- **Scenario:** You have problems during the installation of the Integration Bus component.
- **Solution:** Complete the following steps:
 1. If you experience problems when installing on z/OS, see the *Program Directory for IBM Integration Bus for z/OS* on the IBM Integration Bus Library web page.
 2. Refer to the readme file `readme.html` for any late changes to the installation instructions.
 3. If you transferred the installation media to your local or networked filesystem, you might see the following error:
The Install executable launcher was unable to locate its companion shared library.

Ensure that you have read and execute privileges on all installation directories and files. Also ensure that the length of the path to the installation media is within the limits imposed by your operating system.

4. If installation is successful, the installation wizard returns a return code of zero. If a non-zero return code is returned, check the installation log file for errors and explanations. Problems associated with the installation of the Integration Bus component are recorded in the log file `IBM_Integration_Bus_InstallLog.log`, which is stored in your installation directory.

If you accepted the default location during installation, this directory is as follows. The default directory includes the version, release, modification, and fix of the product that you are installing in the format `v.r.m.f` (version.release.modification.fix):

Linux `/opt/ibm/mqsi/v.r.m.f`

UNIX `/opt/IBM/mqsi/v.r.m.f`

Windows 32-bit

`C:\Program Files\IBM\MQSI\v.r.m.f`

Windows 64-bit

`C:\Program Files\IBM\MQSI\v.r.m.f` for the 64-bit version of IBM Integration Bus

`C:\Program Files (x86)\IBM\MQSI\v.r.m.f` for the 32-bit version of IBM Integration Bus

These locations define the default value of `install_dir` on each platform.

5. If you are still unable to resolve the problem, contact your IBM Support Center.

Installation process is interrupted Before you begin

Integration Bus component on all operating systems:

Procedure

- **Scenario:** You are installing the Integration Bus component but the process is interrupted before completion.
- **Explanation:** The process might be interrupted because of a problem, such as a power failure.

- **Solution:** Delete the `install_dir` and all its contents before you restart the program.

Installation process is interrupted in console mode Before you begin

Integration Bus component on Linux:

Procedure

- **Scenario:** Installation in console mode is interrupted by an `InvocationTargetException` error, with the following information included in the stack trace:

```
java.lang.NoClassDefFoundError: sun.awt.X11GraphicsEnvironment (initialization failure)
  at java.lang.J9VMInternals.initialize(J9VMInternals.java:176)
  at java.lang.Class.forNameImpl(Native Method)
  at java.lang.Class.forName(Class.java:182)
  at java.awt.GraphicsEnvironment.createGE(GraphicsEnvironment.java:113)
  at java.awt.GraphicsEnvironment.getLocalGraphicsEnvironment(GraphicsEnvironment.java:92)
  at sun.awt.X11FontManager.isHeadless(X11FontManager.java:499)
  at sun.awt.X11FontManager.getFontPath(X11FontManager.java:779)
  at sun.font.SunFontManager.getPlatformFontPath(SunFontManager.java:3300)
  at sun.font.SunFontManager$11.run(SunFontManager.java:3326)
  at java.security.AccessController.doPrivileged(AccessController.java:229)
  at sun.font.SunFontManager.loadFonts(SunFontManager.java:3322)
  at sun.awt.X11FontManager.loadFonts(X11FontManager.java:451)
  at sun.font.SunFontManager.findFont2D(SunFontManager.java:2359)
  at java.awt.Font.getFont2D(Font.java:509)
  at java.awt.Font.getFamily(Font.java:1198)
  at java.awt.Font.getFamily_NoClientCode(Font.java:1172)
  at java.awt.Font.getFamily(Font.java:1164)
  at Flexeraanx.a1(Unknown Source)
  ...
```

- **Explanation:** A MIT-MAGIC-COOKIE-1 is present on the system. The installer attempts to connect to a display, but the link to the display is not correctly configured.
- **Solution:**
 1. Delete the `install_dir` directory, and its contents.
 2. In the command prompt, run the command `unset DISPLAY`.
 3. Run the installation again in console mode.

java.lang.UnsatisfiedLinkError Before you begin

Integration Bus component on Linux:

Procedure

- **Scenario:** You are using the Integration Bus component graphical interface to install on Linux but the interface does not work correctly.
- **Explanation:** The additional packages that are required to complete the installation are not installed.
- **Solution:** You must install additional packages for the interface to work correctly. For more information about the required additional packages, see Operating system requirements.

RPM query fails Before you begin

Integration Bus component on Linux:

Procedure

- **Scenario:** You start a Red Hat package manager (RPM) query after you have installed the product, but nothing is returned. An information message like the following example might have been reported in the installation log file:

```
(01-Jun-2005 09:02:27), mqsi.Setup,  
  com.ibm.wizard.platform.linux.LinuxProductServiceImpl, wrn,  
  The installer could not successfully add the product information  
  into the RPM database. Installation will continue as this is not  
  critical to the installation of the product.
```
- **Explanation:** It is likely that your system does not have the required RPM support.
- **Solution:** Install the additional RPM build package that is described in Operating system requirements.

Display problems Before you begin

Integration Bus component on Linux and UNIX:

Procedure

- **Scenario:** You try to install the Integration Bus component by using the graphical interface and see one of two common errors reported.
- **Explanation:** Both errors typically occur if you log in remotely, or you switch user ID.
- **Solution:** Address the appropriate error with the following corresponding solution:

Can't open display localhost:1.0

Check that the DISPLAY variable is set to the correct value. If you are logged in locally, the typical value is :0.0 or localhost:0.0.

Connection to ":0.0" refused by server

Run the following command, where *user* is the user ID you are logged in as:

```
xauth merge ~user/.Xauthority
```

If you are unable to correct these errors, contact your systems administrator for further help.

Insufficient temporary space Before you begin

Integration Bus component on Linux and UNIX:

Procedure

- **Scenario:** You install the Integration Bus component and the installation program tries to unpack product files into the temporary file space of the local system. On Linux and UNIX systems, the temporary space is typically located in /tmp.
- **Explanation:** If sufficient file space is not available in this directory, the command might fail without reason and return no comment, or might report a lack of file space.
- **Solution:** Set the environment variable IATEMPDIR to the temporary file system to use. For example, enter the following command:

```
export IATEMPDIR=/targetemp
```

Do not specify a temporary directory that is NFS-mounted from another server; if you do so, the installation might fail because user permission checks made by the installer sometimes report an error that security principals mqm and mqbrkr do not exist on the local machine.

Or if using SUDO, enter the following command:

```
SUDO env IATEMPDIR=/targetemp ./setupaix
```

For more information about checking how much temporary space is required, see Memory and disk space requirements.

Firewall software reports attempted internet access from an unexpected program

Before you begin

Integration Bus component on Windows 32-bit and Windows 64-bit:

Procedure

- **Scenario:** During installation of the Integration Bus component, your firewall software reports attempted internet access from an unexpected program. One of the following program name displays:

```
C:\long_generated_hexadecimal_string\setup32.exe
```

```
C:\long_generated_hexadecimal_string\setup64.exe
```

- **Explanation:** Internet access is not required to complete the installation of the Integration Bus component. However, this component also installs an updated Microsoft C runtime library that might attempt internet access during its installation, which might cause firewall software to report a warning.
- **Solution:** This is normal behavior for this component, and installation completes successfully whether you allow access or not.

WebSphere MQ service fails to install in a Windows domain environment

Before you begin

About this task

- **Scenario:** During installation of the WebSphere MQ component, you get the following error:

```
WebSphere MQ configuration problem
WebSphere MQ is not correctly configured for Windows domain users.
An unexpected error has occurred while validating the security credentials of user 'US\ousdpdghiugan'.
Ensure that the network is operational, and that all required domain controllers are available.
```

- **Explanation:** You must have a Windows domain account enabled to run WebSphere MQ as a service.
- **Solution:** To give the domain user (that you obtained from your domain administrator) the right to run as a service, carry out the following procedure:
 1. Click **Start > Run....** Type the command `secpol.msc` and click **OK**.
 2. Open **Security Settings > Local Policies > User Rights Assignments**. In the list of policies, right-click **Log on as a service > Properties**.
 3. Click **Add User or Group....** Type the name of the user that you obtained from your domain administrator, and click **Check Names**.
 4. If you are prompted by a Windows Security window, type the user name and password of an account user or administrator with sufficient authority, and click **OK > Apply > OK**. Close the Local Security Policy window.

Note: On Windows 7, Windows 8, Windows Server 2008 and Windows Server 2012, the User Account Control (UAC) is enabled by default.

The UAC feature restricts the actions users can perform on certain operating system facilities, even if they are members of the Administrators group. You must take appropriate steps to overcome this restriction.

Dealing with problems during installation

Before you begin

IBM Integration Toolkit on all operating systems:

About this task

- **Scenario:** During installation, you get the following error message:

You cannot update a 64-bit version of Installation Manager to a 32-bit version.

- **Explanation:** A 64 bit version of IBM Installation Manager is already installed. This version is less than v1.6.2, and is not compatible with the 32 bit version that the IBM Integration Toolkit requires.
- **Solution:**
 1. Cancel the installation.
 2. Launch the existing 64 bit version of IBM Installation Manager.
 3. Click **File > Preferences** and select **Updates**.
 4. Select **Search for Installation Manager updates**. Click **OK**.
 5. Click **Update** from the Installation Manager main wizard and click **Yes** when prompt to upgrade to the latest version.
 6. Close the Installation Manager after the update is completed.
 7. Restart the toolkit installation.

Procedure

- **Scenario:** You have problems during the installation of the IBM Integration Toolkit.
- **Solution:** Complete the following steps:
 1. Refer to the readme file `readme.html` for any late changes to the installation instructions.
 2. If installation is successful, the installation wizard returns a return code of zero. If a non-zero return code is returned, check the Installation Manager log file for errors and explanations. Problems associated with the installation of the IBM Integration Toolkit are recorded in the Installation Manager log file `YYYYMMDD_TIME.xml`. Where `YYYYMMDD_TIME` is the date and time of installation. The Installation Manager log file is stored in the following location:

Linux `/var/ibm/InstallationManager/logs`

Windows

On Windows systems, the directory is created in the following default location, however the actual location might differ on your computer:

`C:\ProgramData\IBM\Installation Manager\logs`

3. If you are still unable to resolve the problem, export the Installation Manager related information from **Installation Manager > Help menu > Export Data for Problem Analysis**, and contact your IBM Support Center with the exported information.

Dealing with problems during installation

Before you begin

IBM Integration Explorer on all operating systems:

Procedure

- **Scenario:** You have problems during the installation of the IBM Integration Explorer.
- **Solution:** Complete the following steps:
 1. See the *Program Directory for IBM Integration Bus for z/OS* if you experience problems when installing on z/OS.
 2. Refer to the readme file `readme.html` for any late changes to the installation instructions.
 3. If installation is successful, the installation wizard returns a return code of zero. If a non-zero return code is returned, check the log file for errors and explanations. Problems associated with the installation of the IBM Integration Explorer are recorded in the log file `IBExplorer_install.log`, which is stored in the installation directory. The default installation directories are as follows:

Linux /opt/IBM/IBExplorer

Windows

C:\Program Files\IBM\IBExplorer

Installation Manager hangs

Before you begin

Installation Manager on Linux on x86 and Windows:

Procedure

- **Scenario:** You click **Next** in the Installation Manager when it first opens and the Installation Manager hangs.
- **Solution:** Close the window and reopen it.

Installation Manager does not show the IBM Integration Toolkit components

Before you begin

Installation Manager on Linux on x86 and Windows:

Procedure

- **Scenario:** You are installing the IBM Integration Toolkit but the initial Install Packages page that is displayed by the Installation Manager does not show the IBM Integration Toolkit components.
- **Explanation:** The location of the update repository has not been set correctly.
- **Solution:** Select **File > Preferences** and click **Add Preferences**. Enter the web address or directory where the installation packages are stored or click **Browse** to search for the correct location. Click **OK**. The packages are listed in the Install Packages page.

IBM Integration Bus Launchpad return code indicates that an installation has failed

Before you begin

IBM Integration Bus Launchpad on Windows:

Procedure

- **Scenario:** You install products by using the IBM Integration Bus Launchpad. The Launchpad waits for a return code from each installation wizard that it initiates, but the return code indicates that an installation has failed. The Launchpad reports the error and refers you to the documentation for the product that has failed.
- **Explanation:** Most installation wizards roll back from the point of the error and return your system to the state it was in before the failed attempt, and you can therefore try again after you have corrected the error. However, in this scenario, the Launchpad has already installed one or more products successfully before the error occurred, so does not roll back these installations. When you restart the Launchpad, the status of installed products reflects successful installations from the previous invocation.
- **Solution:** To resolve the failed installation, you must either correct the error and restart the Launchpad, or click **Refresh** and clear the selection of the product that failed.

If you are unable to install the failed product, complete the following steps:

- Refer to the readme file `readme.html` for late changes to the installation instructions.
- If the Integration Bus component fails to install, check the contents of the installation log file `IBM_Integration_Bus_InstallLog.log`, which is stored in your installation directory.

If you accepted the default location during installation, this directory is as follows. The default directory includes the version, release, modification, and fix of the product that you are installing in the format `v.r.m.f` (version.release.modification.fix):

Linux `/opt/ibm/mqsi/v.r.m.f`

UNIX `/opt/IBM/mqsi/v.r.m.f`

Windows 32-bit

`C:\Program Files\IBM\MQSI\v.r.m.f`

Windows 64-bit

`C:\Program Files\IBM\MQSI\v.r.m.f` for the 64-bit version of IBM Integration Bus

`C:\Program Files (x86)\IBM\MQSI\v.r.m.f` for the 32-bit version of IBM Integration Bus

These locations define the default value of `install_dir` on each platform.

- If the IBM Integration Toolkit fails to install, check the contents of the installation log file `YYYYMMDD_TIME.xml`, where `YYYYMMDD_TIME` is the date and time of installation.
- If the IBM Integration Explorer fails to install, check the contents of the installation log file `IBExplorer_install.log`. The log file is stored in the installation directory.

- If WebSphere MQ fails to install, check the contents of `MQV7_install.date_time.log` stored in the temp directory of your home directory.

If you are still unable to resolve the problem, then locate the installation debug file, `IIB_900x_installer_debug.log`, and contact your IBM Support Center.

Related tasks:



Installing

Installation information for IBM Integration Bus is provided in the IBM Integration Bus Installation Guide.



Installing the Integration Bus component

Use the installation wizard to install the Integration Bus component.



Installing the IBM Integration Toolkit

Use the installation wizard graphical interface to install the IBM Integration Toolkit on Windows and Linux on x86.



Installing by using the Windows Launchpad

Use the Windows Launchpad to install the IBM Integration Bus components and the prerequisite products.



Uninstalling

Remove the Integration Bus component, the IBM Integration Toolkit, or the IBM Integration Explorer from your computer.

“Resolving problems when uninstalling”

Work through the advice provided to help you to deal with problems that can arise when the product is uninstalled.



Finding the latest information

Access the latest information for IBM Integration Bus.

Related reference:



IBM Integration Bus logs

IBM Integration Bus writes information to a number of product-specific logs to report the results of actions that you take.

Related information:



IBM Integration Bus requirements

Resolving problems when uninstalling

Work through the advice provided to help you to deal with problems that can arise when the product is uninstalled.

About this task

The following list describes typical problems when uninstalling, with a corresponding solution or workaround:

- Integration Bus component:
 - “The uninstallation process is interrupted” on page 676

The uninstallation process is interrupted

Procedure

- **Scenario:** When you uninstall the Integration Bus component on any distributed system, the process is interrupted, for example by a power failure.
- **Solution:** Delete the *install_dir* and all its contents. You can now reinstall if required.

Related tasks:



Installing

Installation information for IBM Integration Bus is provided in the IBM Integration Bus Installation Guide.

“Resolving problems when installing” on page 667

Work through the advice provided to help you to deal with problems that can arise when the product is installed.



Uninstalling

Remove the Integration Bus component, the IBM Integration Toolkit, or the IBM Integration Explorer from your computer.

Related reference:



IBM Integration Bus logs

IBM Integration Bus writes information to a number of product-specific logs to report the results of actions that you take.

Resolving problems that occur when migrating or importing resources

Use the advice given here to help you to resolve common problems that can occur when you import or migrate resources.

About this task

Migration information is regularly updated on the IBM Integration Bus support web page with the latest details available. Click **Troubleshoot**, then look for a document with a title like “Problems and solutions when migrating”.

Procedure

- “Resolving problems when migrating or importing message flows and message sets” on page 677
 - “Message flows that refer to a migrated user-defined node have connection errors” on page 677
 - “After migration, message flows cannot locate a user-defined node” on page 677
- “Resolving problems when migrating or importing other resources” on page 677
 - “The `mqsिमigratecomponents` command fails with database error BIP2322” on page 678
 - “The **File > Import** menu provides only the option to import a compressed file inside an existing project” on page 678

Related tasks:



Migrating

To migrate a broker domain to IBM Integration Bus Version 9.0, plan your migration strategy, perform pre-migration tasks, migrate your domain components,

then complete post-migration tasks.

Related reference:



mqsistop command

Use the **mqsistop** command to stop the specified component.

Resolving problems when migrating or importing message flows and message sets

Use the advice given here to help you to resolve common problems that can occur when you import or migrate message flows and message sets.

Message flows that refer to a migrated user-defined node have connection errors:

Procedure

- **Scenario:** After migration, all message flows that refer to a migrated user-defined node have errors indicating that connections cannot be made.
- **Explanation:** One possible cause is that the original user-defined node had space characters as part of one or more terminal names. The spaces are wrongly rendered as 'X20'.
- **Solution:** Edit the user-defined node `.msgnode` file, which is available in the same project as the flows that you have migrated. Correct any terminal names that are at fault. Ensure that the names are exactly as the broker node implementation expects.

After migration, message flows cannot locate a user-defined node:

Procedure

- **Scenario:** After migration, message flows cannot locate a user-defined node.
- **Explanation:** One possible cause is that the flows do not have the correct reference internally to a user-defined node.
- **Solution:** Select the **Locate subflow** menu for the node that cannot be located. Using the Browse dialog box, locate the user-defined node (which is in the same project as migrated flows). The message flow now links to the user-defined node correctly and the task list entry is removed when you save the flow.

Resolving problems when migrating or importing other resources

Use the advice given here to help you to resolve common problems that can arise when you import or migrate resources other than message flows.

About this task

- “You see an error message when you recompile a BAR file from a previous version”
- “The **mqsigratecomponents** command fails with database error BIP2322” on page 678
- “The **File > Import** menu provides only the option to import a compressed file inside an existing project” on page 678
- “COBOL compiler errors when importing a copybook” on page 679

You see an error message when you recompile a BAR file from a previous version:

Procedure

- **Scenario:** You have imported a BAR file with your resources from a previous version, and you then choose to refactor those resources to applications and

libraries. You try to recompile a BAR file after the resources have been migrated to applications and libraries, but you see an error message similar to the following example:

TotalPurchaseOrderFlow.msgflow belongs in an application or library and should be deployed within that container and not independently. Create a new BAR file and select the application or library in the Prepare tab of the BAR editor, then select Build and Save. To deploy the resource separately from the application or library, it must be moved into a Message Broker project.

- **Explanation:** If you have reorganized your imported resources into applications and libraries, you cannot rebuild the original BAR file. If a message flow from your original BAR file has been moved into an application in IBM Integration Bus, you must deploy the flow with the new container, or move it to an integration project, from which you can deploy it separately.
- **Solution:** Create a new BAR file and add the application or library that contains the resources that you want to deploy. To deploy a resource like a message flow on its own, move the flow to an integration project, then deploy the flow separately.

The `mqsigratecomponents` command fails with database error BIP2322: About this task

Procedure

- **Scenario:** The `mqsigratecomponents` command fails with database error BIP2322: The 'CREATE TABLE' command is not allowed within a multi-statement transaction in the 'BROKER1' database.
- **Explanation:** If you are using the `mqsigratecomponents` command to migrate a broker that uses a Sybase database, you must modify the database to enable the Data Definition Language (DDL) that the command uses.
- **Solution:** Take the following steps:

1. Log on to ISQL using a system administrator account.
2. Run the following series of commands:

```
1> use master
2> go
1> sp_dboption "BROKER1","ddl in tran",TRUE
2> go
Database option 'ddl in tran' turned ON for database 'BROKER1'.
Run the CHECKPOINT command in the database that was changed.
(return status = 0)
1> use BROKER1
2> go
1> checkpoint
2> go
```

where *BROKER1* is the name of the Sybase broker database.

The **File > Import** menu provides only the option to import a compressed file inside an existing project:

Procedure

- **Scenario:** You have a compressed file that contains message set projects and message flow projects. When you click **File > Import**, you have only the option to import the compressed file inside an existing project, but you want to re-create the message set projects and message flow projects.
- **Solution:** When you export and import files, do not export or import the root directory, which is created for you because of the project file. When you export your message flow and message set projects:
 1. Click **Create only selected directories**.
 2. Clear the project root folder.

3. Select the files and subdirectories as required. The project root folder is selected, but is displayed as gray.

Then, when you import the compressed file:

1. Clear the root (/) folder.
2. Select the files and subfolders as required. The project root folder is selected, but is displayed as gray.

COBOL compiler errors when importing a copybook:

Procedure

- **Scenario:** The report file that is generated by the import contains COBOL compiler errors. For example, you try to import the following copybook:

```
01 AIRLINE-REQUEST.  
...05 CUSTOMER.  
.....10 NAME.....PIC X(45).  
...05 ADDRESS.  
.....10 STREET.....PIC X(30).  
.....10 CITY.....PIC X(25).  
.....10 STATE.....PIC X(20).  
.....10 ZIP-CODE.....PIC X(5).  
...05 FLIGHT-NO.....PIC X(6).  
...05 TRAN-DATE.....PIC X(10).  
...05 COST.....PIC X(7).  
...05 CC-NO.....PIC X(20).  
...05 RESPONSE.  
.....10 STATUS.....PIC X(100).  
.....10 DETAILS.....PIC X(100).
```

The report file contains errors:

```
Line No : 4 IGYDS1089-S "ADDRESS" was invalid. Scanning was resumed at the next area "A" item, level-number, or the start of the next clause.  
Line No : 14 IGYDS1089-S "STATUS" was invalid. Scanning was resumed at the next area "A" item, level-number, or the start of the next clause.
```

- **Explanation:** The errors are caused by the copybook containing field names that are COBOL reserved keywords.
- **Solution:** Change the name of the fields in question, so that they are not COBOL reserved keywords, and retry the import.

Related tasks:

“Resolving problems when migrating or importing message flows and message sets” on page 677

Use the advice given here to help you to resolve common problems that can occur when you import or migrate message flows and message sets.

Related reference:

 **mqsi stop** command

Use the **mqsi stop** command to stop the specified component.

Resolving problems when running commands

Use the advice given here to help you to resolve common problems that can arise when you run commands.

Backing up or restoring the broker returns error BIP1253 or BIP1262

Procedure

- **Scenario:** An error message is generated when you back up or restore a broker:

- The following message is returned when you run the `mqsibackupbroker` command:
BIP1253E Failed to backup the broker '*broker_name*' (error '*error_code*').
- The following message is returned when you run the `mqsirestorebroker` command:
BIP1262E Could not delete the existing configuration of broker '*broker_name*'.

- **Explanation:** Both backup and restore commands require read/write access to the files that contain broker configuration data. Access has failed and the command cannot complete its operation successfully.
- **Solution:** Typically, you might see one of these errors if you are running the backup or restore command against an active broker, and a conflict of access has occurred. If your broker is active, use the `mqsistop` command to stop the broker, and try the operation again.

You might also see one of these errors if the directory to which the command is writing, or from which it is reading, is temporarily inaccessible because of communications or authorization problems. Check that your user ID has access to the directory that you specified in the command.

ReportEvent() error message is issued on Windows when you attempt to run a command

Procedure

- **Scenario:** A ReportEvent () message is generated whenever you attempt to run any `mqs i*` command, The ReportEvent () message is followed by the result of the command itself.
- **Explanation:** The Windows Application Log has become full.
- **Solution:** Clear the Windows Application Log from the Event Viewer.

You want to run a command that uses SSL to administer a remote broker over a secured channel.

Procedure

- **Scenario:** You want to run an `mqs i*` command using SSL to administer a remote broker over a secured channel. Note, that you can do this only for `mqs i*` commands that have the `-n .broker` option.
- **Solution:** In order to connect to a broker using SSL, you must specify the keystore and truststore password for the connection using the `IBM_JAVA_OPTIONS` environment variable. For example:

```
export IBM_JAVA_OPTIONS=-Djavax.net.ssl.keyStorePassword=MYKEYSTOREPASSWORD
-Djavax.net.ssl.trustStorePassword=MYTRUSTSTOREPASSWORD
```

In the same environment, you must then run the command using the `-n` option to specify a `.broker` file that describes the connection you want to establish.

You can export `.broker` files by using the export option in the IBM Integration Explorer. To do this, right-click on the secured broker you want to export a `.broker` file for, and select **Export *.broker**.

A time-out error is issued when you attempt to run a command on AIX when Stack Execution Disable (SED) is enabled

Procedure

- **Scenario:** You want to run an `mqs i*` command on AIX with Stack Execution Disable enabled, but the command times out even if the `-w` option is set to 360.

- **Explanation:** `mqs i*` commands attempt to create a JVM with Java Native Interface (JNI) calls. Because of Stack Execution Disable, the JVM creation fails. For more information about Stack Execution Disable, see AIX Stack Execution Disable.

- **Solution:** Use the `sedmgr` command to create exemption records from Stack Execution Disable. For example, for `mqs ilist`:

```
sedmgr c exempt /opt/mqsi/9.0/bin/mqs ilist
```

The following commands might be affected by this issue:

- `biphttplistener`
- `bipbroker`
- `bipservice`
- `mqs ideploy`
- `mqs ilist`
- `mqs imode`
- `mqs ireloadsecurity`
- `mqs ireportresourcestats`
- `mqs iwebuseradmin`

Related tasks:

Chapter 4, “Troubleshooting and support,” on page 647

If you are having problems with your message flow applications, use the techniques described in this section to help you to diagnose and solve the problems.

“Dealing with problems” on page 666

Learn how to resolve some of the typical problems that can occur.

“Windows: Viewing the local error log” on page 840

The Windows Event Viewer is where IBM Integration Bus writes records to the local system. Use Windows system facilities to view this log.

Related reference:

 `mqs istop` command

Use the `mqs istop` command to stop the specified component.

 `mqs ibackupbroker` command

Use the `mqs ibackupbroker` command to back up the current configuration of a broker.

 `mqs irestorebroker` command

Use the `mqs irestorebroker` command to restore the broker configuration from a backup file.

Resolving problems when running samples

Use the advice given here to help you to resolve common problems that can arise when you run or remove samples.

About this task

Use the following instructions to diagnose the problem.

1. Use WebSphere MQ Explorer to determine which queue the input message is on:
 - a. Start WebSphere MQ Explorer.
 - b. Expand the folders to display the broker queue manager, IB9QMGR.

- c. Click the Queues folder in the queue manager to display its queues.
 - d. Check the Current depth column to identify the queue that is holding the input message. If several messages are stored on one queue, right-click the queue, then click **Browse Messages** to determine if the message that you are interested in is on the queue.
2. Use the following table to identify the problem, and a suggested solution to overcome it. If the sample that you are running does not use a database, ignore the database-related problems listed in the table.
 3. If the table did not help you solve the problem, return to the IBM Integration Toolkit and check the Problems view for error messages. Use this information to solve the problem.
 4. If you created the sample yourself, you must check that all the objects in the sample have been named and configured correctly.

Problem	Reason	Suggested solution
The input message stays on the IN queue.	The broker, the queue manager, the listener, or the message flow itself has stopped.	Check that all the components are running and that the listener for the queue manager is listening on the port for the queue manager. Start any components that are not running.
	An unidentifiable message already on the IN queue cannot be processed by the message flow.	In WebSphere MQ Explorer, right-click the IN queue, then click All tasks > Clear Messages .
The input message goes to the FAIL queue.	The MQInput node cannot identify which parser it must use to parse the message.	If you are using the Enqueue facility in the workbench or the RfhUtil tool that is supplied in SupportPac IH03, you must type all the necessary message header information in the fields in the tool. If you are using the mqsinput.exe tool, you must add the header information to the message file itself.
The input message goes to the SYSTEM.DEAD.LETTER.QUEUE	The queue on which the input message was supposed to be put does not exist.	Ensure that you have created all the queues required for the sample.
You cannot find the input message on any queue.	You have not refreshed the display in WebSphere MQ Explorer, or you have refreshed only some of the queues.	To refresh all the queues in WebSphere MQ Explorer, right-click the Queues folder, then click Refresh . All the queues in the folder are refreshed.
	The input message was passed to a terminal that was not connected to another node, and the message was discarded.	Ensure that all the nodes are connected to each other as required by the sample.

Problem	Reason	Suggested solution
<p>When using a DB2 database, the input message goes to the FAIL queue or the Event Log contains a message saying that the database was not found, or both.</p>	<p>DB2 is not running.</p>	<p>In a DB2 Command Window, enter the following command: <code>db2 start</code></p> <p>If DB2 is already running, you receive the following message: The database manager is already active.</p>
	<p>The message flow is trying to access a database table that is not in the default schema. The name of the default schema is determined by, and is the same as, the user name that is used to access the database. If the table is not in the default schema, and no other schema is specified in the ESQL for the message flow, the message flow looks for the table in the default schema.</p>	<p>In a DB2 Command Window, enter the following commands: <code>DB2 "CONNECT TO database user <i>username</i>"</code> <code>DB2 "CREATE VIEW <i>tablename</i></code> <code>AS SELECT * FROM <i>tableschema.tablename</i>"</code></p> <p>where:</p> <ul style="list-style-type: none"> • <i>username</i> is the user name of the broker • <i>tableschema</i> is the schema that contains the table that the message flow is accessing • <i>tablename</i> is the table that the message flow is accessing
<p>You receive the following error messages when you try to remove a DB2 database on Windows:</p> <p>BIP9830I: Deleting the DB2 Database <i>Your_database_name</i>.</p> <p>BIP9835E: The DB2 batch command failed with the error code SQLSTATE=57019. The database could not be created/deleted. The error code SQLSTATE=57019 was returned from the DB2 batch command.</p>	<p>If you use the DB2 Control Center to perform a query, a connection is opened to the database. This connection stays open until the DB2 Control Center is closed, when the connection is ended.</p>	<p>Close the DB2 Control Center application. To try to remove the sample again, click Yes.</p>

Problem	Reason	Suggested solution
<p>You run a web services sample by using the prebuilt Test Client scenario and it hangs, then times out.</p>	<p>The problem occurs when you have a SOAPInput node that is being called by a SOAPRequest node.</p> <p>The default port that web services use is 7800, and the SOAPRequest nodes are set up to use this port. However, if this port is already in use, for example, by another sample, the port number is automatically incremented by one. Therefore, the default port must also be changed to match.</p>	<p>Issue the following <code>mqsiereportproperties</code> command on one line, to check which port your provider integration server is using:</p> <pre>mqsiereportproperties IB9NODE -e sampleIntegrationServer -o HTTPConnector -n port</pre> <p>where <code>sampleIntegrationServer</code> is the appropriate integration server for the sample that is being run. To verify that the port that the SOAPRequest node is using is the correct port to call the provider flow, change the port of the SOAPRequest nodes to the port that the provider integration server is using by completing the following steps:</p> <ol style="list-style-type: none"> 1. Open the message flow located in the message set project. 2. (Perform this step for all of your SOAPRequest nodes). Open the HTTP Transport tab in the Properties view. If the port is not correct, change the port in the Web service URL property to the correct port for your web services provider or TCP/IP Monitor. 3. Save the message flow. 4. Rebuild and redeploy the broker archive (BAR) file. <p>If you have set up a TCP/IP Monitor, you have already checked which port the web services provider is using, but you must still configure the consumer to send the messages to your TCP/IP Monitor, then rebuild and redeploy the BAR file.</p> <p>Alternatively, you can remove one of the samples that is using the same port, so that only one sample is deployed at a time.</p>
<p>In some samples, the format of the XML output in the Test Client might be displayed in a different format to the format that is shown in the documentation.</p>	<p>In all cases the output data is identical, it is the format that is different.</p>	<p>You can change the format of the output by selecting either View as Source or View as XML Structure from the menu in the Test Client.</p>

Related concepts:



Samples

The IBM Integration Toolkit provides samples that show the features that are available in IBM Integration Bus, and how to use them. This topic provides links to the information about the individual samples.

Related tasks:

Chapter 4, “Troubleshooting and support,” on page 647

If you are having problems with your message flow applications, use the techniques described in this section to help you to diagnose and solve the problems.

“Dealing with problems” on page 666
Learn how to resolve some of the typical problems that can occur.

Resolving problems when creating resources

Use the advice given here to help you to resolve common problems that can occur when you create resources.

About this task

Look for the problem that you have encountered, and follow the guidance provided to recover from the error.

Problems when creating a broker:

- “Message BIP8081 is issued when creating a broker”
- “You cannot create files when creating a broker on AIX”
- “The JCL BIPGEN fails when you create a component on z/OS” on page 686
- “Your DataFlowEngine ends with an abend when you create a broker on HP-UX using Oracle” on page 686

Problems when creating other resources:

- “Error message BIP2624 is issued when creating an integration server” on page 686
- “The Default Configuration wizard fails with invalid argument specified” on page 687

Message BIP8081 is issued when creating a broker Procedure

- **Scenario:** Message BIP8081E is displayed when you are creating a broker, the inserted message does not format correctly, and the broker is not created.
- **Explanation:** This problem occurs because you are not a member of the correct group.
- **Solution:** Read the explanation of message BIP8081, and ask your IBM Integration Bus administrator to give your user ID access to the mqbrkrs group.

You cannot create files when creating a broker on AIX Procedure

- **Scenario:** When you run the `mqsicreatebroker` command on IBM Integration Bus for AIX, the following message is displayed:
BIP8135E Unable to create files. Operating System return code 1
- **Explanation:** The user ID that you create for IBM Integration Bus testing must have a primary group of mqbrkrs. The following example shows an AIX SMIT panel listing the **Change / Show Characteristics of a User**:

Change / Show Characteristics of a User

Type or select values in entry fields.

Press Enter AFTER making all desired changes.

```
[TOP]                                [Entry Fields]
* User NAME                          peterc
  User ID                             [202] #
  ADMINISTRATIVE USER?               false +
  Primary GROUP                       [mqbrkrs] +
  Group SET                           [mqbrkrs,mqm,system,sys> +
```

The JCL BIPGEN fails when you create a component on z/OS Procedure

- **Scenario:** The BIPGEN job fails when you are copying the broker profile BIPBPROF from the PDSE to the file system.
- **Explanation:** The file system might lack the required space, the component profile might not exist, or you might not have the appropriate authority.
- **Solution:** Make the following checks:
 - The file system has sufficient space. You can check how much space is used and how much is free in a file system using the OMVS command `df -P /pathname`. 100 MB is 3 276 800 512 byte sectors.
 - The profile file exists in the PDSE.
 - Your user ID has the appropriate authority to write to the file system.

Your DataFlowEngine ends with an abend when you create a broker on HP-UX using Oracle Procedure

- **Scenario:** Your message flow (DataFlowEngine or DFE) ends with an abend when you create a broker on HP-UX using Oracle.
- **Explanation:** This problem occurs when you have installed DB2 and Oracle on the same computer.
- **Solution:** Remove the DB2 LIL files that are used by IBM Integration Bus. For example, issue the following commands:

```
mv install dir/lib/imbdfdb2v6.lil install dir/lib/imbdfdb2v6.blank
mv install dir/lib/imbdfdb2.lil install dir/lib/imbdfdb2.blank
```

Error message BIP2624 is issued when creating an integration server Procedure

- **Scenario:** When you create an integration server, you get several BIP2624 messages (MQRC=2012 (MQRC_ENVIRONMENT_ERROR)), and no WebSphere MQ messages are processed.
- **Explanation:** You have created the broker to run as a WebSphere MQ trusted application (that is, the broker runs in the same process as the WebSphere MQ queue manager), but the user ID that you specified does not have the required authority.
- **Solution:** If you request the trusted application option on the `mqsicreatebroker` command by specifying the `-t` parameter, perform the appropriate steps for your operating system:

Windows Windows

Using the **-i** parameter on the **mqsicreatebroker** command, specify a service user ID that is a member of WebSphere MQ group mqm.

Linux

UNIX

Linux and UNIX systems

Specify the user ID mqm on the **-i** parameter on the **mqsicreatebroker** command.

The Default Configuration wizard fails with invalid argument specified

Procedure

- **Scenario:** On a Windows system, the Default Configuration wizard fails. The Default Configuration wizard log contains the following error message:
Invalid argument John Smith specified. Argument specified should be well formed. Correct and reissue the command.
- **Explanation:** You have entered a user name that contains one or more spaces. The Default Configuration does not support the use of user names that contain spaces, because the space character can cause problems in communications with other operating systems.
- **Solution:** Use an alternative user name that does not contain any spaces in the Default Configuration wizard.

Resolving problems when renaming resources

Use the advice in this topic to resolve common problems that can occur when you rename resources.

You rename a library that is referenced by an application or another library, but the containing project does not appear to contain the renamed library

Procedure

- **Scenario:** An application or library refers to a library. You rename that library, but the containing application or library now contains no reference to the original or renamed library in the Application Development view.
- **Explanation:** When you rename a library that is referenced by an application or library, the library reference in the containing application or library is not updated automatically.
- **Solution:** To refer to the renamed library, complete the following steps.
 1. Right-click the containing application or library and click **Manage library references**. The Manage library references dialog box opens.
 2. Select the renamed library and clear the original library, then click **OK**.

The containing application or library now contains a reference to the renamed library. For more information, see Adding and removing library references.

Related concepts:



Applications and libraries

Applications and libraries are deployable containers of resources, such as message flows, subflows, message definitions (DFDL, XSD files), JAR files, XSL style sheets, and WebSphere Adapters files.

Related tasks:



Creating an application

Create an application to contain all the resources that are required to develop your solution.



Creating a library

Create a library to contain a logical grouping of related code, data, or both, that can be reused.



Adding and removing library references

To include a library in an application or integration project, you can add a reference to that library to the application or integration project. You can also add a reference to a library from another library.

Resolving problems that occur when you start resources

Use the advice given here to help you to resolve common problems that can occur when you start resources.

About this task

Procedure

- “Resolving problems when starting a broker” on page 689
 - “The broker fails to start because there is not enough space in the Java TMPDIR directory or access permissions for the Java TMPDIR directory are inadequate” on page 690
 - “Diagnostic message ICH408I is issued on z/OS when your broker fails to start” on page 690
 - “Abend code 047 is issued with a diagnostic message” on page 691
 - “Error message BIP2228 is issued when you try to start a second broker on Linux or UNIX” on page 692
 - “MQIsdp client connection is refused by the broker” on page 692
 - “Error messages BIP2604 and BIP2624 are issued when you start a broker or a new message flow” on page 692
 - “When you start the broker through the DataFlowEngine, it cycles continually” on page 693
 - “You have changed your logon password and cannot start your broker on Windows” on page 693
 - “The Java installation is at an incorrect level” on page 693
 - “Authorization errors are reported on z/OS” on page 694
 - “Error message BIP8875 is issued when you start a broker” on page 694
 - “Broker startup on z/OS is very slow” on page 696
- “Resolving problems when starting other resources” on page 697
 - “Resources terminate during startup” on page 697
 - “Resources hang at startup on Windows” on page 697
 - “Error message BIP8048 is issued when you start a component” on page 698
 - “You experience problems with the default configuration” on page 698
 - “A “Not Found” error is issued when you click a link to a specific sample” on page 699
 - “Error message BIP0832 is issued on startup” on page 699
 - “Your integration servers restart repeatedly” on page 700
 - “You cannot tell whether startup is complete on z/OS” on page 700
 - “Abend code 0C1 is issued when you try to start the DataFlowEngine on z/OS” on page 701

- “Error message BIP2604 with return code MQRC_CONNTAG_IN_USE is issued during the start of a message flow on z/OS” on page 701
- “After creating or changing a configurable service, you restart your broker but your message flow does not start, and message BIP2275 is issued in the system log or Windows Event Viewer” on page 702
- “A device allocation error is issued” on page 703
- “Windows fails to recognize IBM Integration Bus digital signatures: “Unknown Publisher”” on page 704
- “The create command fails, and error message BIP8022 is issued” on page 704

Related concepts:



The IBM Integration Bus environment

An integration node (broker) is a set of execution processes that hosts one or more message flows to route, transform, and enrich in flight messages.

Related tasks:

“Starting and stopping a broker” on page 19

Run the appropriate command to start or stop a broker.



Checking APF attributes of bipimain on z/OS

This task is part of the larger task of setting up your z/OS environment.

Related reference:



mqsistart command

Use the **mqsistart** command to start the specified broker if all initial verification tests complete successfully.



mqsicreatebroker command

Use the **mqsicreatebroker** command to create a broker and its associated resources.

Related information:



WebSphere MQ Version 7 product documentation

Resolving problems when starting a broker

Use the advice given here to help you to resolve common problems that can arise when you start a broker.

About this task

When you start a broker by using the **mqsistart** command, the **mqsicvp** command is run automatically to check that the broker environment is set up correctly (for example, the installed level of Java is supported). On Linux and UNIX, the **mqsicvp** command also verifies that the ODBC environment (if specified) is configured correctly. For more information, see **mqsicvp** command.

For advice about specific problems that can occur when you start a broker, see the following section.

- “The broker fails to start because there is not enough space in the Java TMPDIR directory or access permissions for the Java TMPDIR directory are inadequate” on page 690
- “Diagnostic message ICH408I is issued on z/OS when your broker fails to start” on page 690
- “Abend code 047 is issued with a diagnostic message” on page 691

- “Error message BIP2228 is issued when you try to start a second broker on Linux or UNIX” on page 692
- “MQIsdp client connection is refused by the broker” on page 692
- “Error messages BIP2604 and BIP2624 are issued when you start a broker or a new message flow” on page 692
- “When you start the broker through the DataFlowEngine, it cycles continually” on page 693
- “You have changed your logon password and cannot start your broker on Windows” on page 693
- “The Java installation is at an incorrect level” on page 693
- “Authorization errors are reported on z/OS” on page 694
- “Error message BIP8875 is issued when you start a broker” on page 694
- “Warning messages BIP8288-BIP8297 are shown in the syslog when you start a broker” on page 695
- “Broker startup on z/OS is very slow” on page 696
- “Error messages AMQ7626 and BIP8048 are displayed when you try to start a broker” on page 696

**The broker fails to start because there is not enough space in the Java TMPDIR directory or access permissions for the Java TMPDIR directory are inadequate:
Procedure**

- **Scenario:** The broker fails to start and either an error message indicates that insufficient space is available or a BIP4512 exception indicates a `java.lang.NoClassDefFoundError` in the stack trace.
- **Explanation:** This error has two possible causes:
 - The broker uses Java JAR files. When the broker starts, the Java runtime environment extracts the JAR files into a temporary directory, Java TMPDIR. On Linux, UNIX, and z/OS computers, the TMPDIR directory is typically `/tmp`; on Windows computers, it is `c:\temp`. If this directory is not large enough to hold the JAR files, the broker does not start.
 - If the program is packaged as a PAR file, the user must have access to the system temporary directory and adequate space must be available in the temporary directory.
- **Solution:** Use one of the following methods to specify the location of this temporary JAR directory:
 - Use the environment variable `TMPDIR`.
 - Set the system property `java.io.tmpdir`.

Allow at least 50 MB of space per integration server in this directory for IBM Integration Bus components. You might need more space if you deploy large user-defined nodes or other JARs to the broker. You should ensure that all the dependencies of the compute node class are deployed to the broker.

**Diagnostic message ICH408I is issued on z/OS when your broker fails to start:
About this task**

Two scenarios are described here. Choose the appropriate one.

Procedure

- **Scenario 1:** The following diagnostic message is written to the SDSF SYSLOG on z/OS when your broker fails to start:

```

ICH408I USER(MA10USR ) GROUP(TSOUSER ) NAME(OTHER, A N (ANO) 484
/argo/MA10BRK/ENVFILE
-
--TIMINGS (MINS.)--
----PAGING COUNTS---
-JOBNAME STEPNAM PROCSTEP RC EXCP CPU SRB CLOCK SERV PG
PAGE SWAP VIO SWAPS
CL(DIRSRCH ) FID(01D7D3E2E3F1F9002D08000000000003)
INSUFFICIENT AUTHORITY TO LOOKUP
ACCESS INTENT(--X) ACCESS ALLOWED(OTHER ---)

```

- **Explanation:** The started task ID, under which the broker runs, must be in a RACF or z/OS UNIX System Services (USS) group that has rwx permissions on the broker directory. For example, consider a broker that is created under directory /argo/MA00BRK. It runs under started task ID MA00USR. Issuing the **ls -al** command from the root directory / to find the permission bit settings on /argo returns:

```
drwxrwx--- 5 BPXROOT ARG0USR      8192 Jul 30 13:57 argo
```

If you issue the **id MA00USR** command to find the group membership of started task ID MA00USR you see:

```
uid=14938(MA00USR) gid=5(TSOUSER) groups=229(ARG0USR)
```

These results show that the started task ID MA00USR potentially has rwx permissions on subdirectories to /argo, because these permissions are set for both the user and the group that is associated with MA00USR. If the permissions are not set correctly, you see the type of diagnostic message shown in the scenario.

- **Solution:** Make sure that the started task ID, under which the broker runs, is in a RACF or USS group that has rwx permissions on the broker directory.
- **Scenario 2:** The following diagnostic message is written to the SDSF SYSLOG on z/OS when your broker fails to start:

```

ICH408I USER(MA10USR ) GROUP(WMQIBRKS ) NAME(MA10USR
CSFRNG CL(CSFSERV )
INSUFFICIENT ACCESS AUTHORITY
FROM ** (G)
ACCESS INTENT(READ ) ACCESS ALLOWED(NONE )

```

- **Explanation:** During the broker start up process, Java accesses a secure random number generator. Consequently, the broker started task ID needs access to the CSFRNG resource in the CSFSERV class.
- **Solution:** Make sure that the started task ID that the broker runs under has access to the CSFRNG resource in the CSFSERV class.

Abend code 047 is issued with a diagnostic message:

Procedure

- **Scenario:** On z/OS, your broker generates abend code 047 when you try to start it, and the following diagnostic message is written to the SDSF SYSLOG:

```

IEA995I SYMPTOM DUMP OUTPUT 463
SYSTEM COMPLETION CODE=047
TIME=10.53.47 SEQ=00419 CPU=0000 ASID=008E
PSW AT TIME OF ERROR 078D0000 98D09E52 ILC 2 INTC 6B
ACTIVE LOAD MODULE ADDRESS=18D08828 OFFSET=0000162A
NAME=SPECIALNAME
61819987 968995A2 A3618499 89A58599 */argoinst/driver*
F1F46D82 96858261 A4A29961 93979761 *14_boeb/usr/lpp/*
A6949889 61828995 61828997 89948189 *wmqi/bin/bipimai*

```



```

          95                                     *n                *
DATA AT PSW 18D09E4C - 58109948 0A6B5820 B8E95020
GPR 0-3 00000000 0000003C 00000000 00000000
GPR 4-7 18D10300 18D115F0 00000013 00000004
GPR 8-11 18D111CF 18D101D0 18D0BBBE 18D0ABBF
GPR 12-15 98D09BC0 18D101D0 98D09E22 00000000
END OF SYMPTOM DUMP

```

- **Explanation:** System completion code 047 means that an unauthorized program issued a restricted Supervisor Call (SVC) instruction. The diagnostic message also indicates that the program in error was bipimain.
- **Solution:** When you install IBM Integration Bus, issue the command **extattr +a bipimain** from the bin directory of the installation path to give APF authorization to program bipimain.

Error message BIP2228 is issued when you try to start a second broker on Linux or UNIX:

Procedure

- **Scenario:** Error message BIP2228, which mentions semctl in the syslog, is displayed when you try to start a second broker on Linux or UNIX.
- **Explanation:** This error typically indicates a permissions problem with a semaphore used by IBM Integration Bus. A semaphore is created when the first broker starts after a reboot (or after an initial installation), and only members of the primary group of the semaphore creator can access this semaphore. This problem is a consequence of the UNIX System V IPC primitives that are used by IBM Integration Bus.

The BIP2228 message is logged by any broker that is started by a user who is not a member of the primary group of the semaphore creator. The broker tries to access the semaphore, but fails with a permissions-related error. The broker then terminates with the BIP2228 message.

- **Solution:** Avoid this problem by ensuring that all user IDs used to start IBM Integration Bus have the same primary group. If this action is impractical, ensure that all user IDs are members of primary groups of all other user IDs. Contact your IBM Support Center for further assistance.

MQIsdp client connection is refused by the broker:

Procedure

- **Scenario:** When a new MQIsdp client tries to connect to the broker, its connection is refused.
- **Explanation:** MQIsdp Client ID fields must be unique. If a client sends a CONN packet that contains the same Client ID as a currently connected client, the behavior is undefined.
- **Solution:** Ensure that Client IDs are unique.

Error messages BIP2604 and BIP2624 are issued when you start a broker or a new message flow:

Procedure

- **Scenario:** The following messages are written to the USS syslog on z/OS when your integration server, or a newly deployed or started message flow, fails to start:

```

(PMQ1BRK.default)[8]BIP2624E: Unable to connect to queue manager 'PMQ5':
MQCC=2; MQRC=2025; message flow node 'ComIbmMQConnectionManager'
(PMQ1BRK.default)[8]BIP2604E: Node failed to open WebsphereMQ queue
'INPUT1' owned by queue manager 'PMQ5': completion code 2; reason code 2025

```


- **Explanation:** The WebSphere MQ return code of 2025 indicates that the maximum number of concurrent connections has been exceeded. On z/OS, a typical cause of this problem is the setting for IDBACK in the WebSphere MQ CSQ6SYSP macro.
- **Solution:** See the *z/OS System Setup Guide* section of the WebSphere MQ Version 7 product documentation online for information about setting the IDBACK variable.

**When you start the broker through the DataFlowEngine, it cycles continually:
Procedure**

- **Scenario:** When you start the broker through the DataFlowEngine, it continually cycles, starts and stops, and errors BIP2801 and BIP2110 appear in the log:

Unable to load implementation file '/opt/IBM/DistHub/v2/lib/libdhubNBIO.so',
rc=The file access permissions do not allow the specified action.
Message broker internal program error.

- **Explanation:** The permissions on /opt/IBM have a value of 700, meaning that the broker service user ID cannot read the disthub files.
- **Solution:** Set the permissions on /opt/IBM to 755, which is rwxr-xr-x.

**You have changed your logon password and cannot start your broker on
Windows:
Procedure**

- **Scenario:** You have changed your logon password on Windows, and when you start the broker, you see the following error message:

BIP8026E: It was not possible to start the component.
The component could not be started using the service user ID that was supplied when the component was created. Ensure that the service user ID and password are still valid. Ensure that the service user ID has permission to access all of the products directories, specifically the 'bin' and 'log' directories. Check for system messages (on Windows this would be the application event log).

- **Solution:** Change properties of your broker by completing the following steps:
 1. Change your broker by using the command:

```
mqschangebroker brokername -i ServiceUserID -a ServicePassword
```

For example, to change your logon password to user1pwd for user ID user1 on the broker called WBRK_BROKER, use the following command:

```
mqschangebroker WBRK_BROKER -i user1 -a user1pwd
```

2. Restart your broker.

**The Java installation is at an incorrect level:
Procedure**

- **Scenario:** You issue the command to start a broker, but the broker does not start and the system log includes message BIP8892, for example:
Verification failed. The installed Java level 1.3.2 does not meet the required Java level 1.5.
- **Explanation:** The command performs a check on the level of the Java product installed on the computer to ensure that the Java product is at the required level. The check has failed, therefore the broker is not started.
- **Solution:**
 - On distributed systems, you must use the JVM that is supplied with the broker; no other Java product is supported. Check that you have run **mqsiprofile**, or (on Windows only) that you issued the **mqsistart** command from the correct command console.

If you ran the profile command, check the settings of the environment variables in **mqsiprofile**; MQSI_JREPATH, PATH, and the appropriate library path environment variables for your operating system. Change these settings to point to the integrated JVM, and ensure that no other Java installation is in the path.

- On z/OS, install the correct level of Java that is reported in this message, update the broker profile BIPBPROF and submit BIPGEN to refresh the component ENVFILE.

Authorization errors are reported on z/OS:

Procedure

- **Scenario:** You issue the command to start a broker, but the component does not start and the JOBLOG includes message BIP8903, for example:

```
Verification failed. The APF Authorization check failed
for file '/usr/lpp/mqsi/bin/mqsireadlog'.
```

WebSphere Message Broker requires that only bipimain is APF Authorized for successful operation. File '/usr/lpp/mqsi/bin/mqsireadlog' fails that requirement.

If the file indicated in the message is bipimain, use the USS command extattr to ensure that it is APF Authorized.

If the file indicated in the message is not bipimain, use the USS command extattr to ensure that it is not APF Authorized.

For more information, search the information center for "APF attributes".

- **Explanation:** When you start a broker, a series of checks is made to ensure that the broker environment is set up correctly. One or more of the checks for the broker identified in the message has failed, therefore the broker is not started. The errors shown here indicate that the authorizations for some files are incorrect and incompatible with broker operation.

The broker requires that a single file, bipimain, is APF authorized, but the checks indicate that the authorization for file mqsireadlog is incorrect.

- **Solution:** The file bipimain must be APF authorized, and all other files must not have that authorization. Make the required changes to authorization for the files that are identified in the error messages and try the operation again.

Error message BIP8875 is issued when you start a broker:

Procedure

- **Scenario:** You issue the **mqsistart** command to start a broker, but the broker does not start and the system log shows message BIP8875, for example:

```
The component verification for IB9NODE has finished,
but one or more checks failed.
```

- **Explanation:** The command performs a series of checks to ensure that the broker environment, WebSphere MQ queues, and Java are correct and accessible. One or more of the checks for the broker identified in the message has failed, therefore the broker is not started.

- **Solution:** Look in the system log, or in the Application log in the Event Viewer on Windows. Additional messages have been written before this message to indicate which checks have failed. All the checks are performed every time you issue **mqsistart**, therefore all errors are included in the log. Some messages are also returned when you run the command from the command line.

Investigate the one or more errors that have been reported and check return codes and additional details. Look at the complete message content to check for typical causes of the error, and follow the advice given for the messages that you see in the log. View the complete message text in the Diagnostic Messages reference topics.

For example, you might see one or more of the following messages:

- BIP8875W: The component verification for '*component_name*' has finished, but one or more checks failed.
- BIP8877W: The environment verification for component '*component_name*' has finished, but one or more checks failed.
- BIP8883W: The WebSphere MQ verification for component '*component_name*' has finished, but one or more checks failed.
- BIP8885E: Verification failed. Failed to connect to queue manager '*queue_manager_name*'. MQRC: *return_code* MQCC: *completion_code*
- BIP8887E: Verification failed for queue '*queue_name*' on queue manager '*queue_manager_name*' while issuing '*operation*'. MQRC: *return_code* MQCC: *completion_code*
- BIP8888E: Verification failed. Failed to disconnect from queue manager '*queue_manager_name*'. MQRC: *return_code* MQCC: *completion_code*
- BIP8892E: Verification failed. The installed Java level '*level_installed*' does not meet the required Java level '*level_supported*'.
- BIP8893E: Verification failed for environment variable '*variable_name*'. Unable to access file '*file_name*' with user ID '*user_ID*'. Additional information for IBM support: *data1 data2*.
- BIP8895E: Verification failed. Environment variable '*variable_name*' is incorrect or missing.
- BIP8896E: Verification failed. Unable to access the registry with user ID '*user_ID*'. Additional information for IBM support: *data1 data2*
- BIP8897E: Verification failed. Environment variable '*variable_name*' does not match the component name '*component_name*'.
- BIP8903E: Verification failed. The APF Authorization check failed for file '*file_name*'.
- BIP8904E: Verification failed. Failed to start file '*file_name*' with return code '*return_code*' and errno '*error_number*'.

If you cannot resolve the problems that are reported, and you receive a message such as BIP8893 that includes additional information, include these items in the information that you provide when you contact IBM Service.

Warning messages BIP8288-BIP8297 are shown in the syslog when you start a broker:

Procedure

- **Scenario:** Warning messages BIP8288-BIP8297 are shown in the syslog when you start a broker on Linux and UNIX systems.
- **Explanation:** One or more problems were detected with the ODBC environment on Linux and UNIX systems.

When you start a broker by using the **mqsistart** command, the **mqsicvp** command is run automatically to check that the broker environment is set up correctly. On Linux and UNIX systems, this command also verifies that the ODBC environment is configured correctly. Warning messages are written to the syslog in the following situations:

- The file that is referenced by the ODBCINI environment variable does not exist, or the broker does not have access to read or write to the file.
- The ODBCYSINI environment variable is not set.
- The directory that is referenced by the ODBCYSINI environment variable does not contain a file called `odbcinst.ini`, or the broker does not have access to read or write to the file.

- The IE02_PATH is not set.
- **Solution:** Examine the warning messages in the syslog. To view more information, run the **mqsicvp** command from the command line.

Broker startup on z/OS is very slow:

Procedure

- **Scenario:** The broker startup on z/OS takes many minutes, with an extended time taken in loading the imbjplug2 .lil file.
- **Explanation:** When IBM Integration Bus is run in a shared file system sysplex environment, the LPAR that the broker started in does not necessarily own the file system mount points that the broker uses. In this scenario all file accesses have to pass through the coupling facility, which adversely affects performance. During startup the broker accesses and reads many files, and loads many Java class files. All these file operations are slowed, causing longer startup times.
- **Solution:** For optimal startup performance, mount the directories that the broker accesses in the LPAR in which the broker is started. In particular, mount the IBM Integration Bus installation directories locally. Mounting file systems lists the directories that IBM Integration Bus on z/OS needs on the file system at run time; mount them locally for optimal performance.

Error messages AMQ7626 and BIP8048 are displayed when you try to start a broker:

Procedure

- **Explanation:** You see these messages when using a broker on a queue manager that is configured for global coordination with Oracle.
- **Solution:** Start the queue manager manually with the **-si** flag before starting the broker.

:

Related concepts:



The IBM Integration Bus environment

An integration node (broker) is a set of execution processes that hosts one or more message flows to route, transform, and enrich in flight messages.

Related tasks:

“Resolving problems that occur when you start resources” on page 688

Use the advice given here to help you to resolve common problems that can occur when you start resources.

“Starting and stopping a broker” on page 19

Run the appropriate command to start or stop a broker.



Checking APF attributes of bipimain on z/OS

This task is part of the larger task of setting up your z/OS environment.

Related reference:



mqsistart command

Use the **mqsistart** command to start the specified broker if all initial verification tests complete successfully.



mqsicreatebroker command

Use the **mqsicreatebroker** command to create a broker and its associated resources.

“Diagnostic messages” on page 891

Diagnostic messages are listed in this section in numeric order, grouped according to the component to which they relate.

Related information:

 [WebSphere MQ Version 7 product documentation](#)

Resolving problems when starting other resources


Advice for dealing with some common problems that can arise when you start resources other than a broker.

About this task

- “Resources terminate during startup”
- “Resources hang at startup on Windows”
- “Error message BIP8048 is issued when you start a component” on page 698
- “You experience problems with the default configuration” on page 698
- “A “Not Found” error is issued when you click a link to a specific sample” on page 699
- “Error message BIP0832 is issued on startup” on page 699
- “Your integration servers restart repeatedly” on page 700
- “You cannot tell whether startup is complete on z/OS” on page 700
- “Abend code 0C1 is issued when you try to start the DataFlowEngine on z/OS” on page 701
- “Error message BIP2604 with return code MQRC_CONNTAG_IN_USE is issued during the start of a message flow on z/OS” on page 701
- “After creating or changing a configurable service, you restart your broker but your message flow does not start, and message BIP2275 is issued in the system log or Windows Event Viewer” on page 702
- “A device allocation error is issued” on page 703
- “Windows fails to recognize IBM Integration Bus digital signatures: “Unknown Publisher”” on page 704
- “The create command fails, and error message BIP8022 is issued” on page 704
- “Your integration server restarts repeatedly with a JVM Startup failure” on page 704


Resources terminate during startup:

Procedure

-  **Scenario:** The following error message is displayed when you start a broker on Windows:
ServiceName - DLL initialization failure Initialization of the dynamic link library c:\windows\system32\user32.dll failed.
The process is terminating abnormally.
- **Explanation:** This error is issued by Windows when it fails to start a service because it has insufficient storage.
- **Solution:** This error is an operating system problem. Information about how to recover from this problem is available in the Microsoft Developer Network (MSDN). You can access MSDN on the Web at <http://msdn.microsoft.com>.

Resources hang at startup on Windows:

Procedure

-  **Scenario:** You try to start the broker on Windows, but nothing happens in the Event log to show that a connection has started.

- **Explanation:** This problem is typically caused by processes having only one thread. To see if this is the cause, check the Windows Task Manager. If either of the processes `bipconfigmgr.exe` or `dataflowengine.exe` has started, check the number of threads owned by the process. If the process has only one thread, you are likely to encounter this problem.
- **Solution:**
 1. Shut down the broker using the **mqsistop** command and end the process from within the Task Manager.
 2. From the Windows **Start** button, click **Settings > Control Panel**
 3. Double-click **Administrative Tools**
 4. Double-click **Services** to open the Services window. From the list of available services, locate and right-click the broker resource that you want to start (the service name begins with IBM Integration Bus component). Click **Properties** from the menu.
 5. Make a note of the **This Account** setting. Contact the system administrator to obtain the password associated with **This Account**, because these settings are lost when you change values.
 6. Select **System Account** as the **Log On As** option, and select **Allow Service to Interact with Desktop**. These selections allow you to see any hidden dialog messages. Click **OK** to accept the changes.
 7. Restart the resource that is failing and report any subsequent error messages and dialog box messages to your IBM Service Representative.
 8. When your IBM Service Representative has resolved this problem for you, make sure that you restore the **This Account, Password, and Confirm Password** entries to the values that you used when you created the broker.

Error message BIP8048 is issued when you start a component:

Procedure

- **Scenario:** Error message BIP8048 is issued when you start a component.
- **Explanation:** This message indicates that WebSphere MQ is not responding as expected when it tries to start the queue manager. This problem might be because the **strmqm** executable file is not present on UNIX or Linux systems, or the **amqmdain** executable file is not present on Windows, or permissions are incorrect.
- **Solution:** Check that your WebSphere MQ installation is fully functional:
 - On Windows, start the "IBM MQSeries[®]" service.
 - On UNIX or Linux, issue the **strmqm** command to start the queue manager that is associated with this component.

If the check fails, your WebSphere MQ installation is incomplete. This error occurs typically because you have previously installed WebSphere Application Server, which installs an embedded WebSphere MQ component that does not support IBM Integration Bus.

Uninstall WebSphere Application Server, then install the full WebSphere MQ product that is provided with IBM Integration Bus.

You experience problems with the default configuration:

Procedure

- **Scenario:** You have run the Default Configuration wizard but there are problems with the default configuration.
- **Solution:** Use the Default Configuration wizard to remove the default configuration.

If the Default Configuration wizard does not remove the default configuration completely, a wizard failure window opens with instructions on where to find the log. Take the following steps:

1. Follow any advice that is given in the Default Configuration wizard log and try each step again.
2. If retrying fails, restart the computer and run the Default Configuration wizard again.
3. If the wizard still does not remove the default configuration, remove each component manually by performing the following actions in the order shown:
 - a. Issue the following commands to find out which components are installed:
 - **mqsilist** (lists the brokers)
 - **dspmqs** (lists WebSphere MQ components: the queue manager)
 - b. In the IBM Integration Toolkit, delete the connection file `LocalDomain.broker` from the project `LocalProject`.
 - c. In the IBM Integration Toolkit, delete the project `LocalProject`.
 - d. Stop the default broker by issuing the command:
mqsistop IB9NODE
 - e. Delete the default broker by issuing the command:
mqsdeletebroker IB9NODE -w
The **-w** parameter deletes all files related to this broker from the associated work path.
 - f. If you need to remove the queue manager manually, issue the following commands:
`endmqtsr -w -m IB9QMGR`
`endmqm -i IB9QMGR`
`dltmqm IB9QMGR`
4. If you still experience problems after removing the default configuration manually, contact your IBM Support Center.

A "Not Found" error is issued when you click a link to a specific sample: Procedure

- **Scenario:** You see a "Not Found" error when you click a link to a specific sample, indicating that a URL is invalid.
- **Explanation:** You can view sample applications only when you use the information center that is integrated with the IBM Integration Toolkit. If you are viewing a stand-alone or online information center, you cannot access these resources.
- **Solution:** If you want to access samples, ensure that you are viewing the information center from within the IBM Integration Toolkit.

Error message BIP0832 is issued on startup: Procedure

- **Scenario:** The following error message is displayed on startup:
BIP0832E: A class java.io.FileNotFoundException exception occurred which reported the following message: [filepath] (The process cannot access the file because it is being used by another process). Resolve the reason of error and try again.
- **Explanation:** An invalid WebSphere MQ Java Client trace output file has been specified on the Enqueue preferences screen.

- **Solution:**
 1. Open the Enqueue preferences screen by clicking **Windows > Preferences**, then clicking **Enqueue** on the left.
 2. In the **To file** field, specify a valid output file (one that is not read-only or already in use).

Your integration servers restart repeatedly:

Procedure

- **Scenario:** Your integration servers restart repeatedly. The system log might show an error, such as BIP2060.
- **Explanation:** The problem might be caused by:
 - The broker environment variables incorrectly defined
 - Incorrect Loadable Implementation Library directory permissions
 - Incorrect database permissions
 - Invalid user-written LILs
- **Solution:** Check:
 - Environment variables as described in Environment variables after installation
 - File and directory permissions as described in Checking the permission of the installation directory
 - Group memberships as described in Integration servers and “Creating an integration server using the IBM Integration Toolkit or IBM Integration Explorer” on page 36

You cannot tell whether startup is complete on z/OS:

Procedure

- **Scenario:** You cannot tell whether startup has completed on your z/OS system.
- **Solution:** To determine if startup is complete:

1. Check the messages in the system log. The following example shows a system log entry for a startup of a broker with one integration server:

```
S STU3053
$HASP100 STU3053 ON STCINRDR
IEF695I START STU3053 WITH JOBNAME STU3053 IS ASSIGNED TO USER STU3
, GROUP STCGROUP
$HASP373 STU3053 STARTED
+(broker53) 0 BIP9141W: The component was started.
+(broker53) 0 BIP2001I: The WebSphere Business Integration Message Broker
service has started
process ID 33554919.
+(broker53.default) 0 BIP2201I: Integration Server started: process '67109
442
196'; thread '0'; additional information 'broker53', '76eb7f2d-e800-00
00-0080-974c271866d2', 'default', 'true', 'Q4A3', 'false', 'ARG5D651',
'ARG053', '*****', 'false', 'f9c27f2d-e800-0000-0080-974c271866d2'
, '/local/argo/driver/drv3', '/local/argo/tgrp53/broker53'.
+(broker53.default) 0 BIP9137I: A work manager has been registered by R
443
RMS registration services, work manager name is BIP.STU30532.006710919
6.IBM.UA .
```

2. Display the address spaces. The following example shows the display of a broker with one integration server:

```
D OMVS,U=STU3
BPX0040I 18.49.59 DISPLAY OMVS 446
OMVS      000E ACTIVE      OMVS=(68,05)
USER      JOBNAME ASID      PID      PPID STATE   START   CT_SECS
```

```

STU3      STU30531 0069  33554696  33554919 HR   18.49.15    2.217
LATCHWAITPID= 0 CMD=bipbroker broker53
STU3      STU30532 03FD  67109196  67109222 HR   18.49.23    19.816
LATCHWAITPID= 0 CMD=DataFlowEngine broker53 76eb7f2d-e800-00
STU3      STU3053  0036  33554768  83886483 HRI   18.49.08     .653
LATCHWAITPID= 0 CMD=bipservice Q4A3BRK AUTO
STU3      STU30532 03FD  67109222  33554696 1W   18.49.23    19.816
LATCHWAITPID= 0 CMD=bipimain DataFlowEngine broker53 76eb7f2
STU3      STU3053  0036  83886483          1 1WI  18.49.08     .653
LATCHWAITPID= 0 CMD=/local/argo/driver/drv3/bin/bipimain bip
STU3      STU30531 0069  33554919  33554768 1W   18.49.15    2.217
LATCHWAITPID= 0 CMD=bipimain bipbroker broker53

```

Results

The infrastructure main program bipimain is the first process in every address space. It starts bipservice, bipbroker, or DataFlowEngine as the second process in the same address space. For each integration server, an additional address space is started. In this example, only one integration server is available.

Abend code 0C1 is issued when you try to start the DataFlowEngine on z/OS: Procedure

- **Scenario:** The first two IBM Integration Bus address spaces start successfully, but the third address space (the DataFlowEngine) fails to start. The result is an 0C1 abend.
- **Explanation:** The DataFlowEngine address space is generated by the admin agent. If the region size is too small, either because an insufficient region size was specified in the procedure, or because the region size was overridden by the z/OS IEFUSI exit, the DataFlowEngine address space might fail to start, and fails with an 0C1 abend.
- **Solution:**
 1. Use the IPCS command on the dump (move the dump from the file system to a traditional MVS data set if required):

```
verbx vsmdata,'noglobal',jobname(vcp0brk2)'
```

where *vcp0brk2* is the name of the failing job.
 2. Find the string 'VSM LOCAL DATA AREA AT ADDRESS '. The field ELIM gives the available region size and must be greater than 0C800000. If the field SMFEL is not ffffffff, the IEFUSI exit has changed the allowable region size. This value must also be greater than 0C80000.
 3. If you have an IEFUSI exit, check that the exit does not limit the broker address spaces. For example, a commonly used field is OUCBSUBN. This field can be STC or OMVS for the broker, and indicates how the address space was started.

Error message BIP2604 with return code MQRC_CONNTAG_IN_USE is issued during the start of a message flow on z/OS:

Procedure

- **Scenario:** Error message BIP2604 is issued with return code MQRC_CONNTAG_IN_USE during the start of a message flow on z/OS:

```
BIP2604E: Node failed to open WebSphere MQ queue [queue name]
owned by queue manager [queue manager name]
```

This message is output every 30 minutes.

- **Explanation:** On z/OS, WebSphere MQ supports serialized access to shared resources, such as shared queues, through the use of a connection tag (serialization token) when an application connects to a queue manager that participates in a queue sharing group.

In this case, a message flow node fails to connect to the indicated IBM Integration Bus queue manager that is associated with the input queue, because the serialization token that it passed is already in use within the queue sharing group.

This message is for information only. It indicates that serialization is occurring when two or more message flow input nodes try to connect to a queue manager to get messages from a shared queue.

- **Solution:** Check whether another instance of the message flow, or a flow using the same serialization token, is already running. If so, no further action is needed. Otherwise contact your IBM Support Center.

After creating or changing a configurable service, you restart your broker but your message flow does not start, and message BIP2275 is issued in the system log or Windows Event Viewer:

Procedure

- **Scenario:** After creating or changing a configurable service, you restart your broker but your message flow does not start, and message BIP2275 is issued in the system log or Windows Event Viewer, indicating that an error occurred while loading the message flow from the persistent store.
- **Explanation:** When you change or create the configurable service, the connection properties are not fully validated at that point; the broker does not attempt to use them to make a connection. For inbound adapters, the connection is made only when the broker is restarted. Therefore, the properties that you set on the configurable service might be invalid.
- **Solution:** Look at the messages following the BIP2275 message to determine if the message flow failed to start because of invalid connection properties. For example, in SAP you would see message BIP3414 with a reason such as:

```
Connect to SAP gateway failed
Connect_PM GWHOST= invalidhost.test.co, GWSERV=sapgw00, ASHOST= invalidhost.test.co,
  SYSNR=00
LOCATION CPIC (TCP/IP) on local host
ERROR partner not reached (host invalidhost.test.co, service 3300)
TIME Fri Nov 28 15:27:32 2008
RELEASE 640
COMPONENT NI (network interface)
VERSION 37
RC -10
MODULE nixxi_r.cpp
LINE 8728
DETAIL NiPConnect2
SYSTEM CALL SiPeekPendConn
ERRNO 10061
ERRNO TE'
```

followed by a BIP3450 message with an adapter error message such as:

```
Connect to SAP gateway failed
Connect_PM GWHOST= invalidhost.test.co, GWSERV=sapgw00, ASHOST= invalidhost.test.co,
  SYSNR=00
LOCATION CPIC (TCP/IP) on local host
ERROR partner not reached (host invalidhost.test.co, service 3300)
TIME Fri Nov 28 15:27:32 2008
RELEASE 640
COMPONENT NI (network interface)
VERSION 37
RC -10
```

```

MODULE nixxi_r.cpp
LINE 8728
DETAIL NiPConnect2
SYSTEM CALL SiPeekPendConn
ERRNO 10061
ERRNO TE

```

This error was detected by the adapter. The following message describes the diagnostic information that is provided by the adapter:

```

Connect to SAP gateway failed
Connect_PM GWHOST= invalidhost.test.co, GWSERV=sapgw00, ASHOST= invalidhost.test.co,
  SYSNR=00
LOCATION CPIC (TCP/IP) on local host
ERROR partner not reached (host invalidhost.test.co, service 3300)
TIME Fri Nov 28 15:27:32 2008
RELEASE 640
COMPONENT NI (network interface)
VERSION 37
RC -10
MODULE nixxi_r.cpp
LINE 8728
DETAIL NiPConnect2
SYSTEM CALL SiPeekPendConn
ERRNO 10061
ERRNO TE

```

This message suggests that the **applicationServerHost** and **gatewayHost** properties are incorrect. When you have determined which properties are incorrect, use the **mqsichangeproperties** command to correct the properties, or use the **mqsdeleteconfigurablesevice** command to revert to the properties that were deployed in the adapter. Restart the broker.

A device allocation error is issued:

Procedure

- **Scenario:** A device allocation error is issued.
- **Explanation:** A likely cause of this problem is that you do not have the correct permissions set on the component file system for the started task ID.
- **Solution:** Check the system log; if the problem is caused by having incorrect permissions set for the started task ID, you often see an RACF authorization failure message, as shown in the following example.

```

ICH408I USER(TASKID1 ) GROUP(TSOUSER ) NAME(FRED (FRED) 959
  /argo/MA11BRK/ENVFILE
CL(DIRSRCH ) FID(01D7C7E2E3F0F8000F16000000000003)
INSUFFICIENT AUTHORITY TO LOOKUP
ACCESS INTENT(--X) ACCESS ALLOWED(OTHER ---)
IEE132I START COMMAND DEVICE ALLOCATION ERROR
IEA989I SLIP TRAP ID=X33E MATCHED. JOBNAME=*UNAVAIL, ASID=00A8.
D J,BPXAS
IEE115I 11.13.04 2001.212 ACTIVITY 601

```

In this example, the started task ID does not have access to the file system component. The ICH408I message shows:

- The file that the task is trying to access
- The user ID that is trying to access the file
- The permissions that the ID is expecting to have (INTENT in the message)
- The permissions that the ID actually has (ALLOWED in the message)

You can use this information to correct the permissions, then reissue, in this example, the start broker request. This type of message is produced if the user

who is issuing the command (which might be to start the broker, or to submit JCL to start one of the utility jobs) does not have the correct file system permissions for the file system component. Use the ICH408I information to rectify the problem.

Another possible reason for authorization failures is inconsistencies in the RACF definitions for a user ID in the MVS image and the OMVS segment. Also check with your system administrator that the RACF ID that is used on MVS has a corresponding OMVS image created.

Windows fails to recognize IBM Integration Bus digital signatures: "Unknown Publisher":

Procedure

- **Scenario:** User A installs IBM Integration Bus, and can run all programs (mqsi*.exe, bip*.exe, including the command console launcher). User B is created and given appropriate privileges. When User B runs an executable file such as the command console launcher, a window opens and reports that the executable file is from an unidentified publisher.
- **Explanation:** The operating system has not installed the appropriate digital certificates for User B.
- **Solution:** User B must manually install the certificates:
 1. In Windows Explorer, navigate to the bin directory for the IBM Integration Bus installation; for example, on 32-bit systems, C:\Program Files\IBM\MQSI\7.0\bin
 2. Right-click any .exe file to open the Properties window.
 3. Click the Digital Signatures tab.
 4. Select the appropriate certificate from the list, then click **Details**. The Digital Signature Details window is displayed.
 5. Click **View Certificate**. The Certificate window is displayed.
 6. Click **Install Certificate** and complete the steps in the wizard. (Click **Next**, **Next**, **Finish**, then click **OK**.)
 7. Close the Certificate window. You return to the Digital Signature Details.
 8. Select the countersignature from the list, then click the **Details** button. A new Digital Signature Details window is displayed. You can repeat the preceding steps to install other certificates.

The create command fails, and error message BIP8022 is issued:

Procedure

- **Scenario:** Error message BIP8022 is displayed when you use the **mqsicreatebroker** command on Windows, even if the supplied user name and password are correct.
- **Explanation:** The Microsoft component "Shared File and Printer Services" is required.
- **Solution:** Correct this error by installing the "Share File and Printer for Microsoft network" service on the Windows system.

Your integration server restarts repeatedly with a JVM Startup failure:

Procedure

- **Scenario:** When you start the DataFlowEngine it continually starts and stops, displaying errors BIP2116E and BIP7409S in the log:

BIP2116E: Message broker internal error: diagnostic information 'Fatal Error; exception thrown before initialisation completed', 'JVM Startup'
BIP7409S: The broker was unable to create a JVM. The return code indicates that an unrecognized option was passed in to it.

- **Explanation:** When you start an integration server, it creates a Java virtual machine (JVM) for executing Java user-defined nodes, and its creation failed due to an incorrect JVM option.
- **Solution:** Correct the JVM option by completing the following steps:
 1. Stop the broker.
`mqsistop brokerName`
 2. Check the `jvmSystemProperty` value of the failing integration server.
`mqsireportproperties brokerName -e egName -o ComIbmJVMManger -n jvmSystemProperty -f`
 3. If the `jvmSystemProperty` has an invalid option, correct or reset its value.
`mqsichangeproperties brokerName -e egName -o ComIbmJVMManger -n jvmSystemProperty -v "" -f`
 4. Start the broker.
`mqsistart brokerName`

Related concepts:



The IBM Integration Bus environment

An integration node (broker) is a set of execution processes that hosts one or more message flows to route, transform, and enrich in flight messages.

Related tasks:

“Resolving problems that occur when you start resources” on page 688

Use the advice given here to help you to resolve common problems that can occur when you start resources.

“Starting and stopping a broker” on page 19

Run the appropriate command to start or stop a broker.



Checking APF attributes of `bipimain` on z/OS

This task is part of the larger task of setting up your z/OS environment.

Related reference:



mqsistart command

Use the **mqsistart** command to start the specified broker if all initial verification tests complete successfully.



mqsicreatebroker command

Use the **mqsicreatebroker** command to create a broker and its associated resources.



mqsichangeproperties command

Use the **mqsichangeproperties** command to modify broker properties and properties of broker resources.



mqsdeleteconfigurable service command

Use the **mqsdeleteconfigurable service** command to delete a configurable service, such as a JMS provider, JDBC provider, or FTP server, that you have created by using the **mqscreateconfigurable service** command.

Resolving problems when stopping resources

Use the advice given here to help you to resolve problems when you stop resources.

About this task

Procedure

- “You cannot stop the broker”
- “You cannot stop the broker queue manager”
- “The integration server ends abnormally”

You cannot stop the broker

Procedure

- **Scenario:** You run the `mqsistop` command to stop the broker, but the system freezes, and does not stop any of the integration servers.
- **Explanation:** One possible cause is that a message flow is being debugged and it is currently stopped at a breakpoint. IBM Integration Bus regards this as a message in flight situation, and refuses to stop the broker through the normal command.
- **Solution:** Click **Stop debugging** in the Integration Development perspective of the IBM Integration Toolkit. After that operation has completed, the broker stops.

If you cannot stop the debug session, end all integration server processes that are associated with that broker to allow the broker to stop. Your messages are backed out. Click **Stop debugging** after the broker restarts.

You cannot stop the broker queue manager

Procedure

- **Scenario:** You are trying to use the WebSphere MQ `endmqm` command to stop a broker queue manager on a distributed system, but it does not stop.
- **Explanation:** In certain circumstances, attempting to stop a broker queue manager does not cause the queue manager to stop. This situation can occur if you have configured any message flows with multiple threads (you have set the message flow property Additional Instances to a number greater than zero).
- **Solution:** If you want to stop the broker's queue manager, stop the broker by running the `mqsistop` command and specifying the `-q` parameter. (The `-q` parameter is not available on z/OS.) This command runs the WebSphere MQ `endmqm` command on your behalf in a controlled fashion that shuts down the broker and the queue manager cleanly.

The integration server ends abnormally

About this task

Procedure

- **Scenario:** Your integration server processes end abnormally.
- **Explanation:** When integration server processes end abnormally, they are restarted automatically by the `bipbroker` process. If an integration server process fails, it is restarted three times during each five-minute interval. The first five-minute interval begins when the integration server is first started. `RetryInterval` defaults to 5

Remove the integration server from the broker configuration, deploy the broker configuration, then later add the integration server, and redeploy the broker configuration. The row is re-created and `RetryInterval` is set to its default value of 5.

- **Solution:** To change the default value:
 1. Stop the broker.

2. Change the value of the RetryInterval in the database table.
3. Restart the broker.

Related tasks:

“Starting and stopping a broker” on page 19

Run the appropriate command to start or stop a broker.

“Stopping a WebSphere MQ queue manager when you stop a broker” on page 28

If you are preparing to stop a broker, you can stop the broker's WebSphere MQ queue manager at the same time.

Related reference:



mqsistart command

Use the **mqsistart** command to start the specified broker if all initial verification tests complete successfully.



mqsistop command

Use the **mqsistop** command to stop the specified component.

Resolving problems when deleting resources

Use the advice given here to help you to resolve problems when you delete resources.

About this task

You cannot delete a project from your workspace

Procedure

- **Scenario:** You cannot delete a project from your workspace. You get error messages indicating that the containing directory cannot be deleted, or the project file is missing.
- **Explanation:** If you attempt to delete a project, and the directory that contains the project is in use, or you have any files that are contained within the project that have been opened by programs other than the IBM Integration Toolkit, some of the resources in the project are not deleted, but others, including the project file, might be deleted.
- **Solution:** Before you delete a project, make sure that other applications do not have the files open, and that you do not have an open command prompt located in the directory. To recover from this problem, manually delete any remaining files and directories from your workspace directory, then click **Delete** from the project in the IBM Integration Toolkit.

Related tasks:

“Deleting a broker” on page 29

Delete a broker using the command line on the system where the Integration Bus component is installed.



Deleting an integration project

An integration project is the container in which you create and maintain all the resources associated with one or more message flows. These resources are created as files, and are displayed in the project in the Application Development view. If you do not want to retain an integration project, you can delete it.

Related reference:

mqsichangebroker command

Use the **mqsichangebroker** command to change one or more of the configuration parameters of the broker.

mqsideletebroker command

Use the **mqsideletebroker** command to delete a named broker. The command also deletes the queues on the associated queue manager (created when the broker was created). You can also specify that the queue manager is to be deleted.

Resolving problems when developing message flows

Use the advice given here to help you to resolve common problems that can arise when developing message flows.

About this task

- “Resolving appearance problems when developing message flows”
- “Resolving problems when you use CORBA nodes” on page 709
- “Resolving problems when you use Email nodes” on page 711
- “Resolving ESQL problems when developing message flows” on page 713
- “Problems when developing message flows with file nodes” on page 716
- “Resolving problems when you use HTTP and SOAP nodes” on page 720
- “Resolving implementation problems when developing message flows” on page 724
- “Resolving problems when you use IMS nodes” on page 734
- “Resolving mapping and message reference problems when developing message flows” on page 738
- “Resolving trace problems when developing message flows” on page 742
- “Resolving problems when developing message flows with WebSphere Adapters nodes” on page 743
- “Resolving other problems when developing message flows” on page 749

Resolving appearance problems when developing message flows

This topic contains advice for dealing with some common appearance problems that can arise when developing message flows:

The task list does not update when you make corrections to your files:

Procedure

- **Scenario:** The task list does not update any modifications that you make to an ESQL or mapping file. You have made corrections to the files, and while there are no error flags in the file or file icon, the error remains as a task list item.
- **Solution:** To work around this problem, set the following environment variable:

```
JITC_COMPILEOPT=SKIP{org/eclipse/ui/views/tasklist/TaskListContentProvider}  
{resourceChanged}
```

You rename a flow that contains errors, but the task list entries remain:

Procedure

- **Scenario:** When you rename a message flow in the IBM Integration Toolkit for which there are error icons (red crosses) displayed on nodes and connections, those error icons are removed when changes are made. However, the task list entries remain.
- **Solution:** Refresh the Message Flow editor by closing and reopening it.

Terminals on a subflow get out of sync as changes are made:

Procedure

- **Scenario:** You have a message flow that contains subflow nodes. The name or number of terminals on the subflow gets out of sync when changes are made on the subflow itself. The same problem can happen with promoted properties.
- **Solution:** Refresh the Message Flow editor by closing and reopening it. Close the Message Flow editor that contains subflows while the subflows are being changed.

Related concepts:

 [Message flows overview](#)

A message flow is a sequence of processing steps that run in the broker when an input message is received.

Related tasks:

“Resolving problems when developing message flows” on page 708


Use the advice given here to help you to resolve common problems that can arise when developing message flows.

 [Developing integration solutions](#)

IBM Integration Bus provides a flexible environment in which you can develop integration solutions to transform, enrich, route, and process your business messages and data. You can integrate client applications that use different protocols and message formats.

 [Handling errors in message flows](#)

The broker provides basic error handling for all your message flows. If basic processing is not sufficient, and you want to take specific action in response to certain error conditions and situations, you can enhance your message flows to provide your own error handling.

 [Designing a message flow](#)

A message flow can perform a wide range of operations, depending on your business and operational requirements. For best performance and capability, you must design it to include the most appropriate nodes.

Related reference:

 [Message flows](#)

Use the reference information in this section to develop your message flows and related resources.

Resolving problems when you use CORBA nodes

Advice for dealing with common problems that can arise when you develop message flows that contain CORBA nodes.

Before you begin

IBM Integration Bus does not currently support all CORBA operations and types. Ensure that you are passing in a valid IDL file that contains supported operations and types. For a full list of what is supported, see CORBA support. To ensure that the IDL file is valid, run it through an IDL parser.

About this task

- “Error message BIP4891 is issued when you include a CORBARequest node in a message flow” on page 710

- “A CORBA IDL file drop error is issued when you are using an IDL file that contains includes”
- “Error message BIP4910 is issued during deployment when you are using an IDL file that contains includes”

Error message BIP4891 is issued when you include a CORBARequest node in a message flow:

Procedure

- **Scenario:** You have created a message flow that contains a CORBARequest node, but error message BIP4891 is issued, indicating that the node did not receive a valid body.
- **Explanation:** This error message indicates that the CORBARequest node is trying to call an operation but cannot find the required input parameters in the incoming tree. IBM Integration Bus uses the DataObject parser to read and write message from CORBA applications. If you use an input node to pass XML into the CORBARequest node, ensure that it uses the DataObject domain.
- **Solution:** Ensure that the incoming message has the correct structure. If you are using an input node to pass XML into the CORBARequest node, set the Message domain property on the **Input Message Parsing** tab of the input node to DataObject.

A CORBA IDL file drop error is issued when you are using an IDL file that contains includes:

Procedure

- **Scenario:** You have dragged a CORBA IDL file onto the canvas but a CORBA IDL file drop error is issued.
- **Explanation:** If you have imported an IDL file that contains includes, you must drag the top-level IDL file onto the canvas so that the CORBARequest node has all the relevant information. Similarly, when setting properties on the CORBARequest node, if you have imported an IDL file that contains includes, you must select the top-level IDL file in the IDL file property.
- **Solution:** Drag the top-level IDL file onto the canvas, or set the IDL file property on the CORBARequest node to the top-level IDL file.

Error message BIP4910 is issued during deployment when you are using an IDL file that contains includes:

Procedure

- **Scenario:** You are deploying a message flow that contains a CORBARequest node, but error message BIP4910 is issued.
- **Explanation:** This error is issued when you have imported an IDL file that contains includes, but not all the included IDL files have been added to the BAR file. For example, you might have dragged only the message flow onto the integration server. If you have imported an IDL file that contains includes, you must ensure that all the included IDL files are added to the BAR file so that all the relevant information is available to the message flow.
- **Solution:** When you deploy a message flow that contains a CORBARequest node and an IDL file that contains includes, ensure that all included IDL files are added to the BAR file.

If you are dragging a message flow that uses a multifile IDL file onto an integration server in the Integration Development perspective, included IDL files are not deployed. To deploy message flows that use multifile IDL files, you must create a BAR file.

Related concepts:



Common Object Request Broker Architecture (CORBA)

The Common Object Request Broker Architecture (CORBA) is a standard defined by the Object Management Group (OMG) that enables software components written in multiple computer languages and running on multiple computers to work together.



CORBA nodes

Use CORBA nodes to connect IBM Integration Bus with CORBA Internet Inter-Orb Protocol (IIOP) applications.



CORBA support

The CORBA nodes in IBM Integration Bus support a set of types and operations in imported IDL files.

Related tasks:



Connecting to an external CORBA application

Connecting to an external CORBA application involves importing an IDL file, creating a message flow, building a message, and processing the response from the CORBARequest node.



Message Sets: Importing an IDL file

You can use the New Message Definition File wizard in the IBM Integration Toolkit to create a message definition from an IDL file.

Related reference:



CORBARequest node

Use the CORBARequest node to call an external CORBA application over Internet Inter-Orb Protocol (IIOP).

Resolving problems when you use Email nodes

Advice for dealing with common problems that can arise when you develop message flows that contain Email nodes.

About this task

- “A negative email size displays in the local environment”
- “A parsing error displays when you reparse an email attachment as XML” on page 712
- “Removing unwanted null characters from an email” on page 712

A negative email size displays in the local environment:

Procedure

- **Scenario:** The EmailInput node receives an email from an email server that supports Post Office Protocol 3 (POP3) but the size of the email, including any attachments, might display a negative value in the `Root.EmailInputHeader.Size` Multipurpose Internet Mail Extensions (MIME) logical tree.
- **Explanation:** The email server provider that supports POP3 uses the **TOP** command to fetch the headers for the email message and the **LIST** command to determine the size of the entire message. The server then subtracts the two values to determine the size of the message body. If the server reports the size of the entire message incorrectly, you might see a negative number in the local environment Size field.
- **Solution:** You can use a Compute node to calculate the size of the email message and the size of any attachments. The following example ESQL can be

used to calculate the size of the email content and attachments for a multipart MIME document. In this example, the result is stored in the LocalEnvironment:

```
DECLARE CURSOR REFERENCE TO InputRoot.MIME.Parts;
    DECLARE I INTEGER 0;

    FOR SOURCE AS CURSOR.Part[] DO
        SET I = I + LENGTH( SOURCE.Data.BLOB.BLOB);
    END FOR;

    SET OutputLocalEnvironment.Variables.EmailSize = I;
```

For more information about messages that belong to the MIME domain, see *Manipulating messages in the MIME domain*.

A parsing error displays when you reparse an email attachment as XML:

Procedure

- **Scenario:** Your message flow retrieves emails from an email server by using an EmailInput node. The email contains an XML document attachment that you want to reparse. However, when you try to reparse the attachment you receive parsing errors from IBM Integration Bus reporting that you have an invalid XML character.
- **Explanation:** Some email servers might insert carriage return (CR) and line feed (LF) characters at the end of an email. Typically you would want to keep these characters, but in this scenario you must remove them so that you can reparse your XML data.
- **Solution:** Use the following ESQL in a Compute node to remove the CR and LF characters:

```
DECLARE NEWEMAIL BLOB TRIM( TRAILING X'0d0a' FROM InputRoot.
MIME.Data.BLOB.BLOB );
```

Removing unwanted null characters from an email:

Procedure

- **Scenario:** Your email attachment contains unwanted null characters that you would like to remove.
- **Explanation:** Your email attachment might contain null characters that you would like to remove; for example, you intend to reparse the data in the attachment.
- **Solution:** Use the following ESQL in a Compute node to remove the null characters:

```
DECLARE NEWEMAIL BLOB TRIM( TRAILING X'00' FROM InputRoot.MIME.
Data.BLOB.BLOB)
```

Related concepts:



Local environment tree structure

The local environment tree is a part of the logical message tree in which you can store information while the message flow processes the message.

Related tasks:



Processing email messages

You can configure the EmailOutput node to deliver an email from a message flow to an email server that supports Simple Mail Transfer Protocol (SMTP). You can also configure the EmailInput node to retrieve an email from an email server that supports Post Office Protocol 3 (POP3) or Internet Message Access Protocol (IMAP).

Sending emails

You can configure IBM Integration Bus to send an email, with or without attachments, to a static or dynamic list of recipients.

Receiving emails

You can configure the EmailInput node to receive an email, with or without an attachment, from an email server that supports Post Office Protocol 3 (POP3) or Internet Message Access Protocol (IMAP).

Related reference:

EmailInput node

Use the EmailInput node to retrieve an email, with or without attachments, from an email server that supports Post Office Protocol 3 (POP3) or Internet Message Access Protocol (IMAP).

Resolving ESQL problems when developing message flows

This topic contains advice for dealing with some common ESQL problems that can arise when developing message flows:

A Routine not defined error message is issued in ESQL when you move a routine:

Procedure

- **Scenario:** A Routine not defined error message is displayed in ESQL when you move a routine from one schema to another.
- **Explanation:** If a routine that was referenced by code in one schema is moved to another schema, where it is still visible, a false error is generated stating that the routine cannot be resolved.
- **Solution:** Clean the project by clicking **Project > Clean**.

The product fails to respond when you paste ESQL statements from Adobe Reader:

Procedure

- **Scenario:** When you copy and paste certain ESQL statements from Adobe Reader into the ESQL editor, IBM Integration Bus stops responding.
- **Explanation:** This problem occurs when you paste text directly from Adobe Reader into either the ESQL editor or the Java editor.
- **Solution:** To work around this problem, either enter the text manually, or copy and paste it to a text editor (such as Notepad), then perform another copy and paste action from there.

You do not know how message flows handle the code page of ESQL files:

Procedure

- **Scenario:** You do not know how message flows handle the code page of ESQL files.
- **Solution:** The code page of an ESQL file is the code page of the IBM Integration Toolkit on which the file is created. You must deploy a message flow using an ESQL file on an IBM Integration Toolkit with the same code page setting as the ESQL file. When multiple ESQL files are involved in a single compiled message flow (.cmf) file, all these ESQL files must be in the same code page.
See Editor preferences and localized settings for more information.

You do not know the naming restrictions for ESQL procedures and functions:

Procedure

- **Scenario:** You do not know the restrictions for choosing names for ESQL modules or schema scope ESQL and mapping procedures and functions.
- **Solution:** Module and schema scope procedures and functions cannot have names starting with IBM_WBIMB_ because IBM_ is reserved for IBM use, and IBM_WBIMB_ is reserved for IBM Integration Bus.

Error message BIP5431 is issued and the broker fails:

Procedure

- **Scenario:** Error message BIP5431 is displayed and the broker fails.
- **Explanation:** When setting output message properties, you have specified an incorrect physical format name for the message format.
- **Solution:** The name that you specify for the physical layer must match the name that you have defined for it. The default physical layer names are Binary1, XML1 and Text1.

You are unable to call Java from ESQL:

Procedure

- **Scenario:** Your Java class files are not being found.
- **Explanation:** When creating the class files, you have not placed them in the correct location within the system CLASSPATH.
- **Solution:** See the CREATE PROCEDURE statement for further information.

Error message BIP3203 is issued: Format expression is not a valid FORMAT expression for converting expression to type:

Procedure

- **Scenario:** Your format expression contains an unrecognized character for the conversion.
- **Explanation:** Your format expression for a numeric conversion was used to convert to or from a *DATE*, *TIME*, *TIMESTAMP*, *GMTTIME* or *GMTTIMESTAMP* variable. Another possible explanation is that your format expression for a DateTime conversion was used to convert to or from an *INTEGER*, *DECIMAL* or *FLOAT* variable.
- **Solution:** Replace the format expression with one from the applicable types. For more information about valid data types and expressions, see the ESQL reference topic.

Error message BIP3204 is issued: Input expression does not match FORMAT expression. Parsing failed to match:

Procedure

- **Scenario:** You have used an input string that does not match the format expression.
- **Explanation:** Your format expression contains data that does not match the current element of the format expression.
- **Solution:** Either rewrite the format expression to match the input data, or modify the input data to match the format expression. For more information about valid data types and expressions, see the ESQL reference topic.

The CAST function does not provide the expected DST offset for non-GMT time zones:

Procedure

- **Scenario:** You are using the CAST function to convert a string to a TIME variable, in a broker that is running in a time zone other than GMT. The daylight saving time (DST) offset is not correctly calculated.
- **Explanation:** If no time zone is associated with the time string passed to CAST, it is converted to GMT time. If no date is supplied, the current system date is assumed.
- **Solution:** Specify the correct time zone and date. See Formatting and parsing dateTimes as strings for more information.

Error message BIP3205 is issued: The use of a FORMAT expression is not allowed when converting:

Procedure

- **Scenario:** You have used a format expression when it is not applicable, for example when converting from decimal to integer.
- **Explanation:** The use of format expressions is limited to casting between datetime and string values or numeric and string values. Your format expression cannot be applied in this case.
- **Solution:** Either remove the FORMAT clause, or change the parameters. For more information about valid data types and expressions, see the ESQL reference topic.

Related concepts:

 [ESQL overview](#)

Extended Structured Query Language (ESQL) is a programming language defined by IBM Integration Bus to define and manipulate data within a message flow.

 [Message flows overview](#)

A message flow is a sequence of processing steps that run in the broker when an input message is received.

Related tasks:

“Resolving problems when developing message flows” on page 708

Use the advice given here to help you to resolve common problems that can arise when developing message flows.

 [Developing integration solutions](#)


IBM Integration Bus provides a flexible environment in which you can develop integration solutions to transform, enrich, route, and process your business messages and data. You can integrate client applications that use different protocols and message formats.

 [Handling errors in message flows](#)

The broker provides basic error handling for all your message flows. If basic processing is not sufficient, and you want to take specific action in response to certain error conditions and situations, you can enhance your message flows to provide your own error handling.

“Using trace” on page 845

You can use different types of trace to help you with problem determination and troubleshooting.

 [Accessing the Properties tree](#)

The Properties tree has its own correlation name, Properties, and you must use this in all ESQL statements that refer to or set the content of this tree.

Creating destination lists

Create a list of destinations to indicate where a message is sent.

Related reference:

ESQL reference

SQL is the industry standard language for accessing and updating database data and ESQL is a language derived from SQL Version 3, particularly suited to manipulating both database and message data.

Message flows

Use the reference information in this section to develop your message flows and related resources.

Problems when developing message flows with file nodes

Use the advice given here to help you to resolve some common problems that can arise when you develop message flows that contain file nodes.

About this task

- “A file node flow stops processing files and error message BIP3331 or BIP3332 is issued”
- “During processing of a large file, error message BIP2106 is issued or the broker stops because of insufficient memory” on page 717
- “Missing or duplicate messages after recovery from failure in a flow attached to a FileInput node” on page 718
- “No file is created in the output directory after FileOutput node processing” on page 718
- “Output file name overrides have not been applied” on page 719

A file node flow stops processing files and error message BIP3331 or BIP3332 is issued:

Procedure

- **Scenario:** Files in the specified input directory are not being processed. Error message BIP3331 or BIP3332 is issued.
- **Explanation:** The error messages explain that the FileInput node encountered an exception and could not continue file processing. This problem can be caused when the FileInput node cannot move files from its input directory to the archive or backout directory because of file system permissions or another file in the target directory preventing the file to be transferred. In this situation, the node is unable to process input files without losing data so processing stops. Two messages are issued; the first is either BIP3331 or BIP3332 which specifies a second message which describes the cause of the problem in more detail.
- **Solution:** If the first error message issued is BIP3331, stop the flow and resolve the problem. The FileInput node is unable to complete successful processing of the file.
 1. Stop the flow.
 2. Find the error message referenced in the BIP3331. This second error message identifies the problem and the files and directories causing it.
 3. Ensure the broker has the required access to these files and directories.
 4. You might need to move, delete or rename files in the archive or transit directories.

5. Check whether the input file causing the problem has been successfully processed (except for being moved to the archive or backout directory). If it has been successfully processed, remove it from the input directory.
6. Restart the flow.

If the first error message issued is BIP3332, you do not need to stop the flow because the FileInput node has detected the problem before starting file processing. Find the error message referenced in the BIP3332 message. This second error message identifies the problem and the files and directories causing it.

During processing of a large file, error message BIP2106 is issued or the broker stops because of insufficient memory:

Procedure

- **Scenario:** Large input files cause the broker to issue messages, or stop, because insufficient memory is available.
- **Explanation:** The FileInput node can process very large files. Subsequent processing in the flow attached to its Out terminal might require more memory than is available to the broker.
- **Solution:** In most cases, the FileInput node imposes a limit of 100 MB on the records propagated to the attached flow. If your application needs to access large amounts of data, you might need to increase its available memory and reduce the number of available instances. See “Resolving problems with performance” on page 823 for more information. If your application needs to process messages larger than 100 MB, you can override the FileInput node record size limit by taking the following actions:
 - Before starting the broker, set the environment variable `MQSI_FILENODES_MAXIMUM_RECORD_LENGTH` to the required limit as an integer number of bytes, for example:
`SET MQSI_FILENODES_MAXIMUM_RECORD_LENGTH=268435456`
 - When the broker first initializes a FileInput node, it will use the environment variable value instead of the default value of 100 MB. Subsequent changes to the environment variable value will not affect the broker limit until the broker is restarted.
 - If the Record detection property is set to `Whole File`, the limit applies to the file size. If the Record detection property is set to `Fixed Length` or `Delimited`, the limit applies to the record size. The FileOutput node is not affected by changes to this limit.
 -

Note: Increasing the FileInput node record size limit might require additional broker resources, particularly memory. You must thoroughly test and evaluate your broker's performance when processing these files. The number of factors that are involved in handling large messages make it impossible to provide specific broker memory requirements.

You can also reduce the memory required to process the file's contents in the following ways:

- If you are processing a whole file as a single BLOB, split it into smaller messages by specifying on the **Records and Elements** tab of the FileInput node's properties:
 - A value of `Fixed Length` in the Record detection property
 - A large value in the Length property, for example 1000000.

- If you are writing the file's contents to a single output file, specify Record is Unmodified Data in the FileOutput node's Record definition property; this reassembles the records in an output file of the same size as the input file. Wire the FileInput node's End of Data terminal to the FileOutput node's Finish File terminal. Configure the flow to have no additional instances to ensure that the output records arrive in sequence.
- If you are processing large records using the techniques shown in the Large Messaging sample, ensure that you do not cause the integration server to access the whole record. Avoid specifying a value of \$Body in the Pattern property of a Trace node.
- If you have specified a value of Parsed Record Sequence in the FileInput node's Record definition property, the broker does not limit the size of the record. If subsequent nodes in the message flow try to access an entire large record, the broker might not have sufficient memory to allow this and stop. Use the techniques in the Large Messaging sample to limit the memory required to handle very large records.

Missing or duplicate messages after recovery from failure in a flow attached to a FileInput node:

Procedure

- **Scenario:** After the failure of a message flow containing a FileInput node processing the input file as multiple records, a subsequent restart of the flow results in duplicate messages being processed. If the flow is not restarted, some input records are not processed.
- **Explanation:** If a record produces a message which causes the flow to fail and retry processing does not solve the problem, the node stops processing the file and moves it to the backout directory. Records subsequent to the failing message are not processed. The FileInput node is not transactional; it cannot roll back the file input records. Transactional resources in the attached flow can roll back the effects of the failing input record but not preceding records. Records before the failing record will have been processed but records subsequent to the failing record will not have been processed. If you restart the flow by moving the input file from the backout directory to the input directory, messages from records preceding the point of failure are duplicated.
- **Solution:** If the input messages have unique keys, modify your flow to ignore duplicate records. If the messages do not have unique keys but each input file has a unique name, you can modify your flow to form a unique key based on the file name and record number. Define a database table and add a Database node to your flow to record the key of each record that is processed. Add a DatabaseRoute node to filter input messages so that only records without keys already in the database are processed. See the Simplified Database Routing sample to understand how to use the DatabaseRoute node to filter messages.
If you cannot generate unique keys for each record, split your flow into two separate flows. In the first flow, wire the FileInput node to an MQOutput node so that each input record is copied as a BLOB to a WebSphere MQ queue. Ensure there are adequate WebSphere MQ resources, queue size for example, so that the first flow does not fail. In the second flow, wire an MQInput node to the flow previously wired to your FileInput node. Configure the MQInput and other nodes to achieve the desired transactional behavior.

No file is created in the output directory after FileOutput node processing:

Procedure

- **Scenario:** A file created by the FileOutput node does not appear in the output directory. The node is configured so that the Record definition property has a

value of Record is Unmodified Data, Record is Fixed Length Data, or Record is Delimited Data and the flow runs one or more times.

- **Explanation:** The FileOutput node accumulates messages, record by record, in an incomplete version of the output file in the transit subdirectory of the output directory. It moves the file from the transit subdirectory to the output directory only when it receives a message on its Finish File terminal; at this point, the file is complete. If the node's input processing fails before a message is sent to the Finish file terminal, the file remains in the transit directory. The file might be completed by a subsequent flow if it uses the same file name and output directory; if this does not happen, the file is never moved to the output directory.
- **Solution:** If you need to ensure that incomplete files are moved to the output directory if the input flow fails, wire the input node's Failure terminal to the FileOutput node's Finish File terminal, in addition to all other flows that are wired to this terminal.

If you need all output files to be available for a downstream process at a particular time or after a particular event, wire a separate flow to the FileOutput node's Finish File terminal to send a message at that particular time or on that particular event. If duplicate messages which identify the same file are sent to the Finish File terminal, the FileOutput node ignores them.

If your flows use the Request directory property location, Request file name property location (default Directory and Name in the \$LocalEnvironment/Destination/File folder), or \$LocalEnvironment/Wildcard/WildcardMatch, ensure that messages sent to the Finish File terminal contain the correct elements and values to identify the output file and directory.

Output file name overrides have not been applied:

Procedure

- **Scenario:** The message elements set in the flow to override the output file name or directory values specified in the FileOutput node's **Basic** properties have not been applied. The output file is created using the name and directory set in the FileOutput node's **Basic** properties.
- **Explanation:** One of the following might be the cause of this problem:
 - The message sent to the FileOutput node does not contain the expected changes.
 - The FileOutput node is configured to use different elements in the message from the ones set to the new values.
 - Not all messages contain the overriding values.
- **Solution:** Use the debugger or a Trace node inserted in front of the FileOutput node's In terminal to check that the expected overriding values appear in the correct message elements. If they do not, check that the Compute mode property has been set correctly in Compute nodes that are upstream in the flow; for example, if \$LocalEnvironment/File/Name has not changed following a Compute node, check that the Compute node has its Compute mode property set to LocalEnvironment and Message.

If the message elements are set correctly, check that the FileOutput node's Request directory property location and Request file name property location properties identify the correct elements in the message.

If you have specified Record is Unmodified Data, Record is Fixed Length Data, or Record is Delimited Data in the FileOutput node's Record definition property, ensure that messages that go to the Finish File terminal have the same override values as those that go to the In terminal. Unless you do this, the Finish file terminal message and the In terminal messages will apply to different files.

Related concepts:



How the broker processes files

The broker reads files with the FileInput, FTEInput, CDInput, and FileRead nodes, and writes files with the FileOutput, CDOOutput, and FTEOutput nodes.



How multiple file nodes share access to files in the same directory

IBM Integration Bus controls access to files so that only one file node at a time can read or write to a file.



Using local environment variables with file nodes

You can use fields in the local environment to dynamically alter the behavior of the FileInput, FileOutput, FTEInput, and FTEOutput nodes. You can also find what values the output nodes used to process the file.

Related tasks:



Handling errors in message flows

The broker provides basic error handling for all your message flows. If basic processing is not sufficient, and you want to take specific action in response to certain error conditions and situations, you can enhance your message flows to provide your own error handling.



Reading files

Use the FileInput, CDInput, FTEInput, and FileRead nodes to read files.



Writing a file

Use the FileOutput, CDOOutput, and FTEOutput nodes to write files.

Related reference:



FileInput node

Use the FileInput node to process messages that are read from files.



FileOutput node

Use the FileOutput node to write messages to files.

Resolving problems when you use HTTP and SOAP nodes

Use the advice that is given here to help you to resolve common problems that can arise when you develop web Services message flows that contain HTTP and SOAP nodes.

About this task

If you experience problems when you use HTTP and SOAP nodes in message flows, complete the instructions for the following scenarios to diagnose and solve the problem.

- “Error message BIP3689 is issued when you deploy or restart an integration server” on page 721
- “Warning message BIP3690 is issued when you deploy or restart an integration server” on page 721
- “A HandshakeException is issued when you use an HTTPRequest node to make an HTTPS call” on page 721

Use the replies to the following questions to assist you in diagnosing problems with HTTP or SOAP nodes:

- “How do I tell which listener the HTTP and SOAP nodes are using?” on page 722
- “How do I collect HTTPListener trace?” on page 723

Error message BIP3689 is issued when you deploy or restart an integration server:

Procedure

- **Scenario:** You configure Web Services Reliable Messaging (WS-RM) on one or more nodes in your message flow configuration. When you attempt to deploy the configuration to an integration server, or to restart the integration server, you see a BIP3689 error message.
- **Explanation:** Two or more nodes in your deployment configuration have the same value for the Path suffix for URL property, and at least one of these nodes is using WS-RM. Using the same URL twice is not allowed, as one integration server might get messages intended for another integration server, therefore breaking the WS-RM message ordering.
- **Solution:** Review the values for the Path suffix for URL property in the nodes that you plan to deploy to the same broker. Ensure that the URL for a node that is using WS-RM is not used anywhere else in the deployment configuration.

Warning message BIP3690 is issued when you deploy or restart an integration server:

Procedure

- **Scenario:** Your integration server contains SOAP or HTTP input nodes in one or more message flows. When you attempt to deploy the integration server, or to restart it, you see a BIP3690 warning message.
- **Explanation:** This warning message indicates that you are using the same value for the Path suffix for URL property in both SOAP and HTTP nodes in your deployed topology. You do not see the warning if you use the same URL in two or more HTTP nodes, or in two or more SOAP nodes.

For example, you might deploy the same flow to multiple integration servers for scaling and performance reasons; this configuration is valid and does not trigger the warning. However, using the same URL across SOAP and HTTP nodes might result in unexpected behavior, and always triggers a warning, even if the configuration is valid.

If you are using SOAP nodes and HTTP nodes in message flows on a single broker, you can choose to handle HTTP messages by using either the broker listener or embedded integration server listeners. If a listener in your configuration receives messages that both SOAPInput and HTTPInput nodes might get, you must carefully check the URL specifications in these nodes. If both URL specifications match an incoming message, the wrong type of node might get the message, and processing might fail or produce unexpected results. This situation occurs if you specify identical values for the Path suffix for URL properties of the HTTPInput node and the SOAPInput node. It can also occur if you use wildcards in either or both specifications, and an incoming message matches both properties.

- **Solution:** Check the URL specifications in your HTTP and SOAP input nodes and confirm that message routing works as intended.

A HandshakeException is issued when you use an HTTPRequest node to make an HTTPS call:

Procedure

- **Scenario:** You are trying to make an HTTPS call to an external web service from your IBM Integration Bus message flow by using an HTTPRequest node. You receive the following error:

```
javax.net.ssl.SSLHandshakeException:  
com.ibm.jsse2.util.h: PKIX path building failed:
```

```
java.security.cert.CertPathBuilderException:  
PKIXCertPathBuilderImpl could not build a valid CertPath.;
```

internal cause is:

```
java.security.cert.CertPathValidatorException:  
The certificate issued by OU=Class 3 Public Primary Certification Authority,  
O="VeriSign, Inc.", C=US is not trusted;
```

internal cause is:

```
java.security.cert.CertPathValidatorException: Certificate chaining error
```

- **Explanation:** The broker cannot build the entire certificate path. The keystore of this broker must contain all the certificates in this chain of certificate authorities (CA). For a broker to verify the digital signature on a signed certificate, the keystore must contain the public key of the certificate authority (CA) that issued that certificate. If this public key is itself issued on a signed certificate, the keystore must contain the public key of the CA that issued that certificate. This chain continues until the broker reaches a root certificate authority that issues a self-signed certificate.
- **Solution:** Verify that you added all the required certificates to your keystore. If any of the components in the certificate chain are missing from your keystore, re-create your keystore by using keytool with the genkey option, then reimport your application certificates.

How do I tell which listener the HTTP and SOAP nodes are using?:

Procedure

- **Scenario:** HTTP and SOAP nodes that you include in a message flow can use either the broker-wide listener or the embedded listener that is defined to the integration server to which the containing message flow is deployed. Both listeners can handle both HTTP and HTTPS messages by handling the different message types on different ports.
- **Solution:** Use the `mqsireportproperties` command to check the properties that define what listener is in use.
 1. Check whether the broker listener is disabled:

```
mqsireportproperties IB9NODE -b httpListener -o HTTPListener -n startListener
```

If this property is false, all HTTP and SOAP nodes in all integration servers are using an embedded listener.
 2. If the broker listener is active, you must check the specific integration server. For example, check whether the listener in EG_A is being used to process HTTP messages for HTTP nodes:

```
mqsireportproperties TEST -e EG_A -o ExecutionGroup -n httpNodesUseEmbeddedListener
```

If this property is true, all HTTP nodes that you deploy to this integration server are using the embedded listener.
Check all integration servers for which you want to know this information.
Check whether the listener in EG_A is being used to process HTTP messages for SOAP nodes:

```
mqsireportproperties TEST -e EG_A -o ExecutionGroup -n soapNodesUseEmbeddedListener
```

If this property is true (the default value), all SOAP nodes that you deploy to this integration server are using the embedded listener.
Check all integration servers for which you want to know this information.
 3. If both properties are false, all HTTP and SOAP nodes that you deploy to all integration servers are using the embedded listener. The value for the integration server property is ignored.

Results

If you are experiencing problems with message flows that contain HTTP or SOAP nodes, and you want to collect trace information to provide to your IBM Support Center, trace the integration server. If you are using the broker-wide listener for HTTP or SOAP nodes, also trace the HTTPListener component.

How do I collect HTTPListener trace?:

About this task

To gather information about HTTP nodes and listeners, you must start trace, run your message flows, then retrieve and format the trace information.

Trace the integration server and the HTTPListener component:

Procedure

- Start trace for the integration server.

For example:

```
mqsichangetrace IB9NODE -t -e default -l debug
```

- If you are using the broker-wide listener for one or more integration servers, start trace for the HTTPListener component in one of the following two ways:

- Trace all broker components:

1. Run the **mqsichangetrace** command to start trace with the following options:

```
mqsichangetrace component -t -b -l debug
```

where *component* is the broker name.

2. Retrieve the HTTPListener trace log by using the **mqsireadlog** command with the HTTPListener qualifier for the **-b** parameter.

For example:

```
mqsireadlog brokerName -t -b httplistener -f -o listenertrace.xml
```

3. Format the trace log by using the **mqsiformatlog** command to view its contents.

- Trace only the HTTPListener component:

1. Run the **mqsichangeproperties** command to start trace with the following options:

```
mqsichangeproperties brokerName -b httplistener -o HTTPListener  
-n traceLevel -v debug
```

2. Retrieve and format the HTTPListener trace log as shown in the previous example.

What to do next

Save the trace output so that you can send it to the IBM Support Center, if requested.

Turn off trace when you finish collecting information to avoid affecting the performance of the broker.

:

Related concepts:

Processing HTTP messages

Hypertext Transfer Protocol (HTTP) is an Internet protocol that is used to transfer and display hypertext and XML documents on the Web.

HTTP listeners

You can choose between broker-wide listeners and integration server (embedded) listeners to manage HTTP messages in your HTTP or SOAP flows. Learn about the two types of listener, how ports are assigned to them, and how you can switch from one to the other for individual integration servers.

Web Services Reliable Messaging

IBM Integration Bus supports WS-RM (Web Services Reliable Messaging), which allows two systems to reliably exchange messages with each other.

Related reference:

Broker-wide HTTP listener parameters

Select the resources and properties that are associated with the broker-wide HTTP listener that you want to change.

`mqsichangeproperties` command

Use the `mqsichangeproperties` command to modify broker properties and properties of broker resources.

`mqsichangetrace` command

Use the `mqsichangetrace` command to set the tracing characteristics for a broker.

HTTPInput node

Use the HTTPInput node to receive an HTTP message from an HTTP client for processing by a message flow.

HTTPReply node

Use the HTTPReply node to return a response from the message flow to an HTTP client. This node generates the response to an HTTP client from which the input message was received by the HTTPInput node, and waits for confirmation that it has been sent.

Resolving implementation problems when developing message flows

Use the advice given here to help you to resolve some common problems that can arise when running message flows.

About this task

- “Messages are directed to the Failure terminal of an MQInput node” on page 725
- “Error message BIP2211 is issued on z/OS by the MQInput node” on page 725
- “Messages enter the message flow but do not exit” on page 725
- “Your integration server is not reading messages from the input queues” on page 727
- “The integration server ends while processing messages” on page 728
- “Your integration server hangs, or ends with a core dump” on page 728
- “Your XSLTransform node is not working after deployment and errors are issued indicating that the style sheet could not be processed” on page 729
- “Output messages are not sent to expected destinations” on page 729
- “You experience problems when sending a message to an HTTP node's URL” on page 729

- “When using secure HTTP connections, you change a DNS host’s destination but the broker is using a cached DNS host definition” on page 730
- “The TimeoutControl node issues error message BIP4606 or BIP4607 when the timeout request start time that it receives is in the past” on page 731
- “You are using a TimeoutControl node with a TimeoutNotification node, with multiple clients running concurrently, and messages appear to be being dropped” on page 731
- “Error message BIP5347 is issued on AIX when you run a message flow that uses a message set” on page 731
- “Error message BIP2130 is issued with code page value of -1 or -2” on page 732
- “The integration server restarts before an MQGet node has retrieved all messages” on page 732

Messages are directed to the Failure terminal of an MQInput node:

Procedure

- **Scenario:** Messages that are received at a message flow are directed immediately to the Failure terminal on the MQInput node (if it is connected), or are rolled back.
- **Explanation:** When a message is received by WebSphere MQ, an error is signalled if the following conditions are all true:
 - The MQInput node requests that the message content is converted (the Convert property is set to yes on the node).
 - The message consists only of an MQMD followed by the body of the message.
 - The message format, as specified in the MQMD, is set to MQFMT_NONE.

This error causes the message to be directed to the Failure terminal.

- **Solution:** In general, you do not need to request WebSphere MQ to convert the message content, because the broker processes messages in all code pages and encodings that are supported by WebSphere MQ. Set the Convert property to no to ensure that messages flow from the MQInput node to successive nodes in the message flow.

Error message BIP2211 is issued on z/OS by the MQInput node:

Procedure

- **Scenario:** The following error message is issued by the MQInput node, indicating an invalid attribute:

BIP2211: (Invalid configuration message containing attribute value [attribute value] which is not valid for target attribute [target attribute name], object [object name]; valid values are [valid values])

- **Explanation:** On z/OS, WebSphere MQ supports serialized access to shared resources, such as shared queues, through the use of a connection tag (serialization token) when an application connects to the queue manager that participates in a queue sharing group. In this case, an invalid attribute has been specified for the z/OS serialization token.
- **Solution:** Check that the value that is provided for the z/OS serialization token conforms to the rules as described in the *Application Programming Reference* section of the WebSphere MQ Version 7 product documentation online.

Messages enter the message flow but do not exit:

Procedure

- **Scenario:** You have sent messages into your message flow, and they have been removed from the input queue, but nothing appears at the other end of the message flow.

- **Explanation:** Several situations might cause this error to occur. Consider the following scenarios to try to identify the situation that is causing your failure:
 1. Check your message flow in the IBM Integration Toolkit.

You might have connected the MQInput node Failure terminal to a successive node instead of the Out terminal. The Out terminal is the *middle* terminal of the three. Messages directed to an unconnected Out terminal are discarded.
 2. If the Out terminal of the MQInput node is connected correctly to a successive node, check the broker's local error log for an indication that message processing has been ended because of problems. Additional messages give more detailed information.

If the Failure terminal of the MQInput node has been connected (for example, to an MQOutput node), these messages do not appear.

Connecting a node to a Failure terminal of another node indicates that you have designed the message flow to deal with all error processing. If you connect a Failure terminal to an MQOutput node, your message flow ignores all errors that occur.
 3. If the Out terminal of the MQInput node is connected correctly to a successive node, and the local error log does not contain error messages, turn user tracing on for the message flow:
 - a. Open the IBM Integration Explorer.
 - b. In the Navigator view, expand the Integration Nodes folder.
 - c. Right-click the message flow, and click **User TraceNormal**.

This action produces a user trace entry from only the nodes that the message visits.

On distributed systems, you can retrieve the trace entries by using the **mqsireadlog** command, format them by using the **mqsiformatlog** command, and view the formatted records to check the path of the message through the message flow.

z/OS

For z/OS, edit and submit the BIPRELG job in COMPONENTPDS to execute the **mqsireadlog** and **mqsiformatlog** commands to process traces.
 4. If the user trace shows that the message is not taking the expected path through the message flow, increase the user trace level to Debug by selecting the message flow, right-clicking it, and clicking **User Trace > Debug**.

Send your message into the message flow again. Debug-level trace produces much more detail about why the message is taking a particular route, and you can then determine the reasons for the actions taken by the message flow.

Do not forget to turn tracing off when you have solved the problem, because performance might be adversely affected.
 5. If the MQPUT command to the output queue that is defined on the MQOutput node is not successful (for example, the queue is full or **put** is disabled), the final destination of a message depends on:
 - Whether the Failure terminal of the MQOutput node is connected.
 - Whether the message is being processed transactionally (which in turn depends on the transaction mode setting of the MQInput node, the MQOutput node, and the input and output queues).
 - Whether the message is persistent or nonpersistent. When transaction mode is set to the default value of Automatic, message transactionality is

derived from the way that it was specified at the input node. All messages are treated as persistent if transaction mode=yes, and as nonpersistent if transaction mode=no.

In general, if a path is not defined for a failure (that is, neither the Catch terminal nor the Failure terminal of the MQInput node is connected):

- Non-transactional messages are discarded.
 - Transactional messages are rolled back to the input queue to be tried again:
 - If the backout count of the message is less than the backout threshold (BOTHRESH) of the input queue, the message is tried again and sent to the Out terminal.
 - When the backout count equals or exceeds the backout threshold, one of the following might happen:
 - The message is placed on the backout queue, if one is specified (using the BOQNAME attribute of the input queue.)
 - The message is placed on the dead-letter queue, if there is no backout queue defined or if the MQPUT to the backout queue fails.
 - If the MQPUT to the dead-letter queue fails, or if there is no dead-letter queue defined, then the message flow loops continuously trying to put the message to the dead-letter queue.
 - If a path is defined for the failure, then that path defines the destination of the message. If both the Catch terminal and the Failure terminal are connected, the message is propagated through the Catch terminal.
6. If your message flow uses transaction mode=yes on the MQInput node properties, and the messages are not appearing on an output queue, check the path of the message flow.
- If the message flow has paths that are not failures (but that do not end in an output queue), either:
 - The message flow has not failed and the message is not backed out.
 - The message flow is put to an alternative destination (for example, the Catch terminal, the dead-letter queue, or the queue's backout queue).
 - Check that all possible paths reach a final output node and do not reach a dead end. For example, check that you have:
 - Connected the Unknown terminal of a Filter node to another node in the message flow.
 - Connected both the True and False terminals of a Filter node to another node in the message flow.

Your integration server is not reading messages from the input queues: Procedure

- **Scenario:** Your integration server has started, but is not reading messages from the specified input queues.
- **Explanation:** A started integration server might not read messages from the input queues of the message flows because previous errors might have left the queue manager in an inconsistent state.
- **Solution:** Complete the following steps:
 1. Stop the broker.
 2. Stop the WebSphere MQ listener.
 3. Stop the WebSphere MQ channel initiator.
 4. Stop the WebSphere MQ queue manager.

5. Restart the WebSphere MQ queue manager.
6. Restart the WebSphere MQ channel initiator.
7. Restart the WebSphere MQ listener.
8. Restart the broker.

The integration server ends while processing messages:

About this task

Procedure

- **Scenario:** While processing a series of messages, the integration server (DataFlowEngine) process size grows steadily without levelling off. This situation might cause the DataFlowEngine process to end if it cannot allocate more memory, and restart. The error message BIP2106 might be logged to indicate the out of memory condition.

In addition, if you are using DB2 on distributed systems, you might get the message:

SQL0954C Not enough storage is available in the application heap to process the statement.

z/OS On z/OS, an SQLSTATE of HY014 might be returned with an SQL code of -99999, indicating that the DataFlowEngine process has reached the DB2 z/OS process limit of 254 prepared SQL statement handles.

- **Explanation:** When a database call is made from within a message flow node, the flow constructs the appropriate SQL, which is sent using ODBC to the database manager. As part of this process, the SQL statement is prepared using the SQLPrepare function, and a statement handle is acquired so that the SQL statement can be executed.

For performance reasons, after the statement is prepared, the statement and handle are saved in a cache to reduce the number of calls to the SQLPrepare function. If the statement is already in the cache, the statement handle is returned so that it can be re-executed with newly bound parameters.

The statement string is used to perform the cache lookup. By using hardcoded SQL strings that differ slightly for each message, the statement is not found in the cache, and an SQLPrepare function is always performed (and a new ODBC cursor is opened). When using PASSTHRU statements, use parameter markers so that the same SQL prepared statement can be used for each message processed, with the parameters being bound at run time. This approach is more efficient in terms of database resources and, for statements that are executed repeatedly, it is faster.

However, it is not always possible to use parameter markers, or you might want to dynamically build the SQL statement strings at run time. This situation potentially leads to many unique SQL statements being cached. The cache itself does not grow that large, because these statements themselves are generally not big, but many small memory allocations can lead to memory fragmentation.

- **Solution:** If you encounter these types of situations, disable the caching of prepared statements by setting the MQSI_EMPTY_DB_CACHE environment variable to an arbitrary value. When this environment variable has been created, the prepared statements for that message flow are emptied at the end of processing for each message. This action might cause a slight performance degradation because every SQL statement is prepared.

Your integration server hangs, or ends with a core dump:

Procedure

- **Scenario:** While processing a message, an integration server either hangs with high CPU usage, or ends with a core dump. The stack trace from the core dump or abend file is large, showing many calls on the stack. Messages written to the system log might indicate "out of memory" or "bad allocation" conditions. The characteristics of the message flow in this scenario often include a hard-wired loop around some of the nodes.
- **Explanation:** When a message flow thread executes, it requires storage to perform the instructions that are defined by the logic of its connected nodes. This storage comes from the integration server's heap and stack storage. The execution of a message flow is constrained by the stack size, the default value of which differs depending on the operating system.
- **Solution:** If a message flow that is larger than the stack size is required, you can increase the stack size limit and then restart the brokers that are running on the system so that they use the new value. For information on setting the stack size for your operating system, see "System resources for message flow development" on page 517.

Your XSLTransform node is not working after deployment and errors are issued indicating that the style sheet could not be processed:

Procedure

- **Scenario:** Your XSLTransform node is not working after deploying resources, and errors are displayed indicating that the style sheet could not be processed.
- **Solution:**
 - If the broker cannot find the style sheet or XML files that are required, migrate the style sheets or XML files with relative path references.
 - If the contents of a style sheet or XML file are damaged and therefore no longer usable (for example, if a file system failure occurs during a deployment), redeploy the damaged style sheet or XML file.

Output messages are not sent to expected destinations:

Procedure

- **Scenario:** You have developed a message flow that creates a destination list in the LocalEnvironment tree. The list might contain queues for the MQOutput node, labels for a RouteToLabel node, or URLs for an HTTPRequest node. However, the messages are not reaching these destinations, and there are no error messages.
- **Solution:**
 - Check that you have set Compute mode to a value that includes the LocalEnvironment in the output message, for example All. The default setting of Compute mode is Message, and all changes that you make to LocalEnvironment are lost.
 - Check your ESQL statements. The content and structure of LocalEnvironment are not enforced, so the ESQL editor (and content assist) does not provide guidance for field references, and you might have specified one or more of these references incorrectly.

Some example procedures to help you set up destination lists are provided in Populating Destination in the local environment tree. You can use these procedures unchanged, or modify them for your own requirements.

You experience problems when sending a message to an HTTP node's URL:

Procedure

- **Scenario:** Sending a message to an HTTP node's URL causes a timeout, or the message is not sent to the correct message flow.
- **Explanation:** The following rules are true when URL matching is performed:
 - There is one-to-one matching of HTTP requests to HTTPInput nodes. For each HTTP request, only one message flow receives the message. This statement is true even if two message flows are listening on the same URL. Similarly, you cannot predict which MQInput node that is listening on a particular queue will receive a message.
 - Messages are sent to wildcard URLs only if no other URL is matched. Therefore a URL of /* receives all messages that do not match another URL.
 - Changing a URL in an HTTPInput node does not automatically remove the entry from the HTTP listener. For example, if a URL /A is used first, then changed to a URL of /B, the URL of /A is still used to listen on, even though there is no message flow to process the message. This incorrect URL does get removed after the broker has been stopped and restarted twice.
- **Solution:** To find out which URL the broker is currently listening on, look at the file `wspugin6.conf` in the following location:
 - **Linux** **UNIX** On Linux and UNIX: `/var/mqsi/components/broker_name/config`
 - **Windows** On Windows, `%ProgramData%\IBM\MQSI\components\broker_name\config`, where `%PROGRAMDATA%` is the environment variable that defines the system working directory. The default directory depends on the operating system.If problems persist, empty `wspugin6.conf`, restart the broker, and redeploy the message flows.

When using secure HTTP connections, you change a DNS host's destination but the broker is using a cached DNS host definition:

Procedure

- **Scenario:** You are using a broker with secure HTTP connections that use the Java virtual machine (JVM). You have changed a DNS destination, but the broker is using a cached DNS host definition, therefore you have to restart the broker to use the new definition.
- **Explanation:** By default, Java caches the host lookup from DNS, which is not appropriate if you want to look up the host name each time or if you want to cache it for a limited amount of time. This situation occurs only when you use SSL connections. (When using a secure HTTPS connection, the HTTPRequest node uses the SSL protocol, which issues Java calls, whereas a non-SSL protocol uses native calls.)

To avoid this situation, you can empty the cache on the JVM by setting the `networkaddress.cache.ttl` property to zero. This property dictates the caching policy for successful name lookups from the name service. The value is specified as an integer to indicate the number of seconds for which to cache the successful lookup. The default value of this property is -1, which indicates that the successful DNS lookup value is cached indefinitely in the JVM. If you set this property to 0 (zero), the successful DNS lookup is not cached.
- **Solution:** To pick up DNS entry changes without the need to stop and restart the broker and JVM, disable DNS caching. Edit file `$JAVA_HOME/jre/lib/security/java.security`, and set the value of the `networkaddress.cache.ttl` property to 0 (zero).

The TimeoutControl node issues error message BIP4606 or BIP4607 when the timeout request start time that it receives is in the past:

Procedure

- **Scenario:** When a TimeoutControl node receives a timeout request message that contains a start time in the past, it issues error message BIP4606 or BIP4607: The Timeout Control Node '&2' received a timeout request that did not contain a valid timeout start date/time value.
- **Explanation:** The start time in the message can be calculated by adding an interval to the current time. If a delay occurs between the node that calculates the start time and the TimeoutControl node, the start time in the message will have passed by the time it reaches the TimeoutControl node. If the start time is more than approximately five minutes in the past, a warning is issued and the TimeoutControl node rejects the timeout request. If the start time is less than five minutes in the past, the node processes the request as if it were immediate.
- **Solution:** Ensure that the start time in the timeout request message is a time in the future.

You are using a TimeoutControl node with a TimeoutNotification node, with multiple clients running concurrently, and messages appear to be being dropped:

Procedure

- **Scenario:** You are using a TimeoutControl node with a TimeoutNotification node, with multiple clients running concurrently, and messages appear to be being dropped. In the timeout request message, allowOverwrite is set to TRUE.
- **Explanation:** If multiple clients are running concurrently, and allowOverwrite is set to TRUE in the timeout request message, messages can overwrite each other.
- **Solution:** Ensure that different TimeoutNotification nodes that are deployed on the same broker do not share the same unique identifier.

Error message BIP5347 is issued on AIX when you run a message flow that uses a message set:

About this task

Procedure

- **Scenario:** Error message BIP5347 (MtilmbParser2: RM has thrown an unknown exception) is issued on AIX in either of these circumstances:
 - When you are deploying a message set
 - When you are running a message flow that uses a message set
- **Explanation:** BIP5347 is typically caused by a database exception, and it is issued when an integration server tries to load an MRM dictionary for use by a message flow. This process involves two steps:
 1. The integration server retrieves the dictionary and wire format descriptors from the broker data store.
 2. The integration server stores the dictionary in the memory that a message flow would use to process an MRM message.

BIP5347 is typically issued during step 1. This problem can appear to be intermittent; if you restart the integration server, the message is sometimes processed correctly.

BIP5347 might also be caused by the presence of a datetime value constraint in the message set, which causes the error each time the message set is deployed.

- **Solution:** To identify the cause of the error, capture a service level debug trace to confirm that the database exception is occurring.

- If the error is caused by the presence of a datetime value constraint, a message similar to the following message appears in the service level debug trace (the exact message depends on the datetime constraint in the message set):

```
Unable to parse datetime internally, 9, 2001-12-17T09:30:47.0Z,
yyyy-MM-dd'T'HH:mm:ss.SZZZ
```

This error occurs because the MRM element in question has a datetime value that is not compatible with the datetime format string, so the dictionary is rejected. To solve this problem, ensure that the datetime value is compatible with the datetime format string.

**Error message BIP2130 is issued with code page value of -1 or -2:
Procedure**

- **Scenario:** The following error message is issued:

BIP2130: Error converting a character string to or from codepage [code page value]

where [code page value] is either -1 or -2. You have not, however, specified a code page of either -1 or -2 in your message tree. You have, however, used one of the WebSphere MQ constants MQCCSI_EMBEDDED or MQCCSI_INHERIT.

- **Explanation:** The WebSphere MQ constants MQCCSI_EMBEDDED and MQCCSI_INHERIT are resolved when the whole of the message tree is serialized to produce the WebSphere MQ bit stream. This happens when the message is put on the WebSphere MQ transport. Until that time, these values exist in the message tree as either -1 (for MQCCSI_EMBEDDED) or -2 (for MQCCSI_INHERIT). If one or more parts of the message tree are serialized independently, such as with a ResetContentDescriptor node or ESQL ASBITSTREAM function, this error occurs.
- **Solution:** You do not have to set MQCCSI_EMBEDDED or MQCCSI_INHERIT in the message tree's CodedCharSetId field. You can achieve the same result by explicitly setting the required CodedCharSetId to the previous header's CodedCharSetId value. For example, you would need to replace:

```
SET OutputRoot.MQRFH2.(MQRFH2.Field)CodedCharSetId = MQCCSI_INHERIT;
```

with

```
SET OutputRoot.MQRFH2.(MQRFH2.Field)CodedCharSetId = InputRoot.MQMD.CodedCharSetId;
```

where the MQMD folder is the header preceding the MQRFH2 header.

The integration server restarts before an MQGet node has retrieved all messages:

Procedure

- **Scenario:** You have created a message flow that contains an MQGet node. Not all of the messages are retrieved from the queue because the integration server restarts before the node has retrieved all the messages. No abend files are generated.
- **Explanation:** In IBM Integration Bus, processing that involves nested or recursive processing can cause extensive use of the stack. Message flow processing occurs in a loop until the MQGet node has retrieved all the messages from the queue. Each time that processing returns to the MQGet node, the stack size increases.
- **Solution:** Use a PROPAGATE statement. The statement propagates each message through the message flow in a loop, but each time that processing returns to the PROPAGATE statement, the stack is cleared.

Use an ESQL variable (for example, set Environment.Complete to true) in the environment tree to terminate the ESQL loop, stop the propagations, and wait for the next trigger message. If you need to store content from the messages, store it in the environment tree because other trees are deleted when message flow processing returns to the PROPAGATE statement. For more information about how to use this statement, see PROPAGATE statement.

:

Related concepts:



IBM Integration Toolkit

The IBM Integration Toolkit is an integrated development environment and graphical user interface based on the Eclipse platform.



IBM Integration Explorer

The IBM Integration Explorer is a graphical user interface based on the Eclipse platform for administering your integration nodes (brokers).



Message flows overview

A message flow is a sequence of processing steps that run in the broker when an input message is received.

Related tasks:

“Resolving problems when developing message flows” on page 708

Use the advice given here to help you to resolve common problems that can arise when developing message flows.



Developing integration solutions

IBM Integration Bus provides a flexible environment in which you can develop integration solutions to transform, enrich, route, and process your business messages and data. You can integrate client applications that use different protocols and message formats.



Handling errors in message flows

The broker provides basic error handling for all your message flows. If basic processing is not sufficient, and you want to take specific action in response to certain error conditions and situations, you can enhance your message flows to provide your own error handling.

“Using trace” on page 845

You can use different types of trace to help you with problem determination and troubleshooting.



Accessing the Properties tree

The Properties tree has its own correlation name, Properties, and you must use this in all ESQL statements that refer to or set the content of this tree.



Creating destination lists

Create a list of destinations to indicate where a message is sent.

Related reference:



Message flows

Use the reference information in this section to develop your message flows and related resources.



Built-in nodes

IBM Integration Bus supplies built-in nodes that you can use to define your

message flows.



MQInput node

Use the MQInput node to receive messages from clients that connect to the broker by using the WebSphere MQ Enterprise Transport, and that use the MQI and AMI application programming interfaces.



MQOutput node

Use the MQOutput node to send messages to clients that connect to the broker using the WebSphere MQ Enterprise Transport and that use the MQI and AMI application programming interfaces.

Related information:



WebSphere MQ Version 7 product documentation

Resolving problems when you use IMS nodes

Advice for dealing with common problems that can arise when you develop message flows that contain IMS nodes.

Before you begin

Before you start:

- Read about IMS in IBM Information Management System (IMS).
- Ensure that you have set up the broker runtime environment correctly, as described in Preparing the environment for IMS nodes.

About this task

If you experience problems when you use IMS nodes in message flows, follow the instructions for the following scenarios to diagnose and solve the problem.

- “How can I tell if my broker is connected to IMS Connect?”
- “How can I discover the correct settings for Hostname, Portnumber, and DataStoreName?” on page 735
- “What should I do when my transaction times out?” on page 736
- “How many physical connections should I expect in IMS Connect?” on page 736

How can I tell if my broker is connected to IMS Connect?:

Procedure

- **Scenario:** You need to check if your broker is connected to IMS Connect.
- **Explanation:** You can use SDSF on z/OS to issue a command to see which ports have clients connected to them.
- **Solution:** Using SDSF, enter the following **QUERY MEMBER** command, where *IM0ACONN* is the name of your IMS Connect job:

```
/F IM0ACONN,QRY MEMBER TYPE(IMSCON)
```

The output is in the following format:

```

/F IM0ACONN,QRY MEMBER TYPE(IMSCON)
HWSC00011 HWS ID=IM0ACONN RACF=Y PWDMC=N
HWSC00011 MAXSOC=50 TIMEOUT=0
HWSC00011 RRS=N STATUS=REGISTERED
HWSC00011 VERSION=V10 IP-ADDRESS=009.017.252.024
HWSC00011 SUPER MEMBER NAME=
HWSC00011 ADAPTER=N
HWSC00011 DATASTORE=IM0A STATUS=ACTIVE
HWSC00011 GROUP=IM0AGRNM MEMBER=IM0ACONN
HWSC00011 TARGET MEMBER=IM0A
HWSC00011 DEFAULT REROUTE NAME=HWSEDEF
HWSC00011 RACF APPL NAME=
HWSC00011 OTMA ACEE AGING VALUE=2147483647
HWSC00011 OTMA ACK TIMEOUT VALUE=120
HWSC00011 OTMA MAX INPUT MESSAGE=5000
HWSC00011 NO ACTIVE IMSPLEX
HWSC00011 PORT=1000 STATUS=ACTIVE
HWSC00011 CLIENTID USERID TRancode STATUS SECOND CLNTPORT IP-ADDRESS
HWSC00011 HWSEHYMO JDOE IVTNO RECV 21 1109 009.017.137.11
HWSC00011 TOTAL CLIENTS=1 RECV=1 CONN=0 XMIT=0 OTHER=0
```


This example shows that one client is connected on TCP port 1109, and that the client is connecting from the IP address 9.17.137.11.

You can run netstat on that system to find out in which process that client is running.

Use the following IMS command to view the TPIPEs that are created for those connections:

```
/DISPLAY TMEMBER IMSConnect_Name TPIPE ALL
```

- For transactions that have a commit mode of 1, the TPIPE name is the port number that is used for that interaction (for example, 1080 in the previous example).
- For transactions that have a commit mode of 0, the TPIPE name is the same as the client ID (for example, HWSEHYMO in the previous example). The client ID is generated automatically in the IMSRequest node.

How can I discover the correct settings for Hostname, Portnumber, and DataStoreName?:

Procedure

- **Scenario:** Your broker fails to connect to IMS and you want to verify your settings.
- **Explanation:** You can use SDSF on z/OS to issue a command to see members of the XCF group to which your IMS control region belongs. One of these members should be IMS Connect. You can then run a command against IMS Connect to discover these properties.
- **Solution:** Use SDSF to enter the following command:

```
/xx/display OTMA
```

where *xx* is the reply ID for your IMS control region job.

For example, if you see *26 DFS996I *IMS READY* IM0A in S.log, run the command /26/DISPLAY OTMA.

The output from that command shows the name of the IMS Connect that is in the same XCF group as this IMS control region:

```
DFS000I  GROUP/MEMBER      XCF-STATUS  USER-STATUS  SECURITY  TIBINPT  SMEM  IM0A
DFS000I                DRUEXIT  T/O                IM0A
DFS000I  IM0AGRNM                IM0A
DFS000I  -IM0A             ACTIVE      SERVER        CHECK                IM0A
DFS000I  -IM0A             N/A         0                IM0A
DFS000I  -IM0ACONN          ACTIVE      ACCEPT TRAFFIC CHECK    05000  IM0A
DFS000I  *08350/175112*      IM0A
```

Use SDSF to enter the following **QUERY MEMBER** command, where *IM0ACONN* is the name of your IMS Connect job that was reported by the previous command:

```
/F IM0ACONN,QRY MEMBER TYPE(IMSCON)
```

The output is in the following format:

```
/F IM0ACONN,QRY MEMBER TYPE(IMSCON)
HWSC0001I  HWS ID=IM0ACONN RACF=Y  PSWDMC=N
HWSC0001I  MAXSOC=50  TIMEOUT=0
HWSC0001I  RRS=N  STATUS=REGISTERED
HWSC0001I  VERSION=V10  IP-ADDRESS=009.017.252.024
HWSC0001I  SUPER MEMBER NAME=
HWSC0001I  ADAPTER=N
HWSC0001I  DATASTORE=IM0A  STATUS=ACTIVE
HWSC0001I  GROUP=IM0AGRNM  MEMBER=IM0ACONN
HWSC0001I  TARGET MEMBER=IM0A
HWSC0001I  DEFAULT REROUTE NAME=HWS£DEF
```

```

HWSC0001I      RACF APPL NAME=
HWSC0001I      OTMA ACEE AGING VALUE=2147483647
HWSC0001I      OTMA ACK TIMEOUT VALUE=120
HWSC0001I      OTMA MAX INPUT MESSAGE=5000
HWSC0001I      NO ACTIVE IMSPLEX
HWSC0001I      PORT=1080      STATUS=ACTIVE
HWSC0001I      CLIENTID USERID  TRANCOD STATUS      SECOND CLNTPORT IP-ADDRESS
HWSC0001I      HWSEHYMO JDOE   IVTNO  RECV          21      1109 009.017.137.11
HWSC0001I      TOTAL CLIENTS=1 RECV=1 CONN=0 XMIT=0 OTHER=0

```

What should I do when my transaction times out?:

Procedure

- **Scenario:** The transaction times out and the message is sent to the Timeout terminal, or an exception is issued.
- **Explanation:** Your transaction is taking longer than the values that are set for the execution or socket timeouts, therefore the node stops waiting for a response, and issues an exception or sends the message to the Timeout terminal.
If the transaction subsequently completes successfully, the result depends on the commit mode that is set on the IMSRequest node:
 - If the Commit mode property is set to 0: COMMIT_THEN_SEND, the unit of work is committed and the response is discarded.
 - If the Commit mode property is set to 1: SEND_THEN_COMMIT, the response is not sent and the unit of work is rolled back.
- **Solution:** Increase the execution or socket timeout values to give enough time for the transaction to complete.
 - Configure the execution timeout by using the Timeout waiting for a transaction to be executed property on the IMSRequest node.
 - Configure the socket timeout on the configurable service.

How many physical connections should I expect in IMS Connect?:

Procedure

- **Scenario:** You need to set the values for the number of connections that are required by clients that are connecting to IMS Connect.
- **Explanation:** The number of physical connections that can be opened in IMS Connect is limited. The limit depends on the MAXSOC and MAXFILEPROC settings.
 - The MAXSOC setting in IMS Connect determines the number of sockets that can be opened in IMS, which is the number of ports on which IMS listens for connections, plus the number of physical connections.
 - MAXFILEPROC is a UNIX System Services (USS) setting, which must be greater than or equal to MAXSOC, otherwise IMS reaches this limit before it reaches its own MAXSOC limit.

If the IMS Connect process is granted superuser authority in USS, it sets MAXFILEPROC automatically.

If the MAXSOC value is reached, IMS Connect issues warning message HWSS0771W, and refuses new requests for connections from clients. This behavior continues until the number of open sockets is below the limit (for example, after some clients have disconnected).

If the MAXFILEPROC value is reached, USS issues information message BPXI040I.

- **Solution:** When you set values for MAXSOC and MAXFILEPROC, consider how many clients are likely to connect concurrently to IMS Connect, and how many connections those clients will require.

IBM Integration Bus acts as a client to IMS Connect, and opens connections to IMS Connect. Therefore, find out how many connections are required by IBM Integration Bus by gathering the following information:

- The number of message flows with IMS nodes that are deployed

- For those message flows, the values of the Additional instances property

The maximum number of connections required for each broker is determined by the number of threads that can be running concurrently in the IMS nodes. In the following example, three message flows with IMS nodes exist:

- Message flow A has 0 additional instances, therefore one thread is running.

- Message flow B has 3 additional instances, therefore four threads can be running concurrently.

- Message flow C has 4 additional instances, therefore five threads can be running concurrently.

In this example, the maximum number of connections required by the broker is 10 (1+4+5). If you have four other similar brokers, all connecting to the same instance of IMS Connect, which has five ports configured, you would set the maximum number of sockets (MAXSOC) to at least 55 (the maximum number of connections for five brokers plus the number of ports: 10x5+5).

Related concepts:

 IBM Information Management System (IMS)

IMS is a message-based transaction manager and hierarchical-database manager for z/OS. External applications can use transactions to interact with applications that run inside IMS.

 Execution and threading models in a message flow

The execution model is the system used to start message flows which process messages through a series of nodes.

Related tasks:

 Changing connection information for the IMSRequest node

You can create a configurable service that the IMSRequest node or message flow refers to at run time for connection information, instead of defining the connection properties on the node or the message flow. The advantage being that you can change the host name, performance, and security values without needing to redeploy your message flow.

Related reference:

 IMSRequest node

Use the IMSRequest node to send a request to run a transaction on a local or remote IMS system, and wait for a response. IMS Connect must be configured and running on the IMS system.

 Configurable services properties

The supplied configurable services, and the configurable services that you create, are defined by their names and properties. You can use the supplied services.

 Configurable message flow properties

When you add a message flow to a broker archive (BAR) file in preparation for deploying it to a broker, you can set additional properties that influence its run time operation. These properties are available for review and update when you select the **Manage and Configure** tab for the broker archive file.



Issuing commands to the z/OS console

You operate the broker using the z/OS START, STOP, and MODIFY commands.

Resolving mapping and message reference problems when developing message flows

Advice for dealing with some common mapping and message reference problems that can arise when developing message flows:

Resources that are referenced by the mapping file cannot be resolved:

Procedure

- **Scenario:** You have imported some message flows into the IBM Integration Toolkit that contain mappings. An error is issued, indicating that the resources that are referenced by the mapping file cannot be resolved.
- **Explanation:** Mappings can use resources that exist in other projects. For example, a mapping reference to a message set might exist in a different project. If the reference cannot be resolved, it probably means that the reference to the other project has been lost.
- **Solution:** Create a new project reference. For more information, see Adding and removing library references.

Errors are issued when you import table schemas into the Graphical Data

Mapping editor:

Procedure

- **Scenario:** When you try to import and add table schemas in the Graphical Data Mapping editor, you encounter errors like:
`/flow2/schema1/SAMPLE.conxmi cannot be loaded.`
The following error was reported: `schema1/SAMPLE.conxmi`
- **Explanation:** This error usually means that you have the same database files under the same broker schema name in another project. The relative paths are the same, so the Graphical Data Mapping editor cannot resolve this ambiguity and does not know which table to add.
- **Solution:** There are two courses of action:
 - If the table file already exists in the workspace, and this is what you want to use for mapping, reuse the file by clicking the **Add database table schemas from workspace** option in the Add Database Table Schemas dialog box.
 - If you want a different copy of the tables, rename the broker schema.

Warnings or errors are issued for message references:

Procedure

- **Scenario:** Warnings or errors are issued for message references, yet you are certain that your references are correct.
- **Explanation:** This is never the case with messages that are using the XML parser. For these message references, direct validation is not performed because the references could be used for generic XML.
There is an ESQL editor preference that allows you to choose to ignore message reference mismatches, or to have them be reported as a warning or an error. By default, this type of problem is reported as a warning, so that you can still deploy the message flow.
- **Solution:** To use the validation feature, ensure that you have set up a project reference from the project that contains the ESQL to the project that contains the message set. For more information, see Adding and removing library references.
If you are using reference in a subroutine, take the following steps:

1. Create a reference to the tree and the parser in the module's main procedure.
2. Associate the reference to the correlation name, for example InputRoot or Root. Alternatively, create the OutputRoot.*parser* node, where *parser* is the name of the parser that you want to use.
3. Pass the reference as a parameter to an ESQL subroutine that identifies the XSD type of the reference.

Results

This practice is beneficial because the passed reference supports content assistance and validation for ESQL. The message type content properties open, or open defined are not used in validation, and the assumption is that this property is closed.

A \$db:select out of scope error is generated when you map from a database source:

Procedure

- **Scenario:** You have specified a database as the data source and when you save the map file, there is an error saying \$db:select out of scope
- **Explanation:** A \$db:select expression must be within the scope of the \$db:select entry in the Map Script column of the Spreadsheet pane, meaning that it must be a descendant of the select statement. If a \$db:select expression is out of scope the Graphical Data Mapping editor moves the \$db:select entry to a position where the \$db:select expression is in scope. The \$db:select expression can remain out of scope if it is positioned above the \$db:select entry in the Map Script column of the Spreadsheet pane.
- **Solution:** Delete the out of scope \$db:select expression or move the \$db:select entry in the Map Script column. You can drag the element out of the 'for' row, and then drag the \$db:select entry in the Map Script column higher in the message, above the out of scope \$db:select expression. Ensure that the out of scope \$db:select expression is now a descendant of the \$db:select entry.

A \$db:proc out of scope error is generated when you map from a database stored procedure:

Procedure

- **Scenario:** You have specified a stored procedure as the source and when you save the map file, there is an error saying \$db:proc out of scope
- **Explanation:** A \$db:proc expression must be within the scope of the \$db:proc entry in the Map Script column of the Spreadsheet pane, meaning that it must be a descendant of the stored procedure statement. If a \$db:proc expression is out of scope the Graphical Data Mapping editor moves the \$db:proc entry to a position where the \$db:proc expression is in scope. The \$db:proc expression can remain out of scope if it is positioned above the \$db:proc entry in the Map Script column of the Spreadsheet pane.
- **Solution:** Delete the out of scope \$db:proc expression or move the \$db:proc entry in the Map Script column. You can drag the \$db:proc entry in the Map Script column higher in the message, above the out of scope \$db:proc expression. Ensure that the out of scope \$db:proc expression is now a descendant of the \$db:proc entry.

A \$db:func out of scope error is generated when you map from a database user-defined function:

Procedure

- **Scenario:** You have specified a user-defined function as the source and when you save the map file, there is an error saying \$db:func out of scope
- **Explanation:** A \$db:func expression must be within the scope of the \$db:func entry in the Map Script column of the Spreadsheet pane, meaning that it must be a descendant of the user defined function statement. If a \$db:func expression is out of scope the Graphical Data Mapping editor moves the \$db:func entry to a position where the \$db:func expression is in scope. The \$db:func expression can remain out of scope if it is positioned above the \$db:func entry in the Map Script column of the Spreadsheet pane.
- **Solution:** Delete the out of scope \$db:func expression or move the \$db:func entry in the Map Script column. You can drag the \$db:func entry in the Map Script column higher in the message, above the out of scope \$db:func expression. Ensure that the out of scope \$db:func expression is now a descendant of the \$db:func entry.

Target is not referencing a valid variable warning when you set the value of a target:

Procedure

- **Scenario:** You have set the value for a target to a variable, such as a WebSphere MQ constant, and when you save the map file the warning The target "\$target" is not referencing a valid variable is generated.
- **Explanation:** The variable that you have referenced is not recognized. For example, you might have entered an expression of the form \$mq: followed by a WebSphere MQ constant, but the constant is not recognized. This might be because the variable has been entered incorrectly or it is not supported. Alternatively, you might be referencing a new variable or constant that can be resolved only at run time. If this is the case you can ignore the warning.
- **Solution:** Try one of the following to solve the problem:
 - Check that the variable has been entered correctly.
 - If you are using WebSphere MQ constants, use **Edit > Content Assist** to select from the list of available WebSphere MQ constants.

There are missing or unexpected targets in a message map:

Procedure

- **Scenario:** In your message map, warning messages are displayed that indicate a target element is missing, or a target element is at an unexpected location. As a result the output message generated by the message map might be incorrect.
- **Explanation:** If target elements are missing from the Spreadsheet pane when you edit and save the message map, a warning is displayed that a target is missing. This situation can occur if you use the Insert Children wizard and do not select all the required elements, or if you create mappings using the drag-and-drop method and do not create mappings for all the required fields. If target elements in the Spreadsheet pane are in an unexpected order, a warning that the element is unexpected at that location is displayed. This situation can occur if you drag elements to new locations in the Spreadsheet pane.
- **Solution:** To solve the problem:
 - Use **Insert Children** on the parent element to add any missing target elements to the Spreadsheet pane.
 - Drag any target elements that are in an unexpected location to the correct location. Use the message tree in the Target pane as a guide to the expected structure of the output message.

Error message BIP6118 is issued: The remaining bitstream is too small contain the indicated structure.:

Procedure

- **Scenario:** You have used an unsupported message domain for a target message in your message map.
- **Explanation:** The message domain that is associated with a target message is determined by the **Message Domain** property of your message set. Mapping nodes generate a target message that matches the message domain of the message set. Using a message domain that is not supported by the message mapper can result in an output message with a structure that is not valid for the chosen parser.
- **Solution:** To solve the problem, change the target message domain for your message set.

Error message BIP4680 is issued: Unsupported message domain encountered in mapping node.:

Procedure

- **Scenario:** You have used an unsupported message domain for a target message in your message map, for example BLOB.
- **Explanation:** The message domain that is associated with a target message is determined by the **Message Domain** property of your message set. Mapping nodes generate a target message that matches the message domain of the message set. Using a message domain that is not supported by the message mapper can result in an output message with a structure that is not valid for the chosen parser.
- **Solution:** To solve the problem, change the target message domain for your message set.

Related concepts:

 [Message flows overview](#)

A message flow is a sequence of processing steps that run in the broker when an input message is received.

Related tasks:

“Resolving problems when developing message flows” on page 708

Use the advice given here to help you to resolve common problems that can arise when developing message flows.

 [Developing integration solutions](#)


IBM Integration Bus provides a flexible environment in which you can develop integration solutions to transform, enrich, route, and process your business messages and data. You can integrate client applications that use different protocols and message formats.

 [Handling errors in message flows](#)

The broker provides basic error handling for all your message flows. If basic processing is not sufficient, and you want to take specific action in response to certain error conditions and situations, you can enhance your message flows to provide your own error handling.

“Using trace” on page 845

You can use different types of trace to help you with problem determination and troubleshooting.

 [Accessing the Properties tree](#)

The Properties tree has its own correlation name, Properties, and you must use this

in all ESQL statements that refer to or set the content of this tree.



Creating destination lists

Create a list of destinations to indicate where a message is sent.

Related reference:



Message flows

Use the reference information in this section to develop your message flows and related resources.

Resolving trace problems when developing message flows

Follow this advice to deal with some common trace problems that can arise when you develop message flows.

About this task

- “You cannot determine which node is being referenced in your trace file”
- “You cannot see any alerts when you change user trace”
- “Data that the Trace node sends to the syslog on UNIX is truncated” on page 743

You cannot determine which node is being referenced in your trace file:

Procedure

- **Scenario:** You have generated a service trace file for your message flow, to trace the path of a message. However, you cannot determine which node is being referenced in the trace file.

- **Explanation:** In the trace file you might see text such as:

```
Video_Test#FCMComposite_1_1 ComIbmMQInputNode , Video_Test.VIDEO_XML_IN
```

The elements in the text have the following meanings:

Video_Test

is the name of the message flow

FCMComposite_1_1

is the internal name for the node

ComIbmMQInputNode

is the type of node

VIDEO_XML_IN

is the node label, and is the name you see in your flow

The number at the end of the internal name is incremented; for example, FCMComposite_1_4 would be the fourth node you added to your flow. In the example, this section of the trace is referring to the first node in the message flow.

You cannot see any alerts when you change user trace:

Procedure

- **Scenario:** You cannot see any alerts for an integration server or message flow in the Alerts viewer when you use the **mqsichangetrace** command to change the user trace setting.
- **Explanation:** No alert is generated when an integration server runs user trace at normal or debug level. Alerts are generated only for message flow trace. For message flow trace, the IBM Integration Toolkit is not notified of trace changes that are initiated by the **mqsichangetrace** command.

- **Solution:** To see the alert, refresh the message flow, or disconnect, then reconnect to the domain.

**Data that the Trace node sends to the syslog on UNIX is truncated:
Procedure**

- **Scenario:** You are using a Trace node on UNIX and have set the Destination property to Local Error Log. You send a message to the Trace node that consists of multiple lines, but the message that appears in the syslog is truncated at the end of the first line.
- **Explanation:** On UNIX, syslog entries are restricted in length and messages that are sent to the syslog are truncated by the new line character.
- **Solution:** To record a large amount of data in a log on UNIX, set the Destination property on the Trace node to File or User Trace instead of Local Error Log.

Related concepts:

 Message flows overview

A message flow is a sequence of processing steps that run in the broker when an input message is received.

Related tasks:

“Resolving problems when developing message flows” on page 708
Use the advice given here to help you to resolve common problems that can arise when developing message flows.

 Developing integration solutions

IBM Integration Bus provides a flexible environment in which you can develop integration solutions to transform, enrich, route, and process your business messages and data. You can integrate client applications that use different protocols and message formats.

 Handling errors in message flows

The broker provides basic error handling for all your message flows. If basic processing is not sufficient, and you want to take specific action in response to certain error conditions and situations, you can enhance your message flows to provide your own error handling.

“Using trace” on page 845

You can use different types of trace to help you with problem determination and troubleshooting.

Related reference:

 Message flows

Use the reference information in this section to develop your message flows and related resources.

Resolving problems when developing message flows with WebSphere Adapters nodes

Advice for dealing with common problems that can arise when you develop message flows that contain WebSphere Adapters nodes.

About this task

WebSphere Adapters nodes

- “Error messages BIP3414 and BIP3450 are issued when you deploy a WebSphere Adapters input node” on page 744

- “Error messages are issued when classes cannot be found, or when problems occur with Java initialization”
- “The WebSphere Adapters are not visible when you run ITLM” on page 745
- “A message flow with an SAPRequest, SiebelRequest, or PeopleSoftRequest node has deployed successfully, but message BIP3540 is issued indicating that connection failed” on page 745

SAP nodes

- “You have deployed an SAP inbound adapter but do not receive expected messages” on page 746
- “You have imported an existing project into your workspace, but messages are issued when you try to build SAP message sets” on page 746
- “An error is issued when you use the message set that is generated by the Adapter Connection wizard” on page 746
- “When you run the SAP samples on Linux or UNIX, IBM Integration Bus does not connect to the SAP gateway” on page 746
- “SAP inbound messages (ALE and BAPI) appear to be missing” on page 747
- “When two ALE inbound modules use the same RFC program ID with SAP JCo version 3.0.2, NullPointerException exceptions are logged in the JCo trace and the Adapter does not receive IDocs” on page 747

Siebel nodes

- “You are using a SiebelInput node with the delivery type set to unordered, and error message BIP3450 is issued with a NullPointerException” on page 747

JD Edwards nodes

- “Error message BIP3450 is issued when you include a JDEdwardsRequest node in a message flow and try to connect to a JD Edwards EnterpriseOne server” on page 748

Error messages BIP3414 and BIP3450 are issued when you deploy a WebSphere Adapters input node: Procedure

- **Scenario:** When you deploy a message flow that contains a SiebelInput node, error message BIP3414 is issued.
- **Explanation:** The error messages explain that the SiebelInput node could not register with the adapter component to receive events. This problem can be caused when the broker does not know where to find the client libraries for the Siebel Enterprise Information Service (EIS). You might also encounter this problem if you are using the WebSphere Adapter for Siebel on an unsupported operating system.
- **Solution:** Use the `mqsireportproperties` and `mqsichangeproperties` commands to configure the broker with the location of the Siebel client libraries, as described in Preparing the environment for WebSphere Adapters nodes.

Error messages are issued when classes cannot be found, or when problems occur with Java initialization:

Procedure

- **Scenario:** You are deploying WebSphere Adapters, and error messages are issued that indicate that classes cannot be found, or problems are occurring with Java initialization. BIP3521 and BIP3522 error messages might also be issued.

- **Explanation:** The SAP, Siebel, and PeopleSoft adapters need client libraries from the manufacturer of the Enterprise Information System (EIS). If these libraries are missing, not installed correctly, or at an incorrect level, errors are issued.
- **Solution:** To solve this problem, complete the following steps.
 1. On the broker that you are using to change the configurable service of the adapter, run the **mqsichangeproperties** command to identify the location of the Java and native libraries.
 2. Ensure that the libraries are installed correctly, are valid for your operating system, and have the correct permission so that the broker can access them.
 3. Ensure that your operating system is supported by IBM Integration Bus and the EIS provider. For details about supported operating systems, visit the IBM Integration Bus Requirements Web site.

The WebSphere Adapters are not visible when you run ITLM: Procedure

- **Scenario:** The adapter is not visible when you run the IBM Tivoli License Manager (ITLM).
- **Explanation:** If you want to use ITLM with the WebSphere Adapters, you must activate the ITLM file for each adapter.
- **Solution:** Follow the instructions in Activating IBM License Metric Tool for WebSphere Adapters.

A message flow with an SAPRequest, SiebelRequest, or PeopleSoftRequest node has deployed successfully, but message BIP3540 is issued indicating that connection failed: Procedure

- **Scenario:** From an SAPRequest, SiebelRequest, or PeopleSoftRequest node, an exception is thrown to indicate that the node is unable to make a connection even though the message flow has deployed successfully. The exception contains message BIP3540 with inserted text that indicates that connection failed. For example, for SAP, the inserted text is:

```
Exception in connecting to SAP:Connect to SAP gateway failed
Connect_PM GWHOST= invalidhost.test.co, GWSERV=sapgw00, ASHOST= invalidhost.test.co,
  SYSNR=00
LOCATION CPIC (TCP/IP) on local host ERROR partner not reached (host
  invalidhost.test.co, service 3300) TIME Mon Dec 01 16:43:52 2008 RELEASE 640
COMPONENT NI (network interface) VERSION 37 RC -10 MODULE nixxi_r.cpp LINE 8719
DETAIL NiPConnect2 SYSTEM CALL SiPeekPendConn ERRNO 10061 ERRNO_TE
```

For PeopleSoft, the inserted text is:

```
001DOWNbea.jolt.ServiceException: Invalid Session
```

- **Explanation:** The connection details are not verified when the message flow is deployed. For request nodes, the connection is made at first use.
- **Solution:** If you have configured a configurable service for this adapter, review the connection properties on that configurable service and review the text in the BIP3540 message to determine if the connection properties are incorrect. If the properties are incorrect, use the **mqsichangeproperties** command to correct them or use the **mqsdeleteconfigurablesevice** command to revert to the properties that are set on the adapter. Reload the integration server or stop and restart the broker.

If no configurable service exists for this adapter, review the connection properties on the adapter. If the properties are incorrect, correct them and redeploy the adapter. Alternatively, use the **mqscreateconfigurablesevice** command to create a new configurable service with the correct properties to override the properties that are set on the adapter.

You have deployed an SAP inbound adapter but do not receive expected messages:

Procedure

- **Scenario:** You have deployed an SAP inbound adapter but do not receive the IDoc messages that you expected to receive.
- **Explanation:** If you have not received IDoc messages from SAP, it is possible that deployment was unsuccessful or the SAP server has not started.
- **Solution:** Check user trace for message BIP3484 occurring at the time of deployment. The adapter component writes diagnostic information to this message, in an insert that begins "CWYAP...". If this message is issued, it explains the cause of the problem.

You have imported an existing project into your workspace, but messages are issued when you try to build SAP message sets:

Procedure

- **Scenario:** You have imported an existing project into your workspace, but when you try to build an SAP message set, you see the message set compile error message BIP0182.
- **Explanation:** This error occurs when you choose the option to "Import existing projects into your workspace" from the Import dialog box. By choosing this option when you import, a link is created from the workspace to the existing projects in an external location and a required file is not available to the workspace. To copy the entire project into your workspace, use the option to import the Project Interchange (PI) file.
- **Solution:** When you import an existing SAP project into your workspace, click **File > Import**, expand the **Other** folder, and click **Project Interchange**. For more information, see Importing resources from previous versions.

An error is issued when you use the message set that is generated by the Adapter Connection wizard:

Procedure

- **Scenario:** You run the Adapter Connection wizard and select an inbound SAP IDoc. You run the wizard again, but this time you select an outbound SAP IDoc. When you use the message set that is generated, the following error is issued:

```
'Selector exception caught from generateEISFunctionname', 'commonj.connector.runtime.SelectorException:
commonj.connector.runtime.SelectorException: For the IDoc type SapYwmspgi01, operation key=YWMSPGIWMS not
found using the application-specific information {Create={MsgType=, MsgCode=, MsgFunction=}} verify appropriate
combination of MsgType, MsgCode, MsgFunction is set in SapYwmspgi01, application-specific information.
```

- **Explanation:** If you run the Adapter Connection wizard for an inbound SAP IDoc, then you run the wizard again for an outbound SAP IDoc, the outbound IDoc definition replaces the inbound IDoc definition. Information that is stored in the inbound definition is used to map MsgType, MsgCode, and MsgFunction to a method binding. The outbound definition does not contain these mappings, so processing of the inbound IDoc fails.
- **Solution:** To avoid this error, ensure that inbound and outbound SAP IDocs have different names if they are stored in the same message set.

When you run the SAP samples on Linux or UNIX, IBM Integration Bus does not connect to the SAP gateway:

Procedure

- **Scenario:** When you run the SAP samples on Linux or UNIX, the message flow deploys successfully, but a connection is not established between IBM Integration Bus and the SAP system. You might see the following error message:

```
message: Connect to SAP gateway failed
Connect parameters: TPNAME=SAMPRFC GWHOST=sapdev10 GWSERV=sapgw00
ERROR service 'sapgw00' unknown
```

- **Explanation:** For the SAP Java Connector (SAP JCo) to communicate across the network, you need to configure the TCP/IP service. If you have installed a working SAP GUI on your workstation, the TCP/IP service is configured as part of the installation. If you have not installed an SAP GUI, you must configure the TCP/IP service manually so that the SAP samples run successfully.
- **Solution:** Locate the services file in `/etc/services` and edit it so that it includes the appropriate gateway service IDs and the port numbers for the TCP/IP service in the format `sapgwSID portnumber/tcp`, where `SID` is the SAP system ID. For example:

```
sapgw00 3300/tcp
sapgw01 3301/tcp
sapgw02 3302/tcp
```

and so on.

For more information about TCP/IP configuration, see the Network Integration section of the SAP Service Marketplace.

SAP inbound messages (ALE and BAPI) appear to be missing:

Procedure

- **Scenario:** SAP inbound messages (ALE and BAPI) appear to be missing. You might find that every other message does not reach the broker but no errors are issued.
- **Explanation:** This problem is typically caused when two brokers share the same program ID (SAP RFC Destination). For example, a developer has deployed a message flow, but someone else has used the same broker archive (BAR) file without having changed the program ID.
- **Solution:** Ensure that no other brokers are running with the same program ID. Use SAP transaction SMGW to determine whether other brokers are connected to the SAP system.

When two ALE inbound modules use the same RFC program ID with SAP JCo version 3.0.2, NullPointerException exceptions are logged in the JCo trace and the Adapter does not receive IDocs:

Procedure

- **Scenario:** When two ALE inbound modules use the same RFC program ID with SAP JCo version 3.0.2, NullPointerException exceptions are logged in the JCo trace and the Adapter does not receive IDocs.

The following example shows a typical exception in the JCo trace.

```
JCoDispatcherWorkerThread [16:44:42:140]: [JCoApi] Dispatcher.getNextListener() returns dispatch next call rfc handle(1) for null
JCoDispatcherWorkerThread [16:44:42:140]: [JCoApi] caught Throwable in DispatcherWorker.run() while trying to dispatch a request java.lang.NullPointerException
at java.util.Hashtable.get(Hashtable.java:518)
at com.sap.conn.jco.rt.DefaultServerManagerDispatcherWorker.run(DefaultServerManager.java:268)
at java.lang.Thread.run(Thread.java:735)
```

```
JCoDispatcherWorkerThread [16:44:42:140]: [JCoApi] Dispatcher.getNextListener() returns no calls
```

- **Explanation:** SAP JCo version 3.0.2 does not support the use of two ALE inbound modules with the same RFC program ID.
- **Solution:** To solve this problem, download a hotfix from SAP; the SAP ticket reference number is 584247/2009.

You are using a SiebellInput node with the delivery type set to unordered, and error message BIP3450 is issued with a NullPointerException:

Procedure

- **Scenario:** You are using a SiebelInput node, you have set the delivery type to unordered in the Adapter Connection wizard, and the minimum number of connections is 1 or less. The following exception is shown in user trace: RecoverableException BIP3450E: An adapter error occurred during the processing of a message. The adapter error message is `java.lang.NullPointerException`.
- **Explanation:** When using unordered events, the minimum connections (MinimumConnections) and maximum connections (MaximumConnections) properties must be greater than 1 for event delivery to be successful.
- **Solution:** Set the MinimumConnections and MaximumConnections properties on the Adapter Connection wizard to values greater than 1. For example, set the minimum number of connections to 2 and the maximum number of connections to 4.

Error message BIP3450 is issued when you include a JDEdwardsRequest node in a message flow and try to connect to a JD Edwards EnterpriseOne server:

Procedure

- **Scenario:** You have created a message flow that contains a JDEdwardsRequest node, but error message BIP3450 is issued, indicating that the node is unable to connect to the JD Edwards EnterpriseOne server.
- **Explanation:** This error message indicates that the JDEdwardsRequest node is trying to connect to the JD Edwards EnterpriseOne server but is unable to do so. This error can be caused by the following situations.
 - The JD Edwards EnterpriseOne server is not running.
 - The JD Edwards adapter has been configured with incorrect connection details; for example, the name of the JD Edwards EnterpriseOne Environment to which to connect.
 - The JDBC drivers that are required to connect to the JD Edwards EnterpriseOne server are missing from the class path. The following table lists the required JDBC driver files for each database.

Database	JDBC driver files	Implementation class
Oracle	tsnames.ora classes12.zip	oracle.jdbc.driver.OracleDriver
SQLServer	sqljdbc.jar	com.ibm.microsoft.sqlserver.jdbc.SQLServerDriver
AS/400®	jt400.jar	com.ibm.as400.access.AS400JDBCdriver
DB2 Type-2 (JDK 1.4/1.5)	db2java.zip	com.ibm.db2.jdbc.app.DB2Driver
DB2 Type-4 (JDK 1.4/1.5)	db2jcc.jar db2jcc_license_cu.jar	com.ibm.db2.jcc.DB2Driver
DB2 Type-4 (JDK 1.6)	db2jcc4.jar	com.ibm.db2.jcc.DB2Driver

- **Solution:** Ensure that the following conditions have been met.
 - The JD Edwards EnterpriseOne server is running.
 - The JD Edwards adapter has been configured with the correct connection details.
 - The drivers that are required to connect to the JD Edwards EnterpriseOne server are in the class path.

Related concepts:





WebSphere Adapters nodes


A WebSphere Adapters node is a message flow node that is used to communicate with Enterprise Information Systems (EIS), such as SAP, Siebel, JD Edwards, and


PeopleSoft.


Related tasks:

 Developing message flows that use WebSphere Adapters
For information about how to develop message flows that use WebSphere Adapters, see the following topics.


 Preparing the environment for WebSphere Adapters nodes
Before you can use the WebSphere Adapters nodes, you must set up the broker runtime environment so that you can access the Enterprise Information System (EIS).


 Deploying a message flow that uses WebSphere Adapters
Deploy the resources that are generated when you run the Adapter Connection wizard by adding them to a broker archive (BAR) file.


 Changing connection details for SAP adapters
SAP nodes can get SAP connection details from either the adapter component or a configurable service. By using configurable services, you can change the connection details for adapters without the need to redeploy the adapters. To pick up new values when a configurable service is created or modified, you must reload the broker or integration server to which the adapter was deployed, by using the **mqsistop** and **mqsistart** commands, or the **mqsireload** command.

 Changing connection details for Siebel adapters
Siebel nodes can get Siebel connection details from either the adapter component or a configurable service. By using configurable services, you can change the connection details for adapters without the need to redeploy the adapters. To pick up new values when a configurable service is created or modified, you must reload the broker or integration server to which the adapter was deployed, by using the **mqsistop** and **mqsistart** commands, or the **mqsireload** command.

Related reference:

 **mqsichangeproperties** command
Use the **mqsichangeproperties** command to modify broker properties and properties of broker resources.

 **mqsicreateconfigurable-service** command
Use the **mqsicreateconfigurable-service** command to create an object name for a broker external resource.

 **mqsdeleteconfigurable-service** command
Use the **mqsdeleteconfigurable-service** command to delete a configurable service, such as a JMS provider, JDBC provider, or FTP server, that you have created by using the **mqsicreateconfigurable-service** command.

IBM Integration Bus Requirements

Resolving other problems when developing message flows

Use the advice given here to deal with problems that can arise when developing message flows, and that are not covered in the specific categories listed in "Resolving problems when developing message flows"

About this task

- "The values of your promoted properties are lost after editing" on page 750

- “The Message Flow editor experiences problems when opening a message flow, and opens in error mode”
- “A message flow has subflows with the same user-defined property set to different values, but only one value is set at run time”
- “You want to move resources to a new broker schema that you have created but it is not visible in the Application Development view”
- “A selector exception is raised when you use WebSphere Adapters” on page 751

The values of your promoted properties are lost after editing:

Procedure

- **Scenario:** You edited a message flow using the Message Flow editor, and the values of your promoted properties are lost.
- **Explanation:** The values of promoted properties for nodes with more than a single subflow definition (that is, two identically named subflows in the same project reference path) are lost if the flow is edited and saved.
- **Solution:** To avoid this problem, ensure that each subflow in your project has a different name.

The Message Flow editor experiences problems when opening a message flow, and opens in error mode:

Procedure

- **Scenario:** You attempt to open an existing message flow in the Message Flow editor and it opens in read-only error mode, displaying a list of parsing or loading errors. The message flow is not open and a message is displayed indicating that the message flow file is not valid.
- **Explanation:** The message flow file is unreadable or is corrupted, and the Message Flow editor cannot render the model graphically.
- **Solution:** Contact IBM Customer Support for assistance with the corrupted file.

A message flow has subflows with the same user-defined property set to different values, but only one value is set at run time:

Procedure


- **Scenario:** You have a message flow that contains identical subflows. Each subflow has the same user-defined property (UDP), but with different values. At run time, only one of the values is set.
- **Explanation:** A UDP has global scope and is not specific to a particular subflow. If you reuse a subflow in a message flow, and those subflows have identical UDPs, you cannot set the UDPs to different values.
- **Solution:** If you need to set a different value for each subflow, use a different UDP for each subflow.

You want to move resources to a new broker schema that you have created but it is not visible in the Application Development view:

Procedure

- **Scenario:** You have created a new broker schema and you want to move a resource to it, but the schema is not visible in the Application Development view.
- **Explanation:** If category mode is selected, you cannot see the broker schema in the Application Development view.
- **Solution:** To move resources to a broker schema that you have created, take one of the following steps.



- Click **Hide Categories** () on the Application Development view toolbar. The new broker schema appears in the Application Development view and you can drag resources onto it.
- Right-click a resource, click **Move**, then select the schema that you have created. When you click **OK**, the resource is moved to the selected schema.

**A selector exception is raised when you use WebSphere Adapters:
Procedure**

- **Scenario:** You run the Adapter Connection wizard for an inbound IDOC, then you run the wizard again for an outbound IDOC. When the message set is generated, you see the following error message:

```
'Selector exception caught from generateEISFunctionname' ,
'commonj.connector.runtime.SelectorException: For the IDoc type
SapYwmSPI01, operation key=YWMSPGIWMS not found using the
application-specific information {Create={MsgType=, MsgCode=, MsgFunction=}}
verify appropriate combination of MsgType,MsgCode, MsgFunction is set in
SapYwmSPI01, application-specific information.
--
```

- **Explanation:** When you run the Adapter Connection wizard for an inbound IDOC, then you run the wizard again for an outbound IDOC, the outbound IDOC definition replaces the inbound IDOC definition. Information that is stored in the inbound definition is used to map MsgType, MsgCode, and MsgFunction to a method binding. The outbound definition does not contain these mappings, so processing of the inbound IDOC fails.
- **Solution:** To avoid this error, ensure that inbound and outbound IDOCs have different names if they are stored in the same message set.

Related concepts:

 Message flows overview

A message flow is a sequence of processing steps that run in the broker when an input message is received.


Related tasks:

“Resolving problems when developing message flows” on page 708

Use the advice given here to help you to resolve common problems that can arise when developing message flows.

 Developing integration solutions

IBM Integration Bus provides a flexible environment in which you can develop integration solutions to transform, enrich, route, and process your business messages and data. You can integrate client applications that use different protocols and message formats.

 Developing message flows that use WebSphere Adapters

For information about how to develop message flows that use WebSphere Adapters, see the following topics.

Related reference:

 Message flows

Use the reference information in this section to develop your message flows and related resources.

Resolving problems when deploying message flows or message sets

Use the advice given here to help you to resolve common problems that can arise when you deploy message flows or message sets.

Initial checks

Procedure

1. To debug problems when deploying, check the following logs:

- The administration log
- The local error log (the Windows Event log or the syslog)
- The WebSphere MQ logs

These logs might be on separate computers, and must be used with the IBM Integration Explorer and IBM Integration Toolkit output to ensure that the deployment was successful.

Use the `mqsilist` command to check that the deployment was successful, or look in the Windows Event or Administration Log view.

2. Use this checklist when you have deployment problems:

- Make sure that the mode that your broker is working in is appropriate for your requirements. See Operation modes.
- Make sure that the remote queue manager is running.
- Make sure that channels are running.
- Display the channel status to see if the number of system messages sent increases.
- Check the channel from the remote end.
- Check the queue manager name.
- Determine whether the channel is a cluster channel.

Common problems

About this task

“Resolving problems that occur when preparing to deploy message flows” on page 754

- “An error is issued when you add a message set to a broker archive file” on page 754
- “You cannot drag a broker archive file to a broker” on page 754
- “You cannot deploy a message flow that uses a user-defined message flow” on page 754
- “The compiled message flow file (.cmf) has not been generated” on page 754

“Resolving problems that occur during deployment of message flows” on page 755


- “You receive a warning message about your broker mode” on page 756
- “The message flow deploys on the test system, but not elsewhere” on page 757
- “Error messages about your broker mode are issued when you create an integration server” on page 757
- “Error messages about your broker mode are issued when you deploy” on page 758
- “Error messages about your function level are issued when you deploy” on page 759

- “Error messages are issued when you deploy to z/OS” on page 759
- “Expected serialization of input is not occurring for a shared queue that serves multiple instances of a message flow on z/OS” on page 759
- “You create a configurable service, then deploy a message flow and inbound adapter, but the deployment fails” on page 760
- “Error messages are issued when you deploy” on page 762
- “You get an authority check failure when you deploy to z/OS” on page 761

“Resolving problems that occur after deployment of message flows” on page 766

- “Your XSLTransform node does not work after deployment” on page 766
- “You receive exception ('MQCC_FAILED') reason '2042' ('MQRC_OBJECT_IN_USE')” on page 766

Related concepts:


 Packaging and deployment overview

Deployment is the process of transferring data to an integration server on a broker so that it can take effect in the broker. Message flows and associated resources are packaged in broker archive (BAR) files for deployment.

 Operation modes

The operation mode that you can use for your broker is determined by the license that you purchase.

Related tasks:

 Packaging and deploying

Package resources that you create in the IBM Integration Toolkit, such as message flows, and deploy them to integration servers on brokers.

“Are the Linux and UNIX environment variables set correctly?” on page 652

Use the **mqsiprofile** command to set a command environment.

Related reference:

 **mqsideploy** command

Use the **mqsideploy** command to make a deployment request to the broker.

 **mqsilist** command

Use the **mqsilist** command to list installed brokers and their associated resources.

 **mqsistart** command

Use the **mqsistart** command to start the specified broker if all initial verification tests complete successfully.

 **mqsistop** command

Use the **mqsistop** command to stop the specified component.

 **mqsicreatebroker** command

Use the **mqsicreatebroker** command to create a broker and its associated resources.

 **mqsichangebroker** command

Use the **mqsichangebroker** command to change one or more of the configuration parameters of the broker.

Resolving problems that occur when preparing to deploy message flows

Use the advice given here to help you to resolve common problems that can occur during preparations to deploy message flows or message sets.

An error is issued when you add a message set to a broker archive file:

Procedure

- **Scenario:** An error is issued when you add a message set to a broker archive (BAR) file.
- **Explanation:** After you create a BAR file and add a message set project to it, two files are created in the BAR file: `messageset.user.txt` and `messageset.service.txt`. The `user.txt` file contains user log information, such as warning message BIP0177W, which states that the dictionary that you have created is not compatible with earlier versions.
- **Solution:** Use the information in the `user.txt` file to diagnose the error. The `service.txt` file contains detailed information that is used by the broker, and can be used by the IBM Support Center to diagnose problems.

You cannot drag a broker archive file to a broker:

Procedure

- **Scenario:** You cannot drag a broker archive (BAR) file to a broker.
- **Explanation:** BAR files can be deployed only on an integration server.
- **Solution:** Select an integration server in the deploy dialog.

You cannot deploy a message flow that uses a user-defined message flow:

Procedure

- **Scenario:** You have created a message flow that contains an input node in a user-defined node project. However, you cannot deploy a message flow that uses this node.
- **Explanation:** Validation, compilation, and deployment do not recognize that a user-defined message flow contains an input node.
- **Solution:** To work around the problem, add a dummy input node to the flow that you intend to deploy.

The compiled message flow file (.cmf) has not been generated:

Procedure

- **Scenario:** The compiled message flow file (.cmf) has not been generated. Therefore, it is not added to the broker archive file, and cannot be deployed.

- **Explanation:** When you create files that define message flow resources in the IBM Integration Toolkit, the overall file path length of those files must not exceed 256 characters, because of a Windows file system limitation.

If you have a message flow that includes resource files that have a path length that exceeds 256 characters, the message flow cannot be compiled when you try to add it to a BAR file, and therefore cannot be deployed.

You might also be affected by this restriction when you use the Adapters Connection wizard; names discovered and returned by the EIS can be very long. The Adapters Connection wizard attempts to write these files to the message set on the local file system, but their path exceeds the operating system limit, and the files appear corrupted.

- **Solution:** To ensure that the path length does not exceed 256 characters, use names that are as short as possible for all resources; for example:
 - The installation path

- Project names and broker schema names
- ESQL and mapping file names
- Inbound and outbound adapter files

Related concepts:



Packaging and deployment overview

Deployment is the process of transferring data to an integration server on a broker so that it can take effect in the broker. Message flows and associated resources are packaged in broker archive (BAR) files for deployment.



User-defined extensions overview

A user-defined extension is an optional component that you design and create to extend the functionality of IBM Integration Bus.

Related tasks:

“Resolving problems when deploying message flows or message sets” on page 752
Use the advice given here to help you to resolve common problems that can arise when you deploy message flows or message sets.



Packaging and deploying

Package resources that you create in the IBM Integration Toolkit, such as message flows, and deploy them to integration servers on brokers.

“Are the Linux and UNIX environment variables set correctly?” on page 652
Use the **mqsiprofile** command to set a command environment.

Related reference:



mqsideploy command

Use the **mqsideploy** command to make a deployment request to the broker.

Resolving problems that occur during deployment of message flows

Use the advice given here to help you to resolve problems that can arise during deployment of message flows or message sets.

Procedure

- “You receive a warning message about your broker mode” on page 756
- “The message flow deploys on the test system, but not elsewhere” on page 757
- “Error messages about your broker mode are issued when you create an integration server” on page 757
- “Error messages about your broker mode are issued when you deploy” on page 758
- “Error messages about your function level are issued when you deploy” on page 759
- “Error messages are issued when you deploy to z/OS” on page 759
- “Expected serialization of input is not occurring for a shared queue that serves multiple instances of a message flow on z/OS” on page 759
- “You create a configurable service, then deploy a message flow and inbound adapter, but the deployment fails” on page 760
- “You have created a WebSphere Adapters message flow that uses secondary adapters, and a naming clash has occurred in the secondary adapters or message sets” on page 760
- “You get an authority check failure when you deploy to z/OS” on page 761
- “Deployment fails when you have circular project dependencies” on page 762
- “Error messages are issued when you deploy” on page 762

**You receive a warning message about your broker mode:
About this task**

Message BIP1806

- **Scenario:** Warning message BIP1806 is displayed.
The broker '*Broker_Name*' did not reveal its mode in time;
any reported information might not be up-to-date.
- **Explanation:** Your broker timed out before the command completed. You can set the timeout value on the **-w** parameter of the **mqs imode** command, or accept the default value of 60 seconds; see **mqs imode** command.
- **Solution:** Check that the broker is started: if it is not started, start it by using the **mqsistart** command. If it is started, increase the timeout value and run the command again.

Message BIP1808

- **Scenario:** Warning message BIP1808 is displayed.
Broker '*Broker_Name*' is not at the required software level
to change the mode.
- **Explanation:** Your broker is running in a previous version of the product. You must migrate brokers that you created in a version earlier than V6.1.0.2 before you can change the operation mode.
- **Solution:** Upgrade your broker to at least V6.1.0.2, and run the command again.

Message BIP1821

- **Scenario:** Warning message BIP1821 is displayed.

WARNING: Broker '*Broker_Name*' is in '*Mode_Name*' mode but has 'x' integration servers, which exceeds the allowed maximum for this mode.

- **Explanation:** Your broker is running in a mode that restricts the number of integration servers that you can use; see Restrictions that apply in each operation mode.
- **Solution:** Contact your IBM representative to upgrade your license, or remove the required number of integration servers; see “Deleting an integration server by using the IBM Integration Toolkit or IBM Integration Explorer” on page 46.

Message BIP1822

- **Scenario:** Warning message BIP1822 is displayed.
WARNING: Broker '*Broker_Name*' is in '*Mode_Name*' mode but has 'x' message flows deployed, which exceeds the allowed maximum for this mode.
- **Explanation:** Your broker is running in a mode that restricts the number of message flows that you can use; see Restrictions that apply in each operation mode.
- **Solution:** Contact your IBM representative to upgrade your license, or delete the required number of message flows; see Deleting a message flow or subflow.

Message BIP1823

- **Scenario:** Warning message BIP1823 is displayed.

WARNING: Broker '*Broker_Name*' has a message flow called '*Message_Flow*' in integration server '*Execution_Group*', which contains one or more nodes that are not valid in this mode: *Mode_Name*

- **Explanation:** Your broker is running in a mode that restricts the types of node that you can use in a message flow; see Restrictions that apply in each operation mode.

- **Solution:** Contact your IBM representative to upgrade your license, rework your message flow to use nodes that are valid in the current mode, or remove the message flows that contain unsupported nodes; see *Deleting a message flow or subflow*.

Message BIP1824

- **Scenario:** Warning message BIP1824 is displayed.
WARNING: The trial period for broker '*Broker_Name*' expired on '*Day_Month_Year*'.
- **Explanation:** Your broker is running in the Trial Edition mode and your trial period of 90 days has expired; see *Restrictions that apply in each operation mode*.
- **Solution:** Contact your IBM representative to upgrade your license.

The message flow deploys on the test system, but not elsewhere: Procedure

- **Scenario:** The message flow that you have developed deploys on the test system, but not elsewhere.
- **Solution:** Carry out the following checks:
 - Ensure that you have verified the installation on the target system by creating and starting a broker, and deploying a single integration server. These actions confirm that the broker is correctly defined.
 - Ensure that the broker archive (BAR) file's *broker.xml* file contains references to the correct resources for the new system.
 - Ensure that any referenced message sets are deployed.
 - If database resources or user-defined nodes are not accessible or authorized from the target system, the deploy fails. On distributed systems, ensure that you have defined either ODBC or JDBC connections to your databases so that they can be accessed from the broker. Also, set the broker environment to allow access to the databases. On Linux or UNIX systems, you might have to run a database profile.
 - Any user-defined extensions that you are using in your message flow might not load if they cannot be found, or are not linked correctly. Consult the documentation for your operating system for details of tools that can help you to check the binary files of your user-defined extension.

Error messages about your broker mode are issued when you create an integration server:

About this task

Message BIP1825

- **Scenario:** Error message BIP1825 is displayed.

You attempted to create an integration server '*Execution_Group*' on broker '*Broker_Name*', but the broker is running in '*Mode_Name*' mode which limits the number of integration servers that can exist at any one time. The integration server cannot be created.

- **Explanation:** The integration server cannot be created because the maximum number of integration servers for the mode of the target broker has been reached, and creating the integration server causes this limit to be exceeded; see *Restrictions that apply in each operation mode*. The integration server was not created.
- **Solution:** Reuse an existing integration server, or delete an existing integration server and try the command again; see *“Deleting an integration server by using*

the IBM Integration Toolkit or IBM Integration Explorer” on page 46.
Alternatively, contact your IBM representative to upgrade your license.

Error messages about your broker mode are issued when you deploy: About this task

Message BIP1826

- **Scenario:** Error message BIP1826 is displayed.
You attempted to deploy a broker archive (BAR) file to integration server '*Execution_Group*' on broker '*Broker_Name*', but the broker is running in '*Mode_Name*' mode which limits the number of message flows that can exist at any one time.
The BAR file cannot be deployed.
- **Explanation:** The BAR file cannot be deployed because it causes the broker to run more message flows than are valid for the current mode of operation of the target broker; see Restrictions that apply in each operation mode. The BAR file was not deployed.
- **Solution:** Delete message flows from the broker and try the command again; see Deleting a message flow or subflow. Alternatively, contact your IBM representative to upgrade your license.

Message BIP1827

- **Scenario:** Error message BIP1827 is displayed.
You attempted to deploy a broker archive (BAR) file to integration server '*Execution_Group*' on broker '*Broker_Name*', but the broker is running in '*Mode_Name*' mode which has a restriction on the types of node that can be deployed.
The BAR file cannot be deployed.
The set of node types found in the BAR file that are not valid are: *Node_Type*.
- **Explanation:** The BAR file cannot be deployed because it contains nodes that are not valid for the current mode of the target broker; see Restrictions that apply in each operation mode. The BAR file was not deployed.
- **Solution:** Rework your message flow to use nodes that are valid in the current mode, or remove the message flows that contain unsupported nodes; see Deleting a message flow or subflow. Alternatively, contact your IBM representative to upgrade your license.

Message BIP1828

- **Scenario:** Error message BIP1828 is displayed.
You attempted to deploy a broker archive (BAR) file to integration server '*Execution_Group*' on broker '*Broker_Name*', but the trial period for the broker has expired.
The BAR file cannot be deployed.
- **Explanation:** The target broker is running in a trial mode that has expired; see Restrictions that apply in each operation mode. The BAR file was not deployed.
- **Solution:** Contact your IBM representative to upgrade your license. If you have already purchased a valid license for the target broker, change the broker to the correct mode by using the `mqsimode` command; see `mqsimode` command.

Message BIP1829

- **Scenario:** Error message BIP1829 is displayed.

You attempted to deploy a broker archive (BAR) file to integration server '*Execution_Group*' on broker '*Broker_Name*', but the broker is running in '*Mode_Name*' mode which limits the number of integration servers that can exist at any one time. The BAR file cannot be deployed.

- **Explanation:** The BAR file cannot be deployed because the maximum number of integration servers for the mode of the target broker has been reached; see Restrictions that apply in each operation mode. The BAR file was not deployed.
- **Solution:** Delete an existing integration server and try the command again; see “Deleting an integration server by using the IBM Integration Toolkit or IBM Integration Explorer” on page 46. Alternatively, contact your IBM representative to upgrade your license.

Error messages about your function level are issued when you deploy:

About this task

Message BIP2276

- **Scenario:** Error message BIP2276 is displayed.
The flow '*Message_flow*', which includes a message flow of node type '*Node_type*', cannot be deployed because the current fix pack function level of '*<version>.<release>.<modification>.<fix>*' does not support this node.
Use **mqsichangebroker -f all** to enable this functionality.
- **Explanation:** The broker received an instruction to create a message flow node of type *Node_type* in message flow *Message_flow*. The broker cannot create nodes of this type because the new functions have not been enabled for this broker. Use the following command to enable this function, and all other functions:
mqsichangebroker IB9NODE -f all
- **Solution:** Either change the flow to avoid using the unavailable node, or enable the new functions by using the **mqsichangebroker** command; see **mqsichangebroker** command.

Error messages are issued when you deploy to z/OS:

Procedure

- **Scenario:** The following messages are written to the log when you deploy to z/OS:
+(MQ05BRK) 0 BIP2070E: A problem was detected with WebSphere MQ while issuing MQPUT for WebSphere MQ queue SYSTEM.BROKER.ADMIN.REPLY, WebSphere MQ queue manager QM_01. MQCC=2, MQRC=2030.
+(MQ05BRK) 0 BIP2068E: The broker was unable to put an internal configuration message to message queue SYSTEM.BROKER.ADMIN.REPLY.
- **Explanation:** The transmission queue is not large enough for the messages that are issued by IBM Integration Bus.
- **Solution:** See the WebSphere MQ documentation for details on how to increase the size of the transmission queue.

Expected serialization of input is not occurring for a shared queue that serves multiple instances of a message flow on z/OS:

Procedure

- **Scenario:** Expected serialization of input is not occurring for a shared queue that serves multiple instances of a message flow on z/OS.

- **Explanation:** On z/OS, WebSphere MQ supports serialized access to shared resources, such as shared queues, through the use of a connection tag (serialization token) when an application connects to the queue manager that participates in a queue sharing group.

This problem can occur for a number of reasons:

- You have assigned different serialization tokens to the input nodes of the flows that get messages from the shared queue.
 - The message flows are running in the same integration server. Serialization can only be effected between flows that are running in different integration servers, either on the same broker, or on different brokers whose queue managers participate in the same queue sharing group.
 - The queue managers on which the brokers are running are not participating in a queue sharing group, or not participating in the same queue sharing group.
- **Solution:** Carry out the following checks:
 - Check that the same z/OS serialization token has been configured for the MQInput nodes in each flow that use the shared queue.
 - Check that the message flows are running in different integration servers.
 - Check that the broker queue managers are part of the same queue sharing group, and that no errors are reported from the queue managers in the z/OS system log (SDSF log).

You create a configurable service, then deploy a message flow and inbound adapter, but the deployment fails:

Procedure

- **Scenario:** You create a configurable service, then deploy a message flow and inbound adapter, but the deployment fails.
- **Explanation:** When you create a configurable service for an adapter that has not yet been deployed, the connection properties are not fully validated until you deploy the adapter and the message flow that uses that adapter. Therefore, the properties that you set on the configurable service might be invalid.
- **Solution:** Inspect the error message that is returned from the deployment and determine whether the deployment failed because of invalid connection properties on the configurable service. If so, use the `mqsichangeproperties` command to correct the properties, or use the `mqsdeleteconfigurable` command to use the properties that are set on the adapter. Restart the broker and redeploy the message flow.

You have created a WebSphere Adapters message flow that uses secondary adapters, and a naming clash has occurred in the secondary adapters or message sets:

Procedure

- **Scenario:** You have created a WebSphere Adapters message flow that uses secondary adapters, and a naming clash has occurred. You need to know which adapters and message sets the WebSphere Adapters node is using, and if the method names in the adapters or the message type names in the message sets are clashing.
- **Explanation:** Method names must be unique across all primary and secondary adapters, and the message set that is created must not contain any types that share the same name and namespace of existing message sets. Information about secondary adapters and message sets is written to user trace at the following stages.

- When the node is first deployed, information about the current set of secondary adapters and message sets is reported.
- When adapters and message sets are deployed subsequently, information about them is reported at the time of deployment.
- When the message flow, broker, or integration server is stopped then restarted, information about the entire set of secondary adapters and message sets is written to user trace, including those secondary adapters and message sets that are deployed after the message flow was first deployed.

This information is reported by the following messages.

- BIP3432 and BIP3434 list the adapters and message sets that are available when the node is deployed or when the broker, integration server, or message flow are restarted.
- BIP3433 reports that a secondary adapter is being deployed and added to the set.
- BIP3435 reports that a secondary message set is being deployed.
- BIP3436 identifies the message set in which each type is being defined.
- BIP3437 identifies message sets that attempt to redefine a type that is already defined.
- BIP3438 identifies the adapter in which each method is being defined.
- BIP3439 identifies adapters that attempt to redefine a method that is already defined.

Messages BIP3436, BIP3437, BIP3438, and BIP3439 are issued when a message is processed to report whether definitions are used for that message.

- **Solution:** The following steps describe how to use user trace when working with secondary adapters.
 - **When you deploy the flow for the first time**
 1. Start user trace.
 2. Deploy your message flow, primary adapter, primary message set, and any secondary adapters or message sets that are ready.
 3. Read user trace, looking out for the messages described.
 4. Optional: Reset user trace, stop and restart the message flow, then read user trace again, looking out for the messages described.
 - **When you add new adapters and message sets**
 1. Start user trace.
 2. Deploy the secondary adapters and message sets.
 3. Read user trace, looking out for the messages described.
 4. Optional: Reset user trace, stop and restart the message flow, then read user trace again, looking out for the messages described.

When you have identified where the naming clash has occurred, you can ensure that names are unique by editing them in the following ways:

- Edit method names by clicking **Edit Operations** on the Service Generation and Deployment Configuration panel of the Adapter Connection wizard.
- Edit the namespaces of the types in the message set by using the Business Object Namespace control on the Adapter Connection wizard.

You get an authority check failure when you deploy to z/OS:

Procedure

- **Scenario:** You deploy a BAR file to z/OS, and you get an authority check failure like the following message:


```

ICH408I USER(MI09STC ) GROUP(TSOUSER ) NAME(WMB TASK ID          ) 672
MI09.SYSTEM.BROKER.AUTH.default CL(MQQUEUE )
PROFILE NOT FOUND - REQUIRED FOR AUTHORITY CHECKING
ACCESS INTENT(READ ) ACCESS ALLOWED(NONE )

```

- **Explanation:** The z/OS queue manager accesses the MQQUEUE security profile class by default. If you have enabled broker administration security, and you have specified the names of one or more integration servers in mixed case, the security profiles that you set up are defined in MXQUEUE, because the associated authority queue names are also in mixed case. MQQUEUE does not hold the expected information.
- **Solution:** Change the security profile case parameter for the queue manager. You can use the ALTER QMGR command to update the SCYCASE parameter to M, or you can modify your start job JCL to set or change this parameter, and restart the queue manager.

Deployment fails when you have circular project dependencies:

Procedure

- **Scenario:** Circular project dependencies exist in the projects that you are deploying and deployment fails.
- **Explanation:** A circular project dependency occurs when two or more projects depend on each other.

For example, File A in Project X depends on File B in Project Y, and File C in Project Y depends on File D in Project X. For Project X to build successfully, Project Y must be built first so that Project X can resolve the dependency on File B. However, for Project Y to build successfully, Project X must be built first so that Project Y can resolve the dependency on File D. The IBM Integration Toolkit is based on the Eclipse platform, which does not support circular project dependencies.

- **Solution:** To deploy successfully, avoid circular project dependencies. For example, if Project X depends on File B in Project Y, and Project Y depends on File D in Project X, move File B to Project Z. The projects must be built in the order Project Z, Project X, then Project Y so that the dependencies can be resolved.

Error messages are issued when you deploy:

About this task

Additional error messages that might be generated during a deployment are explained in this section.

Message BIP1106 with WebSphere MQ reason code 2030

- **Scenario:** Error message BIP1106 is issued with reason code 2030 when you are deploying a large message set.
- **Explanation:** The size of the message exceeds the maximum message length of the transmission queue to the broker queue manager.
- **Solution:** Increase the maximum message length for the transmission queue using the WebSphere MQ **alter ql** command, where the maximum message length (maxmsgl) is in bytes:

```
alter ql(transmit_queue_name) maxmsgl(104857600)
```

For more information about this command, see the *System Administration Guide* section of the WebSphere MQ Version 7 product documentation online.

Message BIP2066

- **Scenario:** You have initiated a deployment request; for example, you have deployed a BAR file to an integration server. Error message BIP2066 is returned one or more times.
- **Explanation:** The deployment request was not acknowledged by the integration server before the sum of the values for the broker timeout parameters **ConfigurationChangeTimeout** and **InternalConfigurationTimeout** expired.
- **Solution:** Increase these timeout values by specifying the **-g** and **-k** parameters of the **mqsicreatebroker** or **mqsichangebroker** command. See “Setting configuration timeout values” on page 548 for information about factors that affect timeout values, and how to set appropriate values.

Message BIP2080

- **Scenario:** The broker has started an integration server; for example, if you have issued **mqsistart** for the broker, or an error has occurred and the integration server is being recovered. Error message BIP2080 is displayed one or more times.
- **Explanation:** The internal configuration request was not acknowledged by the integration server before the value of the **InternalConfigurationTimeout** (default 60 seconds) expired.
- **Solution:** Change the configuration timeout by specifying the **-k** parameter of the **mqsicreatebroker** or **mqsichangebroker** command. See “Setting configuration timeout values” on page 548 for information about factors that affect timeout values, and how to set appropriate values.

Message BIP2241

- **Scenario:** Error message BIP2241 is displayed.
- **Explanation:** You are attempting to deploy a message flow containing a node that is not available on the target broker.
- **Solution:** Ensure that the version of the IBM Integration Toolkit in which the message flow has been developed matches the version of the broker to which the message flow is being deployed. If the message flow is using a user-defined node, or a node supplied in a SupportPac, ensure that the runtime node implementation has been correctly installed on the computer on which the broker is running. If your message flow includes a user-defined node, see Installing user-defined extension runtime files on a broker. If your message flow includes a node provided in a SupportPac, see the installation information, if supplied, for the SupportPac.

Message BIP2242

- **Scenario:** Error message BIP2242 is displayed.
- **Explanation:** The deployment request (configuration change) was not accepted before the timeout value set by the broker parameter **ConfigurationChangeTimeout** expired. This configuration timeout value must be long enough for the message flow to complete processing its current message, then accept the deploy request; the default is 300 seconds.
- **Solution:** Set the configuration timeout values by specifying the **-g** and **-k** parameters of the **mqsicreatebroker** or **mqsichangebroker** command.

Message BIP3226

- **Scenario:** Error message BIP3226 is displayed; for example:
(Semipersistent_Compute1.Main, 27.89) : Array index evaluated to '0' but must evaluate to a positive, nonzero integer value.

The first insert in BIP3226 (in this example, Semipersistent_Compute1.Main) identifies the node and routine in which the statement occurs. The second insert (in this example, 27.89) identifies the approximate line and column of the index value shown in the third insert (in this example, '0').

- **Explanation:** The validity of using a field reference index of zero was corrected in WebSphere Message Broker Version 7.0. If you have statements in your ESQL modules that include an index of zero, error BIP3226E is generated.

For example, your ESQL module might contain the following statement:

```
SET OutputRoot.XMLNSC.Top.A[0].B = 42;
```

- **Solution:**

You must correct all ESQL statements that use an index of zero to use an index of 1. Statements might use a variable as well as a literal value for the index; check for both possible situations. For example, your changed code might read:

```
SET OutputRoot.XMLNSC.Top.A[1].B = 42;
```

Message BIP7053S

- **Scenario:** When you deploy to a broker, error message BIP7053S is displayed.
- **Explanation:** This error occurs in a multi TCP/IP stack environment, and indicates that the UNIX System Services (USS) TCP/IP environment has not been set up correctly.

IBM Integration Bus uses USS functions to obtain the host name for a particular system. The following error message is displayed if the default host name is not set up correctly in the USS environment:

```
BIP7053S: Broker $SYS_mqsi 0 unexpected Java exception java.lang.Error:
-2103399272!java.net.UnknownHostException :
Hostname: Hostname
```

The host name that is reported in the error message is the one that has been returned to the broker as a result of the gethostname call.

- **Solution:** Ensure that the TCP/IP environment is configured correctly in USS.

Tagged/Delimited String (TDS) Validator error

- **Scenario:** You try to deploy a message set with a TDS wire format that has an error.
- **Explanation:** The following extract from an error log illustrates what you might see for a TDS Validator error. In this case, the cause of the problem is that the element Town does not have a tag defined.

```
TDS Extractor Trace File
=====
```

```
Beginning Extract..
```

```
Extracting Identification Info
Extracting Project Info
Extracting Messages
Extracting Elements
Extracting Compound Types
Extracting Type Members
Extracting Type Members
Extracting Type Members
Extracting Type Members
Extracting Type Members
Extracting Type Members
Beginning Indexing..
```

Creating Member IDs to Tags Index Table.

Beginning Validation..

Validating Project

Validating Types

ERROR: TDSValidator::ValidateTypeMemberSimpleElement:

Simple elements in a type with Data Element Separation attribute = Tagged
Delimited must have the following attribute set:

Element Level - Tag

(Element ID: Town)

(Type ID: AddressType)

Return Code: -80

Validating Messages

Trace Info

=====

EXCEPTION: TDSValidator::Validate:

TDS Validation failed.

1 errors

0 warnings

Return Code: -1

- **Solution:** Use the information in the error log to correct the problem.

:

Related concepts:



Packaging and deployment overview

Deployment is the process of transferring data to an integration server on a broker so that it can take effect in the broker. Message flows and associated resources are packaged in broker archive (BAR) files for deployment.



WebSphere Adapters deployment

When you have run the Adapter Connection wizard and created a message flow, you must deploy the resources that are generated by adding them to a broker archive (BAR) file.

Related tasks:

“Resolving problems when deploying message flows or message sets” on page 752
Use the advice given here to help you to resolve common problems that can arise when you deploy message flows or message sets.



Packaging and deploying

Package resources that you create in the IBM Integration Toolkit, such as message flows, and deploy them to integration servers on brokers.

“Are the Linux and UNIX environment variables set correctly?” on page 652

Use the **mqsiprofile** command to set a command environment.

Related reference:



mqsideploy command

Use the **mqsideploy** command to make a deployment request to the broker.



mqsilist command

Use the **mqsilist** command to list installed brokers and their associated resources.



mqsistart command

Use the **mqsistart** command to start the specified broker if all initial verification

tests complete successfully.



mqsistop command

Use the **mqsistop** command to stop the specified component.



mqsicreatebroker command

Use the **mqsicreatebroker** command to create a broker and its associated resources.



mqsichangebroker command

Use the **mqsichangebroker** command to change one or more of the configuration parameters of the broker.



mqsichangeproperties command

Use the **mqsichangeproperties** command to modify broker properties and properties of broker resources.



mqsicreateconfigurable-service command

Use the **mqsicreateconfigurable-service** command to create an object name for a broker external resource.



mqsdeleteconfigurable-service command

Use the **mqsdeleteconfigurable-service** command to delete a configurable service, such as a JMS provider, JDBC provider, or FTP server, that you have created by using the **mqsicreateconfigurable-service** command.

Related information:



[WebSphere MQ Version 7 product documentation](#)

Resolving problems that occur after deployment of message flows

Use the advice given here to help you to resolve common problems that can arise after deploying message flows or message sets.

You receive exception ('MQCC_FAILED') reason '2042' ('MQRC_OBJECT_IN_USE'): Procedure

- **Scenario:** This exception message occurs when the following conditions are met:
 1. You have a message flow containing a SOAPRequest node that is using the JMS Transport.
 2. The message flow has additional instances defined, and is running under load.
 3. You are using the WebSphere MQ JMS provider.
 4. You have not specified a reply-to destination, and so are using JMS temporary dynamic queues.
- **Explanation:** If you do not specify a reply-to destination, and you are using the WebSphere MQ JMS provider in a message flow with additional instances, you must configure JMS temporary dynamic queues before deploying your message flow.
- **Solution:** Complete the steps in the topic [Configuring JMS temporary dynamic queues for the WebSphere MQ JMS provider](#).

Your XSLTransform node does not work after deployment:

About this task

Two scenarios explain why your XSLTransform node might not work after deployment:

Error messages are displayed indicating that the style sheets were not found

Procedure

- **Scenario:** Error messages are displayed indicating that the style sheets were not found.
- **Explanation:** This error message is displayed if the broker cannot find the style sheets or XML files required, or if the content of a style sheet or XML file is damaged and therefore no longer usable. This error message can happen if a file system failure occurs during a deployment.
- **Solution:** If the style sheets or XML files are damaged, redeploy the damaged style sheets or XML files.

What to do next

You get unexpected transformation results

- **Scenario:** You get unexpected transformation results.
- **Explanation:** For complex message flows, incompatibility might arise among style sheets and XML files after a deployment. The two typical reasons for this error are:
 - Only part of the cooperating style sheets or XML files are deployed and updated (a file system failure could cause this failure).
 - Multiple XSLTransform nodes that are running inside the same integration server are supposed to use compatible style sheets, but are using different versions to process the same incoming message.
- **Solution:** If only part of the cooperating style sheets or XML files are deployed and updated, resolve all incompatibility by redeploying the compatible versions. To avoid multiple XSLTransform nodes using different versions of the style sheet, pause relevant message flows in the target integration server before performing the deployment, then restart the flows.

Related concepts:



Packaging and deployment overview

Deployment is the process of transferring data to an integration server on a broker so that it can take effect in the broker. Message flows and associated resources are packaged in broker archive (BAR) files for deployment.

Related tasks:

“Resolving problems when deploying message flows or message sets” on page 752
Use the advice given here to help you to resolve common problems that can arise when you deploy message flows or message sets.



Packaging and deploying

Package resources that you create in the IBM Integration Toolkit, such as message flows, and deploy them to integration servers on brokers.



Configuring JMS temporary dynamic queues for the WebSphere MQ JMS provider

Configure JMS temporary dynamic queues for the WebSphere MQ JMS provider, so that you can use them with SOAPRequest nodes, by using WebSphere MQ

Explorer.

“Are the Linux and UNIX environment variables set correctly?” on page 652

Use the **mqsiprofile** command to set a command environment.

Related reference:



mqsdeploy command

Use the **mqsdeploy** command to make a deployment request to the broker.



mqsistart command

Use the **mqsistart** command to start the specified broker if all initial verification tests complete successfully.



mqsistop command

Use the **mqsistop** command to stop the specified component.



XSLTransform node

Use the XSLTransform node to transform an XML message to another form of message, according to the rules provided by an XSL (Extensible Stylesheet Language) style sheet, and to set the Message domain, Message set, Message type, and Message format for the generated message.

Resolving problems that occur when debugging message flows

Advice on dealing with some of the common problems that you see when debugging message flows.

About this task

“Resolving problems that occur when starting and stopping the debugger” on page 769

- “An endless “waiting for communication” progress bar is displayed when you start the debugger” on page 769
- “The debugger seems to stop” on page 769
- “The session ends abnormally while debugging” on page 770
- “An error message is displayed indicating that the debug session cannot be launched” on page 770
- “Errors are generated when you copy a message map into an application, library, or integration project” on page 770

“Resolving problems when debugging message flows” on page 771

- “The debugger does not pause at the next breakpoint” on page 771
- “The message does not stop executing at any breakpoint” on page 771
- “Editing problems occur in the Message Flow editor” on page 772
- “Editing the MQ message descriptor (MQMD) causes unexpected behavior in the debugger” on page 772
- “You cannot see the message content when debugging your message flow” on page 772
- “An exclamation mark appears above a node during debugging” on page 772
- “The message does not stop processing at breakpoints” on page 772

“Resolving problems that occur after debugging:” on page 774

- “You cannot change a message flow after debugging” on page 774

- “You redeployed a debugged message flow but deployment hangs” on page 774

Related concepts:



Flow debugger overview

Use the flow debugger in the IBM Integration Toolkit to track messages through your message flows.

Related tasks:



Starting the flow debugger

To start the flow debugger, you must attach it to an integration server. When the flow debugger is started, you can introduce test messages to your message flow.



Attaching the flow debugger to an integration server for debugging

Before you can debug your message flow, you must attach the flow debugger to the integration server where your flow is deployed, then start a debugging session.



Debugging a message flow

Use the tasks described in this section of the documentation to manage and work with the flow debugger.



Debug: redeploying a message flow

If you want to change your message flow while you are debugging it, you must redeploy it to the integration server, then reattach the flow debugger.

Resolving problems that occur when starting and stopping the debugger

Use the advice given here to help you to resolve common problems that can arise when you debug message flows.

An endless "waiting for communication" progress bar is displayed when you start the debugger:

Procedure

- **Scenario:** After you click **Start Debugging**, you get an endlessly cycling progress bar entitled "waiting for communication". The "debug session started" message is not displayed in the information pane.
- **Explanation:** If the message flow has nodes with ESQL statements, the flow might not deploy even if the statements are correct syntactically. This situation can occur, for example, because of multiple declarations or uninitialized variables (that is, semantic problems that the syntax parser does not pick up). Always check the IBM Integration Toolkit Administration log to confirm that the debugged version of your message flow has deployed successfully; it has the same name as the original message flow, with the suffix `_debug_`.
If the message flow does not deploy properly, the debugger cannot establish communication with the flow, and you see the endless progress bar.
- **Solution:** Click **Cancel** to clean up and return to a good state, then fix your errors and try again. As a check, see if your flow can deploy without the debugger.

The debugger seems to stop:

Procedure

- **Scenario:** You are debugging a message flow and continue after encountering a breakpoint. However, nothing seems to happen and after about a minute, a progress bar appears, indicating that the debugger is waiting for communication.

- **Explanation:** This situation can occur for the following reasons:
 - The message flow might have encountered a time-intensive operation, such as a huge database query, and you must wait for the flow to complete the action.
 - The broker ended, or some other extraordinary condition occurred, and communication was lost. In this case, click **Cancel** to stop the debug session.


The session ends abnormally while debugging:

Procedure

- **Scenario:** After debugging a message flow, the session ends abnormally and you still have the debug instance of the message flow (mf_debug_) deployed to the broker's integration server. You are concerned that this is going to affect the operation of the flow, and want to put the integration server back to its original state.
- **Explanation:** The orphaned message flow should behave as the flow would have done normally, and the Debug nodes have no effect on message processing. If you have a small number of nodes in the message flow, corrective action makes no noticeable difference to the flow, apart from its name. However, if you have a large message flow (that is, more than 15 nodes or several subflows), take the corrective action described later in this section, because the performance of message processing might be affected.
- **Solution:** Redeploy the broker.
A full redeploy of the broker should replace the orphaned flow with the original message flow. If this action has no effect, remove the orphaned flow from the integration server, deploy, then add the flow and deploy to restore the original state of the broker as it was before the debugging session.

An error message is displayed indicating that the debug session cannot be launched:

Procedure

- **Scenario:** You try to relaunch or invoke a new debug session but when you click the green **Debug** icon , an error message is displayed stating: Cannot launch this debug session.
- **Explanation:** When you click **Debug**, it relaunches the last debug session. It fails if you have not created a debug session previously. It also fails if the broker and integration server that were attached previously in a debug session are no longer running, or have been restarted; the session cannot be reattached without re-selection of the broker and integration server process instance.
- **Solution:**
 1. Close the error message and click the arrow immediately to the right of the **Debug** icon.
 2. Re-select or modify the broker and integration server information from the previous debug launch configuration by clicking **Debug** in the menu and selecting the previous debug launch configuration. See *Attaching the flow debugger to an integration server for debugging* for more information.

Errors are generated when you copy a message map into an application, library, or integration project:

Procedure

- **Scenario:** You are copying a message map into an application, library, or integration project and errors have appeared in the task list.

- **Explanation:** The application, library, or integration project did not have the correct references set before you copied the message mapping.
- **Solution:** These errors remain in the task list, even if you reset the project references immediately after copying. Therefore, you must perform a clean build of the application, library, or integration project.

Related concepts:



Flow debugger overview

Use the flow debugger in the IBM Integration Toolkit to track messages through your message flows.

Related tasks:

“Resolving problems that occur when debugging message flows” on page 768
Advice on dealing with some of the common problems that you see when debugging message flows.



Starting the flow debugger

To start the flow debugger, you must attach it to an integration server. When the flow debugger is started, you can introduce test messages to your message flow.



Attaching the flow debugger to an integration server for debugging

Before you can debug your message flow, you must attach the flow debugger to the integration server where your flow is deployed, then start a debugging session.



Debugging a message flow

Use the tasks described in this section of the documentation to manage and work with the flow debugger.



Debug: redeploying a message flow

If you want to change your message flow while you are debugging it, you must redeploy it to the integration server, then reattach the flow debugger.

Resolving problems when debugging message flows

This topic contains advice for dealing with some of the common problems that can arise when debugging message flows.

The debugger does not pause at the next breakpoint:

Procedure

- **Scenario:** The message flow debugger does not pause at the next breakpoint in your message flow.
- **Solution:** Perform the following checks:
 1. Check whether your DataFlowEngine is running; if it is not, restart it.
 2. Check the input queue. If your input queue has the leftover messages from the previous time that you used the debugger, clear them before you send a new message.

The message does not stop executing at any breakpoint:

Procedure

- **Scenario:** The message does not stop executing at any breakpoint after you attach to the debugger.
- **Explanation:** This error could be caused by a timing problem, or if you have set the wrong parameters for the debug session.
- **Solution:** Perform the following steps.

1. Check your launch configuration settings, ensuring that you have specified the correct Flow Project, HostName and Flow Engine for the debug session.
2. Restart the debug session.

Editing problems occur in the Message Flow editor:

Procedure

- **Scenario:** While debugging a message flow, editing problems occur when you are using the Message Flow editor.
- **Solution:** Do not attempt to edit the message while the flow debugger is attached. To edit a message flow, detach the debugger, edit the message flow, then redeploy the message flow.

Editing the MQ message descriptor (MQMD) causes unexpected behavior in the debugger:

Procedure

- **Scenario:** You edit properties of the message MQMD descriptor in the Message Set editor, but this causes unexpected behavior in the debugger.
- **Explanation:** If you edit the content of the MQMD descriptor, these fields take a certain range of values. You need to know these ranges before editing the properties. Unless you explicitly specify the value of these fields, they take default values, and certain fields might not have been specified in the message. The values in the fields that are not set explicitly in the message are default values; do not change these unless you are aware of their importance or the possible range of values.

You cannot see the message content when debugging your message flow:

Procedure

- **Scenario:** You are using the message flow debugger, and you can see the message passing through the message flow, but you cannot see the content of the message.
- **Solution:** Open the Flow Debug Message view by clicking **Window > Show View > Other > Message Flow > Flow Debug Message**, then **OK**.

An exclamation mark appears above a node during debugging:

Procedure

- **Scenario:** In the Message Flow editor, an exclamation mark (!) is displayed above a node during debugging.
- **Explanation:** An exception has occurred in the node during debugging.
- **Solution:** Look under the ExceptionList in the Variables view of the Debug perspective to find out what error has occurred.

The message does not stop processing at breakpoints:

Procedure

- **Scenario:** Message processing continues when a breakpoint is encountered.
- **Explanation:** This error could be caused by a timing problem, or if you have set the wrong parameters for the debug session.
- **Solution:** Check your launch configuration setting. Ensure you have specified the correct Flow Project, HostName and Flow Engine for the debug session. Restart the debug session.

You cannot see where the debugger is in the Graphical Data Mapping editor:

Procedure

- **Scenario:** The Graphical Data Mapping editor has opened in the toolkit, but it is unclear where the debugger is in the map.
- **Explanation:** The source lookup path for the message map file is not configured correctly.
- **Solution:** Check your debug launch configuration settings and ensure you have configured the source lookup path for the message map file correctly.

When debugging a message map, the debugger does not move to the next field: Procedure

- **Scenario:** You are debugging a message map, and the debugger does not move to the next field. You have to click the **Step over** button multiple times.
- **Explanation:** The source lookup path for the message map file is not configured correctly.
- **Solution:** Check your debug launch configuration settings and ensure you have configured the source lookup path for the message map file correctly.

When debugging a message map, the debugger does not move out of the mapping node: Procedure

- **Scenario:** You are debugging a message map, and the debugger does not move out of the message map.
- **Explanation:** The source lookup path for the message map file is not configured correctly.
- **Solution:** Check your debug launch configuration settings and ensure you have configured the source lookup path for the message map file correctly.

The message flow stops at a collector node: Procedure

- **Scenario:** Message processing stops after selecting the Step into Source Code icon on a Collector node.
- **Explanation:** The collector node is a multithreaded node and the thread is ended by selecting Step into Source Code.
- **Solution:** Set a breakpoint manually after the collector node.

Related concepts:



Flow debugger overview

Use the flow debugger in the IBM Integration Toolkit to track messages through your message flows.

Related tasks:

“Resolving problems that occur when debugging message flows” on page 768
Advice on dealing with some of the common problems that you see when debugging message flows.



Starting the flow debugger

To start the flow debugger, you must attach it to an integration server. When the flow debugger is started, you can introduce test messages to your message flow.



Attaching the flow debugger to an integration server for debugging

Before you can debug your message flow, you must attach the flow debugger to the integration server where your flow is deployed, then start a debugging session.

Debugging a message flow

Use the tasks described in this section of the documentation to manage and work with the flow debugger.

Debug: redeploying a message flow

If you want to change your message flow while you are debugging it, you must redeploy it to the integration server, then reattach the flow debugger.

Resolving problems that occur after debugging:

This topic contains advice for dealing with some of the common problems that can arise after debugging message flows.

You cannot change a message flow after debugging:

Procedure

- **Scenario:** You were debugging, but now your message flow seems to be stuck. When you put a new message in, nothing happens.
- **Explanation:** This might be because a message was backed out, but you have not set the **Backout requeue name** property of your input queue.
- **Solution:** Set the **Backout requeue name** property to a valid queue name (such as the name of the input queue itself) and your flow will become unstuck.

You redeployed a debugged message flow but deployment hangs:

Procedure

- **Scenario:** You found problems in your message flow by using the debugger. You changed your message flow and then redeployed it, but now the deployment hangs.
- **Solution:** Make sure that when you redeploy the flow to an integration server, the integration server is not still attached to the debugger.

Related concepts:

Flow debugger overview

Use the flow debugger in the IBM Integration Toolkit to track messages through your message flows.

Related tasks:

“Resolving problems that occur when debugging message flows” on page 768
Advice on dealing with some of the common problems that you see when debugging message flows.

Starting the flow debugger

To start the flow debugger, you must attach it to an integration server. When the flow debugger is started, you can introduce test messages to your message flow.

Attaching the flow debugger to an integration server for debugging

Before you can debug your message flow, you must attach the flow debugger to the integration server where your flow is deployed, then start a debugging session.

Debugging a message flow

Use the tasks described in this section of the documentation to manage and work with the flow debugger.

Debug: redeploying a message flow

If you want to change your message flow while you are debugging it, you must redeploy it to the integration server, then reattach the flow debugger.

Resolving problems when developing message models

This topic contains advice for dealing with some common problems that can arise when working with message sets.

About this task

Message definition files:

- “Your message definition file does not open”
- “A message definition file error is written to the task list” on page 776
- “Your physical format property values have reverted to defaults” on page 776
- “You are unable to enter text in the Message Definition editors” on page 776
- “Objects in your message definition file are not listed in alphabetic order” on page 776
- “You want to make the Properties tab the default tab in the Message Definition editor” on page 778
- “Error messages are written to the task list after you have imported related XML Schema files” on page 777
- “A group contains two different elements with the same XML name in the same namespace” on page 777
- “You are unable to set up a message definition file to include a message definition file within another message definition file” on page 777
- “Error message BIP5410 is issued because a union type cannot be resolved” on page 778
- “Error message BIP5395 is issued because an xsi:type attribute value does not correspond to a valid member type of the union” on page 779
- “Error message BIP5396 is issued because a data type does not correspond to any of the valid data types of the union” on page 779
- “A union type contains two or more simple types that are derived from the same fundamental type” on page 779
- “A list type is based on a union that also contains a list type” on page 780
- “A union type contains an enumeration or pattern facet” on page 780
- “Error message BIP5505 is issued because input data is not valid for the data type” on page 780

Other:

- “A deprecation error is issued on an imported .mrp file” on page 781
- “User trace detects an element length error” on page 781
- “MRM dateTime value has changed after it has been parsed” on page 781

Your message definition file does not open

Procedure

- **Scenario:** You have created a complex type with a base type that causes the types to be recursive, and now your message definition file does not open.
- **Solution:** Recover the last saved version of your message definition file from the local history or from your repository.

A message definition file error is written to the task list

Procedure

- **Scenario:** An error is written to the task list indicating that the message definition file has been corrupted.
- **Explanation:** This error message appears when the base type of the complex type is itself, or a circular definition of two or more simple types. This type of recursion is not supported.
- **Solution:** Examine your code and ensure that the type of recursion described above does not occur.

Your physical format property values have reverted to defaults

Procedure

- **Scenario:** Someone has sent you a message definition file. You have added it to a message set project, but all the physical format property values have reverted to defaults.
- **Explanation:** You cannot transfer message definition files on their own in this way because these files are not entirely independent. It is the messageSet.mset file that lists all the physical formats that are applicable to a message set. For example, if you have a message set *A* with CWF format *Binary1*, and a message set *B* without any physical formats, a message definition file copied from *A* to *B* does not show *Binary1* as a known physical format (although the properties are still in the message definition file).
- **Solution:** If possible, request the entire message set project.
Alternatively, add physical formats to the receiving message set so that they match the originating message set. Then any dormant properties become visible. However, make sure that the message set level physical format properties match, or those object properties that inherit defaults from the message set level will be incorrect.

You are unable to enter text in the Message Definition editors

Procedure

- **Scenario:** You are unable to enter text in the Message Definition Overview editor, or change the text of an enumeration or pattern facet in the Message Definition Properties editor.
- **Solution:** The Message Definition editors are cell editors, and to enable them you have to double-click the cell. The first click selects the cell; the second click puts the cell into active state, allowing you to edit it.

Objects in your message definition file are not listed in alphabetic order

Procedure

- **Scenario:** The objects in your message definition file (.mxsd) are listed in the order that you created them, but you want them to be in alphabetic order.
- **Solution:**
 1. In the Integration Development perspective, double-click the message definition file to open it in the Outline view.
 2. Click **az** (at the top of the Outline view).

This action changes the order of objects within each of the message definition file's folders (the Messages, Types, Groups, Elements, and Attributes folders) to be alphabetic.

Error messages are written to the task list after you have imported related XML Schema files

Procedure

- **Scenario:** When you have finished importing a collection of related XML Schema files, you find that the IBM Integration Toolkit task list contains error messages for the message set project, indicating that referenced types or other objects cannot be found.
- **Explanation:** These errors are typically an indication that one message definition file includes or imports another message definition file, but that the Message Definition editor is unable to resolve the specified link.
- **Solution:**
 - Check that a message definition file exists in the IBM Integration Toolkit for each of the related XML Schema files that you are importing. If the new message definition files have been created, they appear in the Application Development view.
 - Using the Properties Hierarchy in the Message Definition editor, check that any message definition files that include or import other message definition files are correctly locating the target files.

One common scenario is where you have two XML Schema files *X* and *Y*, which both exist in the same directory in the file system but where *X*, which includes *Y*, is defined with a real target namespace, while *Y* is defined in the notarget namespace. After you import the two files, *X* is placed into the location that is determined by its namespace, but *Y* is placed into the default namespace location used for files defined in the notarget namespace. This situation causes the link between the two files to break, and you must modify *X* so that it correctly includes *Y* in its new relative location.

A group contains two different elements with the same XML name in the same namespace

Procedure

- **Scenario:** A warning is written to the task list because a group contains two different elements that have the same XML element name, in the same namespace.
- **Explanation:** When a message that has an XML physical format is validated, the validation includes a test to identify any part of a message definition where the parser might not be able to uniquely determine which element is represented by the XML name. This test generates a warning when a group contains two different XML elements with the same element name in the same namespace, even in cases where the duplication is legitimate.
- **Solution:** Determine whether the duplication that is identified in the warning message is in fact a problem that needs to be corrected, or whether it has arisen because of two elements on opposite sides of a choice sharing the same XML name, in which case the duplication is legitimate because no ambiguity can occur.

You are unable to set up a message definition file to include a message definition file within another message definition file

Procedure

- **Scenario:** You are unable to set up the include property of a message definition file to include a second message definition file that is included within another message definition file.
- **Explanation:** A message definition file can reference objects in another message definition file only if the first file references the second file directly. For example,

if three message definition files exist, *A*, *B* and *C*, where *A* includes *B* and *B* includes *C*, *A* can reference objects in *B*, and *B* can reference objects in *C*, but *A* cannot reference objects in *C*.

You might also encounter this problem after importing XML Schema files that have nested includes.

- **Solution:** You can work around this problem by including the message definition file directly, which, in the above example, would mean including *C* directly in *A*. Alternatively, you can define all the message set definitions that are in *C* directly in *B*, then delete *C* so that *A* needs to include only *B*.

You want to make the Properties tab the default tab in the Message Definition editor

Procedure

- **Scenario:** When using the Message Definition editor to edit your message definition files, you prefer to use the **Properties** tab rather than the **Overview** tab, but the **Overview** tab is always selected as the default.
- **Solution:** Use the IBM Integration Toolkit preferences to choose the **Properties** tab as the default tab:
 1. From the menu, click **Windows > Preferences**.
 2. Expand **Integration Development**, then **Message Set**.
 3. Expand **Editors** and select **Tab Extensions**.
 4. In the **Message Definition Editor** section, select the **Properties** and the **Overview** check boxes.
 5. Click **OK**.

Error message BIP5410 is issued because a union type cannot be resolved

Procedure

- **Scenario:** Error message BIP5410 is raised indicating that an element or attribute is based on a union type and that the element or attribute could not be cast to any member of the union.
- **Explanation:** When parsing an element or attribute that is based on a union type, the MRM XML parser uses an `xsi:type` attribute, where present, to resolve the union.

If an `xsi:type` attribute is not present, or an attribute is being parsed, the parser tries each union member type in turn, attempting to cast to the associated simple type, until the cast succeeds. The order of precedence for the attempted cast is the order in which the member types are listed in the message model, under the union type, in the Outline view.

If the data cannot be cast to any of the simple types within the union, the union cannot be resolved and a parser error is reported.

- **Solution:** Perform the following checks:
 - Check that the message contains a valid value for the element or attribute that is identified in the error message.
 - Check that the member types of the union that are identified in the error message contain the correct list of simple types.
 - If possible, use an `xsi:type` attribute to resolve the union explicitly.

Error message BIP5395 is issued because an xsi:type attribute value does not correspond to a valid member type of the union

Procedure

- **Scenario:** Error message BIP5395 is issued, indicating that an element is based on a union type that has an xsi:type attribute with a value that should resolve the union explicitly to one of its modeled member types and that the xsi:type attribute value does not correspond to a valid member type of the union.
- **Explanation:** When parsing or writing an element or attribute that is based on a union type, the MRM XML parser or writer uses an xsi:type attribute, where present, to resolve the union. The parser resolves the value of the xsi:type attribute to a simple type in the dictionary and checks that the simple type is a valid member type of the union.
If the xsi:type attribute identifies a type that is not a member type of the union, it reports an error.
- **Solution:** Perform one of the following steps:
 - Modify the message so that the xsi:type attribute identifies a valid member type of the union.
 - Check that the member types of the union identified in the error message contain the correct list of simple types.

Error message BIP5396 is issued because a data type does not correspond to any of the valid data types of the union

Procedure

- **Scenario:** Error message BIP5396 is issued, indicating that an element or attribute is based on a union type but the data type does not correspond to any of the valid data types in the union.
- **Explanation:** When writing an element or attribute that is based on a union type, the MRM XML writer uses an xsi:type attribute, where present, to resolve the union. If an xsi:type attribute is not present, the writer tries to match the data type of the literal value in the tree to a simple type in the union. If the literal value cannot be matched to any of the simple types in the union, it reports a writing error.
- **Solution:** Perform one of the following steps:
 - Check that the message or ESQL contains a valid value for the element or attribute.
 - Check that the union type on which the element is based contains the correct list of simple types.
 - Consider using an xsi:type attribute to resolve the union explicitly.
 - Consider changing the type of the element in the tree to correspond with one of the union member types.

A union type contains two or more simple types that are derived from the same fundamental type

Procedure

- **Scenario:** A warning is issued during logical validation indicating that a union type contains two or more simple types that are derived from the same fundamental type.
- **Explanation:** The broker does not apply value constraints until the data is in the logical tree. Therefore, it is not possible to choose between two simple types that are derived from the same fundamental type but with different constraints. For example, the broker cannot differentiate between a member type of integer with

a range of 1-10 and another member type of integer with a range of 11-20, therefore it resolves the union to the first member type of integer.

- **Solution:** Ensure that the union type that is identified in the warning does not contain more than one simple type that is derived from the same fundamental type or ensure that the ordering of such member types is as desired.

A list type is based on a union that also contains a list type

Procedure

- **Scenario:** An error message is issued during logical validation, indicating that a list type is based on a union type that includes a list type as one of its member types.
- **Explanation:** Lists of lists are not legal. An item type of a list type cannot be a list type itself nor can it be derived at any level from another list type. Therefore, a list type cannot have an item type of a union type that includes a list type as one of its member types.
- **Solution:** Ensure that any union type specified as an item type for a list type does not include a list type as one of its member types.

A union type contains an enumeration or pattern facet

Procedure

- **Scenario:** An error message is raised during logical validation, indicating that a union type contains an enumeration or pattern facet that is not supported because enumeration and pattern facets must be specified directly on the member type.
- **Explanation:** The broker cannot support the union type facets of pattern and enumeration applied directly to a restriction of a union type. It can support facets directly on the chosen member type only. The Message Definition editor provides support for these facets to enable the import of schemas using direct facets on a restriction of a union type but it issues a warning notifying that they will be ignored by the broker.
- **Solution:** If you want to use the enumeration or pattern facets, specify them directly on the member type and not on the union type itself.

Error message BIP5505 is issued because input data is not valid for the data type

Procedure

- **Scenario:** A data conversion fails while a message is being read or written. Error message BIP5505 is issued, indicating that the input data was not valid for the data type.
- **Solution:** Perform the following checks:
 - Ensure that the bit stream is correctly aligned to the model. If the incoming message is not consistent with the model, the wrong bytes are allocated to fields, and the data presented to a field is unlikely to contain valid data for the logical type.
 - Ensure that the correct encoding and CodedCharSetId are applied to the body of the input message. If the wrong encoding is used, the input message bit stream is subject to byte swapping and changes to the byte order. This change can affect the nature of data that applies to a field and make it unusable.
 - Ensure that an appropriate WebSphere MQ format is applied to the body of the input message. For example, if MQSTR is applied to a bit stream that contains numeric data, character conversions, rather than encodings, are used to translate the bytes. This use can change the meaning of bytes and make

them not valid for the logical data type. If character conversions are used with negative external decimals, consider using the EBCDIC custom option.

- If the error is on a numeric field, ensure that the input data is compatible with the physical format settings. For example, if the field is signed, ensure that a valid sign byte has been passed in, that it corresponds to the orientation of the sign, and that it is included or excluded as appropriate.
- If the input message is found not to be compatible with the MRM message definitions, check the message at the source. If it is sent over channels, check that the conversion settings on the channels are correct.

A deprecation error is issued on an imported .mrp file

Procedure

- **Scenario:** You have imported an .mrp file, and get an error message indicating that complex elements or embedded simple types are deprecated.
- **Explanation:** Complex elements and embedded simple types have no exact equivalent in XML Schema. The closest equivalent to a complex type in XML Schema is to derive a complex type from a simple type. However, such a type is not allowed to contain elements; only attributes are allowed.
- **Solution:** If your complex type contains elements that cannot be modeled as attributes, set the **Mixed content** flag on the complex type. If you need to parse messages that contain mixed content, set the **Mixed content** flag on the parent complex type. The anonymous data that used to be modeled by the complex type or the embedded simple type, then appears as a self-defining node in the parsed message tree.

User trace detects an element length error

Procedure

- **Scenario:** You have defined an MRM message and when you execute the message flow, an error message is written to the user trace indicating that the length of an element is invalid.
- **Explanation:** You have added an element of type string to the complex type making up the message, but you have not defined the length of the string in the instance of the element in its type.
- **Solution:**
 1. Open the message definition file in the Message Definition editor.
 2. In the Outline view, expand the appropriate complex type, then click the element to display its properties in the editor area.
 3. In the Properties Hierarchy, make sure that the element is selected within the CWF physical format under **Physical Properties**.
 4. Specify the properties that this element will have when it is part of the complex type; for example, in the above case, make sure that the **Length Count** value is set.
 5. Save any changes in the Message Definition editor and redeploy the message.

MRM dateTime value has changed after it has been parsed

Procedure

- **Scenario:** You have defined an MRM dateTime element but the value created in the message tree is different from the value that you defined.
- **Explanation:** The parser uses lenient dateTime checking, converting out-of-band data values to the appropriate in-band value.

- **Solution:** See the information about lenient dateTime checking in Message Sets: DateTime as string data with particular reference to how years and fractional seconds are represented.

Related concepts:



Message Sets: Message definition files

A *message definition file* contains the messages, elements, types, and groups which make up a message model within a message set.



Message model objects

An introduction to the objects that make up a message model. Message model objects are defined by XML Schema 1.0, except for Message which is an IBM Integration Bus extension to XML Schema.



Message model objects: simple types

A *simple type* is an abstract definition of an item of data such as a number, a string, or a date.

Related tasks:



Message Sets: Creating a message set

Use the New Message Set wizard to create a message set.



Message Sets: Working with MRM message model objects

Add, configure, and delete objects.



Message Sets: Working with a message definition file

Create, open, and delete a message definition file.



Message Sets: Linking from one message definition file to another

Add an 'include', or an 'import' to the file that you want to reference.



Message Sets: Configuring physical properties

Working with the physical properties of message model objects.

Related reference:



Message set projects and files



Message Definition editor

The Message Definition editor is the default editor provided by the Integration Development perspective for editing message definition (.mxsd) files.



Message Sets: DateTime as string data

You can use a string of pattern letters to specify the dateTime format.

XML Schema Part 0: Primer on the World Wide Web Consortium (W3C) Web site

Resolving problems when using messages

Use the advice given here to help you to resolve common problems that can arise when you use messages.

About this task

Procedure

- “A communication error is issued when you use the enqueue facility” on page 783

- “The enqueue facility is not picking up changes made to a message” on page 784
- “You do not know which header elements affect enqueue” on page 784
- “Enqueue message files are still listed after they have been deleted” on page 784
- “The ESQL transform of an XML message gives unexpected results” on page 784
- “An XML message loses carriage return characters” on page 785
- “The broker is unable to parse an XML message” on page 785
- “Unexpected characters are displayed when using the XSLTransform node on z/OS” on page 785
- “Error message BIP5004 is issued by the XMLNS parser” on page 786
- “Error message BIP5378 is issued by the MRM parser” on page 786
- “Error message BIP5005 is issued by the Compute node” on page 787
- “A message is propagated to the Failure terminal of a TimeoutControl node” on page 787
- “Message processing fails within a TimeoutNotification node” on page 787
- “An MRM CWF message is propagated to the Failure terminal” on page 788
- “Problems with XML attributes” on page 788
- “An MRM XML message exhibits unexpected behavior” on page 789
- “The MRM parser has failed to parse a message because two attributes have the same name” on page 790
- “You encounter problems when messages contain EBCDIC newline characters” on page 790
- “The MIME parser produces a runtime error while parsing a message” on page 790
- “Runtime errors are issued when you write a MIME message from the logical message tree” on page 791
- “Output message has an empty message body” on page 791
- “Output message has an invalid message body indicated by error message BIP5005, BIP5016, or BIP5017” on page 792
- “Error message BIP5651 is issued when receiving a SOAP with Attachments message from a WebSphere Application Server client” on page 793
- “WebSphere Application Server produces an error when receiving a SOAP with Attachments message” on page 793
- “java_lang_StackOverflowError on AIX when processing a message flow that contains Java nodes and uses Java 5” on page 795
- “Problems when using code page translation on HP-UX” on page 796

A communication error is issued when you use the enqueue facility

Procedure

- **Scenario:** You use the enqueue or dequeue tools to put a message on a queue, but an error message is issued indicating that a communication error has occurred with the queue manager name.
- **Explanation:** The WebSphere MQ queue manager has not started.
- **Solution:** Restart the WebSphere MQ queue manager.

The enqueue facility is not picking up changes made to a message

Procedure

- **Scenario:** You are using the IBM Integration Toolkit message enqueue facility to put messages to WebSphere MQ queues. You have updated a message and want to put the message to the queue, but your changes do not seem to have been picked up.
- **Solution:**
 1. Close, then reopen your enqueue file.
 2. Select the message that you want to put to the queue.
 3. Save and close the enqueue file.
 4. Select the menu next to the **Put a message to a queue** icon.
 5. Click **Put message**.
 6. Click the enqueue file in the menu.
 7. Click **Finish**.

This action puts your updated message to the queue.

You do not know which header elements affect enqueue

Procedure

- **Scenario:** When using the Enqueue editor, the accounting token, correlation ID, group ID, and message ID in the message header do not seem to affect behavior.
- **Explanation:** These fields do not affect behavior because they are not serialized properly.

Enqueue message files are still listed after they have been deleted

Procedure

- **Scenario:** Enqueue message files are still listed in the menu after they have been deleted.
- **Explanation:** Deleted enqueue files are not removed from the menu. Nothing happens if you selecting these files.

The ESQL transform of an XML message gives unexpected results

Procedure

- **Scenario:** You have created an XML message that looks like the following content:

```
<?xml version="1.0" standalone="no"?><!DOCTYPE doc
[<!ELEMENT doc (#PCDATA)*>]><doc><I1>100</I1></doc>
```

You apply the ESQL transform:

```
SET OutputRoot.XMLNS.doc.I1 = 112233;
```

This transform generates the XML message (after serialization):

```
<?xml version="1.0" standalone="no"?><!DOCTYPE doc
[<!ELEMENT doc (#PCDATA)*>]<I1>112233</I1>><doc><I1>100</I1></doc>
```

The new value for *I1* has been put inside the DOCTYPE, and has not replaced the value of *100*, as you expected.

- **Explanation:** The XML in your message contains two doc elements:
 - The doctype element

- The `xmlElement` that represents the body of the message

The parser has found the first instance of an element called `doc` and has created a child `I1` with the value `112233`.

- **Solution:** To assign a new value to the element `I1` within the message body element `doc`, explicitly identify the second `doc` element, in the following way:
`SET OutputRoot.XMLNS.(XML.tag)doc.I1 = 112233;`

An XML message loses carriage return characters

Procedure

- **Scenario:** You are parsing an input XML message that contains carriage return or line feed characters, or you are writing an output XML message that contains carriage return line feed characters in an XML element. However, some or all the carriage return characters are not present in the output message.
- **Explanation:** This behavior is correct, as described by the XML specification, and occurs in the XML, XMLNS, or XMLNSC domains.

In XML, the main line separator characters is a line feed character. Carriage return characters are accepted in an XML document, but an XML parser normalizes any carriage return line feed characters into a single-line feed character. For more information, see the latest XML specification at Extensible Markup Language (XML), in particular, Section 2.11 *End of Line Handling*.

You cannot work around this problem by embedding your data into a CDATA section; the XML specification states that a CDATA section is intended only to escape blocks of text that contain characters that could be interpreted as markup. In addition, CDATA sections are not protected from the parser.

You cannot use the `xml:space` attribute to preserve carriage return line feed characters; the XML specification states that normalization of carriage return and line feed characters is done before any other processing is performed.

- **Solution:** XML-compliant data must not rely on the presence of carriage return line feed characters because in XML, the line feed character is only a line feed character and XML-compliant applications or data must be aware that the parser normalizes any carriage return line feed characters.

The broker is unable to parse an XML message

Procedure

- **Scenario:** You receive a large XML file and wrap it in a SOAP envelope to be passed to a .NET web service. The broker is unable to parse the XML message
- **Explanation:** The message that you receive is defined as UTF-8 but it contains UTF-16 characters, such as £. The presence of these characters causes a problem with the parser: the broker is unable to parse the XML message because it contains an invalid character.
- **Solution:** Force the coded character set ID (CCSID) to 1208 instead of the default 437.

Unexpected characters are displayed when using the XSLTransform node on z/OS

Procedure

- **Scenario:** When using the XSLTransform node on z/OS, all the uppercase Os that are in an XML file on the incoming message are changed to underscore characters.
- **Explanation:** The XSLTransform node input message must come in as ASCII for the transform to work correctly. The XSLTransform node does not work with XML or XSL data in EBCDIC code. Java assumes a conversion from EBCDIC 1047. IBM Integration Bus then converts to EBCDIC 500, because the CCSID is

set to 500. EBCDIC 1047 and EBCDIC 500 are similar. Only uppercase O, J, and Z are different. (J and Z are also converted incorrectly.) The conversions leave a string that is unreadable because it is really in ASCII. However, it does convert the O from an EBCDIC 1047 character to an EBCDIC 500 character.

- **Solution:** Change your program either to do a string assignment without any conversions, or specify that the string is in ASCII ISO-8859-1 (CCSID 819).

Error message BIP5004 is issued by the XMLNS parser Procedure

- **Scenario:** Error message BIP5004 is issued, indicating that the XMLNS parser has encountered a problem with an input XML message.
- **Explanation:** This message is issued for a number of reasons. Some of the more commonly occurring scenarios when this message is issued are:
 - You have specified an invalid XML character in the input XML message.
 - You have included binary data in your XML message that, when treated as character data, is invalid.
 - The message has arrived as part of a WebSphere MQ message and the MQMD.CodedCharSetId does not correctly represent the XML text bit stream.
 - You have used characters that are recognized as markup.
- **Solution:**
 - Check that your sending application is sending only valid data. If, however, it is not possible to prevent invalid characters from being included in the XML message, represent it in BLOB domain and use the ESQL REPLACE function to replace or remove the invalid characters. You can then assign the modified bit stream to the required XML parser.

In accordance with XML specification, a CDATA section can be used only to protect characters that would be interpreted as markup. It cannot be used to protect invalid characters or binary data from the XML parser.
 - If the input XML message contains binary data, ensure that the sending application is changed to represent this data as base encoded binary encoded data. If the application cannot be changed, represent the message in the BLOB domain, and extract and replace the binary data before the bit stream is assigned to the required XML parser.
 - Check that the incoming XML message is being represented by the correct MQMD.CodedCharSetId.

Error message BIP5378 is issued by the MRM parser Procedure

- **Scenario:** Error message BIP5378 is issued, which reports a problem with a missing mandatory repeating element in an MRM message.
- **Explanation:** This message indicates that a mandatory repeating element is not present in the message. In previous releases, this condition was reported by error message BIP5374 which now reports only when a mandatory repeating element exists in the message but has the incorrect number of instances.

If you have programmed automated error checking routines, review and change these routines if appropriate.
- **Solution:** Check your message definition to ensure that the element must be mandatory, and repeating. The error message tells you the elements that occur before and after the expected location of the missing element. If the definition is correct, the message has not been created correctly by the sending application, which must be amended.

Error message BIP5005 is issued by the Compute node Procedure

- **Scenario:** You send a simple XML message into a simple message flow. The message is:

```
<doc><I1>100</I1></doc>
```

The Compute node in the message flow contains the following ESQL:
SET OutputRoot.XMLNS.abc = InputBody;

You expect the following output message to be created:

```
<abc><doc><I1>100</I1></doc></abc>
```

The Compute node generates error message BIP5005 and does not implement the ESQL.

- **Explanation:** You are assigning an element of one type (root) to an element of another type (xmlElement). The parser does not do this implicit cast for you.
- **Solution:** You can do the cast yourself in the ESQL to achieve the result that you want, using either of the following two casts:

```
SET OutputRoot.XMLNS.(XML.Element)abc = InputBody;
```

or:

```
SET OutputRoot.XMLNS.(XML.tag)abc = InputBody;
```

A message is propagated to the Failure terminal of a TimeoutControl node Procedure

- **Scenario:** The TimeoutControl node receives a message with invalid, corrupt, or missing timeout information. The message is propagated to the Failure terminal of the TimeoutControl node and an exception list is generated.
- **Explanation:** The following error conditions can cause propagation to the Failure terminal:
 - A timeout request has no Action or no Identifier.
 - A SET request has an Identifier that matches an existing stored SET request for this TimeoutControl node (identified by the Unique Identifier property) and AllowOverwrite of the original request is set to FALSE.
 - A CANCEL request has an Identifier that does not match an existing stored SET request for this TimeoutControl node (identified by the Unique Identifier property).
 - A SET request has a Count of 0 or is less than -1.
 - The StartDate or StartTime are not in the correct format (or one of the understood keywords).
 - The calculated timeout is in the past.
 - The Interval is less than 0, or 0 with a Count of -1.
- **Solution:** Check for one or more of the error conditions listed here, and correct them.

Message processing fails within a TimeoutNotification node Procedure

- **Scenario:** A message is propagated to the Failure or Catch terminal of a TimeoutNotification node.
- **Explanation:** If the processing of a timeout generates an error within the TimeoutNotification node, an exception list is generated and a message is

propagated to the Failure terminal. This action is done under the same transaction, if one is being used. If the Failure terminal is not connected, propagation does not occur.

If an error happens downstream of the TimeoutNotification node after a successful propagation (either to the Out or Failure terminal), the message is propagated to the Catch terminal (all under the same transaction). If the Catch terminal is not connected, or the propagation along the Catch flow fails, the processing of that timeout is rolled back.

- **Solution:** Ensure that the Failure and Catch terminals of your TimeoutNotification node have been connected correctly.

An MRM CWF message is propagated to the Failure terminal Procedure

- **Scenario:** Your MRM CWF message is propagated to a Failure terminal, and generates error messages BIP5285, BIP5125, and BIP5181 or messages BIP5285, BIP5125, and BIP5288.
- **Explanation:** These errors report an inconsistency between the length of the message being processed, and the length of the message that is defined in the message model.
- **Solution:** Ensure that the length of the message as defined in the CWF layer is accurate. Check and correct the definition.

Problems with XML attributes

About this task

XML tags are written where XML attributes are expected, and vice versa.

Procedure

- **Explanation:** The XML domains and the XML Wire Format in the MRM domain have their own representation of XML attributes.
 - XML domains rely on setting a field type of XML.Attribute in the message tree, because they have no model to parse against.
 - For the XML Wire Format in the MRM domain, the message model indicates whether an element is an attribute or a tag, therefore the message tree does not need to reflect whether a field is an attribute or a tag.

Therefore, if fields are copied out of the XMLNS or MRM domains, the fact that fields are attributes is lost. This loss happens if they are copied out to each other, or to another message tree, such as the environment tree.

This problem typically appears in the following situations:

- **Scenario 1:** You are writing an XML message in the MRM domain, and XML tags are being written instead of XML attributes.
- **Solution 1:** Check that your message tree has the same structure and sequence as the message model. If the message tree does not match the message model, the field is written as self-defining, and consequently the XML Render property is not used.
 - Switch on message validation. Validation shows where the message tree and message definition do not match.
 - Alternatively, take a user debug trace of the message flow; BIP5493W messages indicate which fields are being written as self-defining. Use this information to ensure that the message tree matches the model. When you have corrected any discrepancies, attributes are correctly written.

- **Scenario 2:** An MRM message has been copied to an XMLNS domain, and the XML attributes are now written as tags.
- **Solution 2:** Take one of these actions:
 - Serialize the XML message in the MRM domain, for example using the ESQL ASBITSTREAM function, then use the ESQL CREATE PARSE clause to reparse the message using the required XML domain.
 - When copying fields between the MRM domain and XMLNS, copy attribute fields individually, and specifically specify XML.Attribute on the target XML field.
- **Scenario 3:** An XML message has been copied to another message tree, such as Environment. When the message is copied back to the XML message tree, XML attributes are now seen as XML tags.
- **Solution 3:** Serialize the XML message, for example using the ESQL ASBITSTREAM function, then use the ESQL CREATE PARSE clause to reparse the XML message into the required target message tree. See CREATE statement for an example.
- **Scenario 4:** A portion of a non-XML message tree has been detached and attached to an XML tree, and XML tags are now written as XML attributes.
- **Solution 4:** Do not detach and attach portions of message trees that are owned by different parsers. Instead, use a tree copy.
- **Scenario 5:** An XML tag is copied to an XML attribute and the XML attribute is not written in the output message.
- **Solution 5:** When referencing the source XML tag, use the ESQL FIELDVALUE function to copy the specific field value to the target XML attribute field.

An MRM XML message exhibits unexpected behavior Procedure

- **Scenario:** Your message flow is processing a message that you have modeled in the MRM. The message tree has not been created as you expected, the output XML message does not have the expected contents, or the message contents are not being validated. No error message has been issued.
- **Explanation:** Two reasons might cause this problem:
 - **Explanation 1:** The XML physical format settings for a message set contain a property called Root Tag Name. This property defaults to MRM, in order to remain compatible with previous releases of the product. If you have not deleted the contents of this field, the MRM XMLNS parser expects the root tag for all XML messages to be MRM.
Solution 1: Clear this field, or set it to the root tag used by all your XML messages. If you provide a value in this field, the root tag does not need to be modeled in all your message definitions.
 - **Explanation 2:** In order to remain backwards compatible, the broker recognizes the format XML and invokes the XMLNS parser with specific default values. If you have created an XML physical layer for this message with the name XML, the broker uses your definition. However, if you have not created an XML physical layer with this name, but have specified XMLNS as the format, either in the input node or the MQRFH2 header (when the input bit stream is parsed to a message tree), the broker accepts the value specified and passes default values to the parser to create the message tree. Similarly, if you set XML in the Properties folder for the output message in the Compute node, this value is passed to the parser when it creates the message bit stream from the message tree, typically in the output node.

The use of these default values by the parser might result in different content, structure, or both, for either message tree or output message. You can find further information about the action taken by the broker in the user trace log where the following information might be written:

```
XMLWorker::initializeParse file:C:\s000\src\cpi\pwf\xml\xmlworker.cpp
line:126 message:5409.BIPmsgs
No dictionary present have you specified Wire Format 'XML' in error? ,
UserTrace BIP5409E: XML Worker: Wire Format 'XML' specified.
Default MRM XML settings are being used because wire format
identifier 'XML' was specified and not found.
This can be due to an incorrect setting of the wire format
identifier in a message.
```

Solution 2: If you have incorrectly entered the identifier of the format that you have defined, correct your code and try again. If you do not want the default action to be taken, define a physical layer that produces the required results.

The MRM parser has failed to parse a message because two attributes have the same name

Procedure

- **Scenario:** Two attributes in different name spaces have identical names. Error message BIP5117 is issued.
- **Explanation:** The MRM parser has failed to parse the message.
- **Solution:** Modify the attribute names so that they are not identical. This problem is a known limitation with the parser.

You encounter problems when messages contain EBCDIC newline characters

Procedure

- **Scenario:** If your bit stream input message contains EBCDIC newline (NL) characters, problems might arise if your message flow changes the target CCSID to an ASCII CCSID. For example, during conversion from CCSID 1047 (EBCDIC used for z/OS Open Edition) to CCSID 437 (US PC ASCII), an NL character is translated from hex '15' to hex '7F', which is an undefined character. The error occurs because no corresponding code point for the newline character exists in the ASCII code page.
- **Solution:** You can overcome the problem in these cases:
 - On a system where the queue manager uses an ASCII code set, make sure that incoming messages do not contain any EBCDIC NL characters by:
 - Specifying that WebSphere MQ performs the conversion at the input node
 - Setting the queue manager attribute to convert NL to Line Feed (LF)
 - Where the input bit stream is character data, you can use MRM Tagged/Delimited message sets in a Compute node to replace the NL character with the required output.

The MIME parser produces a runtime error while parsing a message

Procedure

- **Scenario:** A MIME message is received by a message flow and produces a runtime error when the message is parsed.
- **Explanation:** The following errors can cause the MIME parser to reject a message:
 - The MIME header is not correctly formatted.

- Either the top-level MIME header block, or a MIME header block for a nested multipart part, does not have a valid Content-Type header.
- The top-level Content-Type has a media type of message.
- The top-level Content-Type has a media type of multipart and no boundary definition.
- A MIME header block is not correctly terminated by a blank line.
- The constituent MIME parts are not correctly separated by boundary lines.
- A boundary parameter value occurs in the content of a MIME part.
- **Solution:** Check the MIME message for one or more of the error conditions listed here, and correct them.

Runtime errors are issued when you write a MIME message from the logical message tree

Procedure

- **Scenario:** You are writing a MIME logical message tree as a bit stream and the parser generates a runtime error.
- **Explanation:** The following errors can cause the MIME parser to reject a logical message tree:
 - The root of the tree is not called MIME.
 - The last child of MIME is not called Parts or Data.
 - The Parts element has a value-only element, and this element is not the first or last child of Parts.
 - The Parts element has children that are not value-only elements or Part children.
 - The Parts element does not have any children called Part.
 - The last child of a Data element is not a BLOB.
- **Solution:** Check the MIME logical message tree for one or more of the error conditions listed here, and correct them.

Output message has an empty message body

Procedure

- **Scenario:** You have unexpectedly encountered an empty message body, or the ASBITSTREAM function has produced a zero length BLOB.
- **Explanation:** This error can happen for the following reasons:
 - You have created a message tree folder in a user-defined node but have not associated it with an owning parser. An owning parser is not associated with the message tree if you have created standard elements by using code similar or equivalent to:


```
MbElement createElementAfter(int)
MbElement createElementAfter(int, String, Object)
MbElement createElementAsFirstChild(int)
MbElement createElementAsFirstChild(int, String, Object)
MbElement createElementAsLastChild(int)
MbElement createElementAsLastChild(int, String, Object)
MbElement createElementBefore(int)
MbElement createElementBefore(int, String, Object)
```
 - You have used ESQL to create a message tree folder by using ESQL CREATE but without setting an owning parser for it. You might have used code similar or equivalent to:

```
CALL CopyMessageHeaders();
DECLARE outRef REFERENCE TO OutputRoot;
CREATE LASTCHILD OF outRef AS outRef NAME 'BLOB';
CREATE LASTCHILD OF outRef NAME 'BLOB' VALUE X'01';
```

or the outRef reference variable was passed to an ESQL function or procedure that contained similar CREATE statements. You have not specified an owning parser by using the DOMAIN clause on the CREATE statement.

- An MRM message tree has been constructed, then only part of it has been passed, by specifying a subfolder or field, to the ASBITSTREAM function with the parser mode option set to RootBitStream. This combination is not valid, and results in a zero length BLOB.
- You have copied a message tree, or part of a message tree, to a folder and the owning parser association is not maintained.
- **Solution:** Depending on the reason for the empty message body or zero length BLOB, ensure that:

- When you create a message tree folder in a user-defined node, associate an owning parser with it. Use code similar or equivalent to:

```
createElementAfter(String parserName)
  createElementAsFirstChild(String parserName)
  createElementAsLastChild(String parserName)
  createElementBefore(String parserName)
```

- When you use ESQL CREATE to create a message tree folder, use the DOMAIN clause to associate an owning parser with the message tree, for example:

```
CALL CopyMessageHeaders();
DECLARE outRef REFERENCE TO OutputRoot;
CREATE LASTCHILD OF outRef AS outRef DOMAIN 'BLOB' NAME 'BLOB';
CREATE LASTCHILD OF outRef NAME 'BLOB' VALUE X'01';
```

This code creates a BLOB folder that has the BLOB parser associated with it.

- When copying a message tree, or part of a message tree, ensure that the owning parser association is maintained, by first serializing, then reparsing the message into the appropriate message tree. An example scenario is where you have created a field:

```
SET Environment.Variables.myMsg = InputRoot.XMLNS;
```

Now you must pass it to the ASBITSTREAM function. Unless you use ESQL similar or equivalent to:

```
DECLARE xmlMsgBlob BLOB
  ASBITSTREAM(InputRoot.XMLNS, InputRoot.MQMD.Encoding, InputRoot.MQMD.CodedCharSetId);
CREATE LASTCHILD OF Environment.Variables.myMsg DOMAIN('XMLNS')
  PARSE(xmlMsgBlob,
    InputRoot.MQMD.Encoding,
    InputRoot.MQMD.CodedCharSetId);
```

the result is a zero length bit stream.

Output message has an invalid message body indicated by error message BIP5005, BIP5016, or BIP5017 Procedure

- **Scenario:** You have unexpectedly encountered a multi-root message body or a message without any root.

- **Explanation:** This error can occur when you have created an XML message tree folder with multiple roots or no root at all by:
 - Using a user-defined node
 - Using an MQGet node
 - Using ESQL
- **Solution:** Ensure that the final output message tree has one, and only one, XML root node.

Error message BIP5651 is issued when receiving a SOAP with Attachments message from a WebSphere Application Server client

Procedure

- **Scenario:** When a WebSphere Application Server client sends a SOAP with Attachments message to the broker over JMS, error message BIP5651 is issued stating that no valid Content-Type header has been found.
- **Explanation:** When a WebSphere Application Server client sends a SOAP with Attachments message to the broker over JMS, the MIME Content-Type value appears in the MQRFH2 header and not in the MIME message body.
- **Solution:** To solve this problem, the Content-Type value needs to be copied from the MQRFH2 header to the beginning of the message as a MIME header before the message is parsed. The following ESQL adds the Content-Type value to the beginning of the WebSphere Application Server message, then invokes the MIME parser on the result.

```
create procedure parseWAS_JMS(IN InputMessage reference,IN OutputMessage reference)
/*****
* convert a WAS/JMS message to the correct format for the MIME parser
*****/
begin
  -- get the data as a BLOB
  declare body BLOB InputMessage.BLOB.BLOB;

  -- get the Content-Type value from the RFH2 header. Content-Type is the only
  -- header which is critical for the MIME parser, but the same approach can be
  -- used for any MIME headers which have been stored under the RFH2 header.
  declare contentType char InputMessage.MQRFH2.usr.contentType;

  -- add the contentType to the bit stream as part of an RFC822 header block
  set body = cast(('Content-Type: '||contentType) as blob ccsid 819)||x'0d0a0d0a'||body;

  -- invoke MIME parser on the modified bit stream
  CREATE LASTCHILD OF OutputMessage DOMAIN('MIME') PARSE(body);
end;
```

A message flow can take in the JMS message in the BLOB domain, and call the procedure shown here from a Compute node. The procedure can be called by using the following ESQL from a Compute node:

```
CALL CopyMessageHeaders();           -- standard procedure to copy headers
CALL parseWAS_JMS(InputRoot, OutputRoot); -- parse the 'body' as MIME
```

WebSphere Application Server produces an error when receiving a SOAP with Attachments message

Procedure

- **Scenario:** When sending a SOAP with Attachments message to a WebSphere Application Server client using JMS, an error is generated stating that the bit stream contains unexpected characters.

- **Solution:** When the broker sends a SOAP with Attachments message to WebSphere Application Server over JMS, the MIME Content-Type value must appear in the MQRFH2 header and not in the body of the message. The following procedure removes any MIME style headers from the front of the message bit stream and adds the Content-Type value to the MQRFH2 header. The supplied boundary value allows you to locate the start of the multipart MIME content.

```
create procedure writeWAS_JMS(IN OutputTree reference,IN boundary char)
/*****
* Serialise a MIME tree as normal, but then strip off any initial headers
* and save the Content-Type value in the RFH2 header as expected by WAS/JMS.
* Note: boundary - must be supplied with the leading hyphen pair
*****/
begin
  -- convert MIME subtree to BLOB
  declare body BLOB asbitstream(OutputTree.MIME);

  -- locate first occurrence of boundary and discard any data before this
  declare firstBoundary integer;
  set firstBoundary = position (cast(boundary as blob ccsid 819) in body);
  set body = substring(body from firstBoundary);

  -- save the MIME Content-Type value in the RFH2 header. Any other MIME
  -- headers which need to be preserved in the RFH2 header can be handled
  -- in the same way
  set OutputTree.MQRFH2.usr."contentType" = OutputTree.MIME."Content-Type";

  -- clear the MIME tree and create a new BLOB child for the modified body
  set OutputTree.MIME = null;
  CREATE LASTCHILD OF OutputTree DOMAIN('BLOB')PARSE(body);
end
```

Before calling this procedure, the message flow needs to be able to obtain the value of the boundary. This value might be available only within a Content-type header. The following procedure allows you to extract the Boundary value:

```
create procedure getBoundary(IN ct reference,OUT boundary char)
/*****
* return value of the boundary parameter from a Content-Type value
*****/
begin
  declare boundaryStart integer;
  declare boundaryEnd integer;

  set boundaryStart = position('boundary=' in ct) + 9;
  set boundaryEnd = position('; ' in ct from boundaryStart);
  if (boundaryStart <> 0) then
    if (boundaryEnd <> 0) then
      set boundary = substring(ct from boundaryStart for boundaryEnd-boundaryStart);
    else
      set boundary = substring(ct from boundaryStart);
    end if;
  end if;
end;
```

A Compute node can invoke these procedures for sending a MIME message using the following ESQL:

```
SET OutputRoot = InputRoot;
```

```
declare boundary char;
```

```
CALL getBoundary(OutputRoot.Properties.ContentType, boundary);
```

```
CALL writeWAS_JMS(OutputRoot,boundary);
```

java_lang_StackOverflowError on AIX when processing a message flow that contains Java nodes and uses Java 5 Procedure

- **Scenario:** You get an abend on AIX when processing a message flow that contains Java nodes and uses Java 5. The abend file shows that there was an abend which indicates a segmentation fault, but a check of the stderr file shows a stack overflow in the JVM:

```
Exception in thread "Thread-15" java/lang/StackOverflowError: operating system stack overflow
  at com/ibm/broker/plugin/MbOutputTerminal._propagate (Native Method)
  at com/ibm/broker/plugin/MbOutputTerminal.propagate (MbOutputTerminal.java:103)
  at com/ibm/xsl/mqsi/XMLTransformNode.evaluate (XMLTransformNode.java:1002)
  at com/ibm/broker/plugin/MbNode.evaluate (MbNode.java:1434)
  at com/ibm/broker/plugin/MbOutputTerminal._propagate (Native Method)
  at com/ibm/broker/plugin/MbOutputTerminal.propagate (MbOutputTerminal.java:103)
  at com/ibm/xsl/mqsi/XMLTransformNode.evaluate (XMLTransformNode.java:1002)
  at com/ibm/broker/plugin/MbNode.evaluate (MbNode.java:1434)
  at com/ibm/broker/plugin/MbOutputTerminal._propagate (Native Method)
  at com/ibm/broker/plugin/MbOutputTerminal.propagate (MbOutputTerminal.java:103)
  at com/ibm/xsl/mqsi/XMLTransformNode.evaluate (XMLTransformNode.java:1002)
  at com/ibm/broker/plugin/MbNode.evaluate (MbNode.java:1434)
```

- **Explanation:** Java 5 has a parameter to adjust the stack size for Java threads. The default operating system stack size for Java 5 is only 256 KB. In certain message flows (for example, flows that contain Java user-defined nodes or XMLT nodes) this size might not be sufficient, and so you see a stack overflow error in the stderr file. Use the JVM option `-Xmso` to adjust the operating system stack for Java. You can display information about the stack by using the following command:

```
export MQSIJVERBOSE=-verbose:stack,sizes
```

This command creates in the stderr file on startup an entry that has the following content, or similar:

```
-Xmca32K      RAM class segment increment
-Xmco128K     ROM class segment increment
-Xmns0K       initial new space size
-Xmnx0K       maximum new space size
-Xms125000K   initial memory size
-Xmos125000K  initial old space size
-Xmox250000K  maximum old space size
-Xmx250000K   memory maximum
-Xmr16K       remembered set size
-Xlp0K        large page size
              available large page sizes: 4K 16M
-Xmso256K     OS thread stack size
-Xiss2K       java thread stack initial size
-Xssi16K      java thread stack increment
-Xss256K      java thread stack maximum size
-Xscmx16M     shared class cache size
```

Note: The stack size defaults to 256K.

- **Solution:**
 1. Issue the following command to set the operating system stack size for Java to 2 MB:

```
export IBM_JAVA_OPTIONS=-Xmso2m
```
 2. Restart the broker.

Problems when using code page translation on HP-UX Procedure

- **Scenario:** You experience code page translation problems on HP-UX.
- **Solution:** Check the WebSphere MQ queue manager attribute *CodedCharSetID*. The default value for this attribute is 1051. Change this value to 819 for queue managers that host IBM Integration Bus components.

Related concepts:

XML parsers and domains

You can use XML domains to parse and write messages that conform to the W3C XML standard.

MIME messages

A MIME message consists of both data and metadata. MIME metadata consists of HTTP-style headers and MIME boundary delimiters.

MIME tree details

A MIME message is represented in the broker as a logical tree. When it writes a message, the MIME parser creates a message bit stream by using the logical message tree.

Related tasks:

Message Sets: Configuring Custom Wire Format (CWF) properties: Message sets

Configure the Custom Wire Format (CWF) properties of a message set using the Message Set editor.

Related reference:

Compute node

Use the Compute node to construct one or more new output messages.

MQGet node

Use the MQGet node to receive messages from clients that connect to the broker by using the WebSphere MQ Enterprise Transport, and the MQI and AMI application programming interfaces.

TimeoutControl node

Use the TimeoutControl node to process an input message that contains a timeout request.

TimeoutNotification node

Use the TimeoutNotification node to manage timeout-dependent message flows.

User-defined nodes

You can define your own nodes to use in IBM Integration Bus message flows.

Resolving problems when you are writing business rules

Use the advice that is given here to help you to resolve problems that can arise when you are writing business rules.

Resolving problems when you are writing business rules Procedure

- **Scenario:** You are retrieving rules at run time from an IBM Operational Decision Manager repository but rules are not being updated automatically.

- **Explanation:** This problem can be caused when the TCP/IP port that you configure for IBM Integration Bus to communicate with IBM Operational Decision Manager is used by another application. For example, the default IBM Operational Decision Manager TCP/IP port is 1883, which is also the default TCP/IP port for WebSphere MQ Telemetry.
- **Solution:** Check that the TCP/IP port that is configured for IBM Operational Decision Manager is not used by another application. If a conflict exists, change the port number for the JDBCProviders configurable service that defines the connection to the IBM Operational Decision Manager repository.

Related concepts:


 Business rules

A business rule is a required operation that applies to a specific set of business conditions. For example, you can create a business rule that offers a discount to customers who spend more than a certain amount. The business rule can be modified in the future if the business climate changes and the amount of discount must change.

Related tasks:

 Developing business rules

Build business rules with natural language to control business logic.

 Retrieving business rules at run time from an IBM Operational Decision Manager repository

Use a configurable service to retrieve business rules that are created in IBM Operational Decision Manager.

Related reference:

 Configurable services properties

The supplied configurable services, and the configurable services that you create, are defined by their names and properties. You can use the supplied services.

Resolving problems when you use the IBM Integration Toolkit

Use the advice given here to help you to resolve common problems that might occur you use the IBM Integration Toolkit.

About this task

Resolving problems that occur when connecting:

- “Your broker is not recognized by the IBM Integration Toolkit” on page 799
- “You cannot connect to the broker from the IBM Integration Toolkit or command line” on page 799
- “You cannot connect to a migrated broker” on page 799
- “Connection to the broker is slow” on page 800

Resolving error messages that occur when using the IBM Integration Toolkit:

- “Errors occur while you are using the IBM Integration Toolkit” on page 801
- “An error is issued when you access the help system” on page 801
- “The IBM Integration Toolkit displays an error message after the error has been fixed” on page 802
- “The message Unable to create part: *filename.extension* is issued” on page 802

Resolving problems relating to the appearance of the workbench:

- “A view is missing from the workbench” on page 803
- “You cannot close a message dialog box” on page 803
- “You want to filter entries in the Problems view” on page 803
- “Your message flow and message set projects have changed their appearance” on page 803

Resolving non-specific problems when using the workbench:

- “Main menu entries missing on Linux on x86” on page 804
- “Editors do not update automatically when the same file is open in multiple windows” on page 804
- “Deleting or closing a project takes a long time” on page 804
- “You are experiencing poor performance when working with large or complex projects” on page 804
- “You do not know how to return to the welcome page” on page 805

Related concepts:



IBM Integration Toolkit perspectives

A perspective is a group of views and editors that shows various aspects of the resources in the IBM Integration Toolkit.



Editors

An editor is a component of the IBM Integration Toolkit. Editors are typically used to edit or browse resources, which are the files, folders, and projects that exist in the workbench.

Related tasks:



Changing IBM Integration Toolkit preferences

The IBM Integration Toolkit has a large number of preferences that you can change to suit your requirements. Some of these are specific to the product plug-ins that you have installed, including those for IBM Integration Bus. Others control more general options, such as the colors and fonts in which information is displayed.

“Viewing the Eclipse error log” on page 844

The Eclipse error log captures internal errors that are caused by the operating system or your code.

“Using trace” on page 845

You can use different types of trace to help you with problem determination and troubleshooting.

Related reference:



Resource types in the IBM Integration Toolkit

Links to information on projects and resource files.



Integration Development perspective

The Integration Development perspective is the default perspective that is displayed when you start the IBM Integration Toolkit.



IBM Integration Bus logs

IBM Integration Bus writes information to a number of product-specific logs to report the results of actions that you take.

Resolving problems that occur when connecting the IBM Integration Toolkit and a broker

Use the advice given here to help you to resolve common problems that can arise when you connect to a broker.

Your broker is not recognized by the IBM Integration Toolkit: Procedure

- **Scenario:** Your broker is not recognized by the IBM Integration Toolkit.
- **Explanation:** Broker names in the workbench are case sensitive.
- **Solution:** When you identify a broker in the IBM Integration Toolkit, make sure that you use the same case that was used when it was created. On some operating systems, the name of the broker might be folded to uppercase when it was created, even though you entered its name in lowercase.

You cannot connect to the broker from the IBM Integration Toolkit or command line: Procedure

- **Scenario:** Error messages are issued when you try to connect to the broker from the IBM Integration Toolkit or the command line.
- **Solution:** The following table shows the checks to carry out when an error message is issued in the IBM Integration Toolkit, or returned to the command line, when you try to connect to the broker:

Error message		Actions
IBM Integration Toolkit	Command line	
BIP0914E	BIP1046E with WebSphere MQ reason code 2059	Check that the broker queue manager and listener are running, and that the correct port is specified for the queue manager. (The WebSphere MQ documentation describes the WebSphere MQ return codes.)
BIP0889E	BIP1047E	Check that the broker is running. Check the system event log to ensure that the broker is available for use, and correct all errors that are shown.
BIP0915E	BIP1046E with WebSphere MQ reason code 2035	Check that the IBM Integration Toolkit local user ID is created on the broker system. If it is not, create it. Check if the IBM Integration Toolkit local user ID that is created on the broker system is authorized to connect to the broker queue manager. <ul style="list-style-type: none"> • Use the WebSphere MQ dspmqaut command to determine what WebSphere MQ authorities the user has. • If necessary, add the IBM Integration Toolkit local user ID to the mqm group on the broker system.

You cannot connect to a migrated broker: Procedure

- **Scenario:** You cannot connect from the IBM Integration Toolkit, the IBM Integration Toolkit, or an IBM Integration Toolkit application to a broker that you have migrated to Version 9.0.
- **Explanation:** The connection is failing because the WebSphere MQ channel does not exist. When you migrate a broker to Version 9.0, the default server connection channel SYSTEM.BKR.CONFIG is created if it does not exist.

However, if the broker shared a queue manager with a Configuration Manager, this channel is deleted when you delete the Configuration Manager.

- **Solution:** Re-create the server connection channel SYSTEM.BKR.CONFIG on the queue manager.

You cannot connect to a remote broker:

Procedure

- **Scenario:** You have correctly configured your IBM Integration Toolkit to connect to a remote broker, but attempting to connect to the remote broker by using the IBM Integration Toolkit results in error MQRC 2059. You can connect to your remote broker from the same computer without error by using the IBM Integration Explorer, or by completing the following steps:
 1. Set the following environment variable:
 - On Windows: set MQSERVER=ChannelName/TCP/server-address(port)
 - On Linux: export MQSERVER=ChannelName/TCP/'server-address(port)'
 2. Open an IBM Integration Bus command prompt, and enter the following test command:

```
amqsputc queueName qmgrName
```
- **Explanation:** The network configuration of your IBM Integration Toolkit installation is incorrect.
- **Solution:**
 1. In the IBM Integration Toolkit, click **Window > Preferences > General > Network Connection**. The Preferences window opens, and the Network Connections preferences are displayed.
 2. From the Active Provider list, select **Direct**. For more information about the Active Provider setting, see the "Network Connections" topic in the Eclipse information center. You can now connect to your remote broker by using the IBM Integration Toolkit.

Connection to the broker is slow:

Procedure

- **Scenario:** Connection to the broker is slow and you cannot complete other actions in the IBM Integration Toolkit when you request the following operations:
 - Creating a broker
 - Creating an integration server
 - Deploying a broker archive file, using the Deploy a BAR File wizard
 - Opening the Administration log editor
- **Solution:** Click **Cancel** on the progress monitor dialog box to cancel the in-progress connection. Connection to the broker is canceled, so that you can perform other actions in the IBM Integration Toolkit. You can then resubmit the canceled operation.

Related concepts:



IBM Integration Toolkit perspectives

A perspective is a group of views and editors that shows various aspects of the resources in the IBM Integration Toolkit.



Editors

An editor is a component of the IBM Integration Toolkit. Editors are typically used to edit or browse resources, which are the files, folders, and projects that exist in the workbench.

Related tasks:

“Resolving problems when you use the IBM Integration Toolkit” on page 797
Use the advice given here to help you to resolve common problems that might occur you use the IBM Integration Toolkit.



Changing IBM Integration Toolkit preferences

The IBM Integration Toolkit has a large number of preferences that you can change to suit your requirements. Some of these are specific to the product plug-ins that you have installed, including those for IBM Integration Bus. Others control more general options, such as the colors and fonts in which information is displayed.

“Viewing the Eclipse error log” on page 844

The Eclipse error log captures internal errors that are caused by the operating system or your code.

“Using trace” on page 845

You can use different types of trace to help you with problem determination and troubleshooting.

Related reference:



Resource types in the IBM Integration Toolkit
Links to information on projects and resource files.



Integration Development perspective

The Integration Development perspective is the default perspective that is displayed when you start the IBM Integration Toolkit.



IBM Integration Bus logs

IBM Integration Bus writes information to a number of product-specific logs to report the results of actions that you take.

Resolving error messages that occur when using the IBM Integration Toolkit

Use the advice given here to help you to resolve common problems that can occur when you use the IBM Integration Toolkit.

Errors occur while you are using the IBM Integration Toolkit:

Procedure

- **Scenario:** An error has occurred while you are using the IBM Integration Toolkit, and you want information to help you to diagnose the problem.
- **Solution:** Some errors that occur in the IBM Integration Toolkit are logged in the .log file in your workspace\metadata directory. You can view this log by switching to the Plug-in Development perspective, and clicking the Error Log tab in the lower-right pane. The log shows in which plug-in the error occurred, and gives further information.

You can also use trace to try and determine the cause of your problem. See “Using trace” on page 845 for more information about tracing.

An error is issued when you access the help system:

Procedure

- **Scenario:** You are accessing the help system through the IBM Integration Toolkit, and an error message is issued stating that the Web page that you requested is not available offline.
- **Explanation:** This error might occur if you previously had a connection to a Web-based version of the help system and lost it, or if you have **Work Offline** selected in your Internet Explorer options.

- **Solution:** Click **Connect** to load the help system.

The IBM Integration Toolkit displays an error message after the error has been fixed:

Procedure

- **Scenario:** The IBM Integration Toolkit is in an inconsistent state, or displays an error message after the error has been fixed
- **Solution:** Clean the workspace:
 1. From the IBM Integration Toolkit, click **Project > Clean**.
 2. Click **Clean all projects** and **Finish**.
This action cleans the whole workspace of any internal files, which are then re-created so that none of your data is lost.

The message Unable to create part: filename.extension is issued:

Procedure

- **Scenario:** You open an editor in the IBM Integration Toolkit, and an error message is raised stating that a file cannot be created. This problem is caused by one of following situations:
- **Explanation 1:** The given file is not associated with a recognized editor.
 - **Solution 1:** Right-click the file and choose the default editor to open it, or choose another editor if the default editor cannot open it.
- **Explanation 2:** The given file contains syntax errors and cannot be loaded into the chosen editor. However, if you then try to open a valid file, you get the same error message repeatedly.
 - **Solution 2:** Restart the IBM Integration Toolkit; do not load the file into an editor again until you have fixed the syntax errors.

:

Related concepts:



IBM Integration Toolkit perspectives

A perspective is a group of views and editors that shows various aspects of the resources in the IBM Integration Toolkit.



Editors

An editor is a component of the IBM Integration Toolkit. Editors are typically used to edit or browse resources, which are the files, folders, and projects that exist in the workbench.

Related tasks:

“Resolving problems when you use the IBM Integration Toolkit” on page 797
Use the advice given here to help you to resolve common problems that might occur you use the IBM Integration Toolkit.



Changing IBM Integration Toolkit preferences

The IBM Integration Toolkit has a large number of preferences that you can change to suit your requirements. Some of these are specific to the product plug-ins that you have installed, including those for IBM Integration Bus. Others control more general options, such as the colors and fonts in which information is displayed.

“Viewing the Eclipse error log” on page 844


The Eclipse error log captures internal errors that are caused by the operating system or your code.


“Using trace” on page 845


You can use different types of trace to help you with problem determination and

troubleshooting.

Related reference:

 Resource types in the IBM Integration Toolkit
Links to information on projects and resource files.

 Integration Development perspective
The Integration Development perspective is the default perspective that is displayed when you start the IBM Integration Toolkit.

 IBM Integration Bus logs
IBM Integration Bus writes information to a number of product-specific logs to report the results of actions that you take.

Resolving problems relating to the appearance of the workbench

Use the advice given here to help you to resolve common problems that relate to the appearance of the IBM Integration Toolkit.

A view is missing from the workbench:

Procedure

- **Scenario:** A view you are expecting to be displayed in the workbench is not visible, or you have closed a view.
- **Explanation:** The perspective has changed since it was first created, and the default views are no longer visible.
- **Solution:** To restore the default views in a perspective you can reset the perspective. Click **Window > Reset perspective**.

You cannot close a message dialog box:

Procedure

- **Scenario:** You have opened a menu, and a message dialog box tells you that previous changes made have been processed successfully by the broker. You cannot close the dialog.
- **Solution:** If you cannot close the dialog box by clicking **OK**, press Esc.

You want to filter entries in the Problems view:

Procedure

- **Scenario:** The Problems view has many entries, and it is difficult to find the entry that you want.
- **Solution:** At the top of the Problems View, click the icon with the arrows pointing to the right. This action opens a dialog box, in which you can tailor your selections to display only a subset of the entries in the view. For example, you can select only those entries for the project that you have selected.

Your message flow and message set projects have changed their appearance:

Procedure

- **Scenario:** You have created a new simple project in the IBM Integration Toolkit and now all your message flow and message set projects look different.
- **Explanation:** When you create a new simple project, the IBM Integration Toolkit switches automatically to the Integration Development perspective.
- **Solution:** To return to the previous view, switch to the perspective with which you were working..

Resolving non-specific problems when using the IBM Integration Toolkit

Use the advice given here to help you to resolve some common problems that can occur when you use the IBM Integration Toolkit that are not dealt with in previous categories.

Main menu entries missing on Linux on x86:

Procedure

- **Scenario:** The main menu on Linux on x86 no longer contains items that relate to the IBM Integration Toolkit.
- **Explanation:** If you have installed the IBM Integration Toolkit in more than one package group, and have now removed one of those installations, a known restriction causes the main menu items to be removed when the component is uninstalled.
- **Solution:** Invoke the IBM Integration Toolkit from the command line. Navigate to the installation directory for the package group in which you have installed the IBM Integration Toolkit, and enter the following command:

```
./eclipse -product com.ibm.etools.msgbroker.tooling.ide
```

Editors do not update automatically when the same file is open in multiple windows:

Procedure

- **Scenario:** You are working in the Integration Development perspective, and are using the associated editor to work with one or more resources; for example, you are editing a message flow in the message flow editor or an ESQL module in the ESQL editor. You have clicked **Window > New Window** to create a second Eclipse view, and have opened the same resource in the second window. Changes that you make to the resource in the first editor window are not reflected in the second editor window.
- **Explanation:** The IBM Integration Toolkit editors do not automatically update multiple windows in which you have opened the same resource.
- **Solution:** Save the contents of the resource file in the first editor window, then close and reopen additional windows. The reopened windows reflect the updated content.

Deleting or closing a project takes a long time:

Procedure

- **Scenario:** Deleting or closing a project to save memory takes a long time.
- **Explanation:** If a project is referenced by other projects, removing that project requires all the other projects, and the projects that refer to them recursively, to be built fully. This process occurs to keep the content-assist and validation models current.
- **Solution:** To keep a project open in the workspace requires very little memory, therefore you do not need to close or delete projects.

You are experiencing poor performance when working with large or complex projects:

Procedure

- **Scenario:** You are experiencing poor performance in the IBM Integration Toolkit when working with large or complex projects.
- **Explanation:** Frequent project changes, such as adding and removing projects, or using **Project > Clean**, use large amounts of memory because of the size and number of files and the connections between them.

- **Solution:** Increase your system memory.

You do not know how to return to the welcome page:

Procedure

- **Scenario:** You do not know how to return to the welcome page that was displayed in the IBM Integration Toolkit when you first started using it.
- **Solution:** To open the welcome page:
 1. From the **Help** menu, select **Welcome**. If only one welcome page is available, it is displayed. If more than one is available, a list is displayed.
 2. Select the welcome page that you want, for example, IBM Integration Bus IBM Integration Toolkit.
 3. Click **OK**.

:

Related concepts:

IBM Integration Toolkit perspectives

A perspective is a group of views and editors that shows various aspects of the resources in the IBM Integration Toolkit.

Editors

An editor is a component of the IBM Integration Toolkit. Editors are typically used to edit or browse resources, which are the files, folders, and projects that exist in the workbench.

Related tasks:

“Resolving problems when you use the IBM Integration Toolkit” on page 797

Use the advice given here to help you to resolve common problems that might occur you use the IBM Integration Toolkit.

Changing IBM Integration Toolkit preferences

The IBM Integration Toolkit has a large number of preferences that you can change to suit your requirements. Some of these are specific to the product plug-ins that you have installed, including those for IBM Integration Bus. Others control more general options, such as the colors and fonts in which information is displayed.


“Viewing the Eclipse error log” on page 844


The Eclipse error log captures internal errors that are caused by the operating system or your code.


“Using trace” on page 845

You can use different types of trace to help you with problem determination and troubleshooting.

Related reference:

 Resource types in the IBM Integration Toolkit
Links to information on projects and resource files.

 Integration Development perspective
The Integration Development perspective is the default perspective that is displayed when you start the IBM Integration Toolkit.

 IBM Integration Bus logs
IBM Integration Bus writes information to a number of product-specific logs to report the results of actions that you take.

Resolving problems when using the IBM Integration Explorer

Work through the advice provided to help you to deal with problems that can arise when you are using the IBM Integration Explorer.

About this task

- “Tracing problems with accounting and statistics”
- “The Integration Nodes folder is missing”
- “You cannot open the information center on Red Hat Enterprise Linux”
- “Accounting and statistics do not work for a remote integration node” on page 807
- “SYSTEM.DEF.SVRCONN error” on page 807

Tracing problems with accounting and statistics

Procedure

- **Scenario:** When you view accounting and statistics data using the IBM Integration Explorer you encounter a problem, and you want to collect trace.
- **Explanation:** The IBM Integration Explorer uses the Java Message Service (JMS) to retrieve the accounting and statistics data from the integration node (broker). To trace problems with this connection, you must turn on JMS tracing in WebSphere MQ.
- **Solution:** See the WebSphere MQ documentation for information about how to turn on JMS tracing. You can find this information in the following part of the information center: **Using Java > Using WebSphere MQ classes for JMS > Solving problems > Tracing programs.**

The Integration Nodes folder is missing

Procedure

- **Scenario:** You have opened the WebSphere MQ Explorer to work with integration nodes (brokers) and associated resources, but you cannot see the Integration Nodes folder.
- **Explanation:** The IBM Integration Explorer is an extension to the WebSphere MQ Explorer, and provides the Integration Nodes folder and Broker Archive Files folder, and related actions. You must install the IBM Integration Explorer component on all computers on which you want to use the WebSphere MQ Explorer to manage IBM Integration Bus resources, including integration nodes, integration servers and BAR files.
- **Solution:** Close your MQ Explorer session, follow the instructions to install the IBM Integration Explorer, then start the MQ Explorer again.

You cannot open the information center on Red Hat Enterprise Linux

Procedure

- **Scenario:** You are using the IBM Integration Explorer on Red Hat Enterprise Linux Server 5.2 or Red Hat Enterprise Linux Server 5.3 but you cannot open the information center from inside IBM Integration Explorer.
- **Solution:** Set the environment variable `MOZILLA_FIVE_PATH` then run the `strmqcfg` command. For more information, see the question about running the SWT browser inside Eclipse at: Eclipse SWT FAQ

Accounting and statistics do not work for a remote integration node

Procedure

- **Scenario:** You try to view statistics, but instead see a message box:
Could not connect to integration node *nodeName*.
Is MQ configured for JMS?
- **Explanation:** Java Message Service (JMS) has not been correctly configured. Either JMS has not been enabled for the queue manager, or you do not have access to the channel that the integration node uses for JMS activity. Check the log for more details. Note: In WebSphere MQ Version 7.0, 7.0.1, 7.1 and 7.5 onwards JMS is automatically configured, so the problem could be due to the security settings on the publish/subscribe queues. Verify that IBM Integration Explorer can communicate with the queue manager's publish/subscribe queues. For more information on WebSphere MQ publish/subscribe security, see IBM Education Assistant module: WebSphere MQ V7 Publish/subscribe security.
- **Solution:** If the log shows that JMS has not been correctly set up, take the following steps:
 1. In an IBM Integration Bus command console, change to the *IBExplorerInstallLocation\ eclipse\plugins\ com.ibm.etools.wmadmin.broker.stats_releaseNumber* directory.
 2. Run the following command:

```
runmqsc queuemanager < setup_jms.mqsc
```

If the log shows that you do not have permission to access the channel:

1. Check which channel is being used by viewing the Statistics panel for the integration node in IBM Integration Explorer. The channel is normally SYSTEM.DEF.SVRCONN.
2. In an IBM Integration Bus command console, issue the following command to allow any user on your integration node machine, who is a member of the mqm and mqbrkrs groups, to run with the permissions of the integration node user:

```
alter chl (SYSTEM.DEF.SVRCONN) CHLTYPE(SVRCONN) MCAUSER(nodeUser)
```

Note: Setting SYSTEM.DEF.SVRCONN MCAUSER(*nodeUser*) when the integration node user is a member of the mqm group requires security protocols, such as Transport Layer Security (TLS), Secure Sockets Layer (SSL), channel authentication security, or a security exit, to be set up. For more information on security, see “Security overview” on page 203.

SYSTEM.DEF.SVRCONN error:

Procedure

- **Scenario:** When you try to connect to an integration node to view accounting and statistical information, an error message reports that IBM Integration Explorer cannot connect to SYSTEM.DEF.SVRCONN.
- **Explanation:** This error might indicate that the SHARECNV parameter on the SVRCONN channel of the queue manager is not compatible with the current WebSphere MQ level of JMS.
- **Solution:**
 1. For more information about the error, turn on service trace; see “Changing trace settings from the IBM Integration Explorer” on page 862.
 2. In WebSphere MQ Explorer, set **Sharing Conversations** (SHARECNV) to at least 1.

Related concepts:



IBM Integration Explorer

The IBM Integration Explorer is a graphical user interface based on the Eclipse platform for administering your integration nodes (brokers).

Related tasks:



Installing IBM Integration Explorer

To use IBM Integration Explorer only, without installing the complete IBM Integration Toolkit, use the IBM Integration Explorer installation wizard to install the IBM Integration Explorer.



Configuring integration nodes using the IBM Integration Explorer

Configure your local and remote integration nodes (brokers) by using the IBM Integration Explorer.

“Viewing accounting and statistics data in the IBM Integration Explorer” on page 497

You can use the Broker Statistics and Broker Statistics Graph views in the IBM Integration Explorer to view snapshot accounting and statistics data as it is produced by the integration node (broker).

Related information:



WebSphere MQ Version 7 product documentation

Resolving problems when using Data Analysis

Use the following advice to help you to resolve problems that can arise when you use Data Analysis.

About this task

- “'DOCID' and 'ID' columns are inserted into my database”
- “No DDL script when using a z/OS DB2 database”

'DOCID' and 'ID' columns are inserted into my database

Procedure

- **Scenario:** You map your database to Data Analysis and the columns 'DOCID' and 'ID' are inserted into your database.
- **Explanation:** The 'DOCID' and 'ID' columns are added to provide a unique key string for database operations. The 'DOCID' column is set to a unique string that you can use to identify the insertion of each entry into the database and remove if necessary. The 'ID' is the foreign key that links related tables together.
- **Solution:** Works as designed.

No DDL script when using a z/OS DB2 database:

Procedure

- **Scenario:** You complete the steps in the Generate DDL wizard, but the DDL script is empty.
- **Explanation:** When the **Check model** option in the Generate DDL wizard is selected, the DDL script is not generated.
- **Solution:** When you configure the DDL script in the Generate DDL wizard, do not select **Check model**.

Related concepts:

Data Analysis

Data Analysis analyzes and filters information in complex XML documents. You can use this analysis to create a library, that contains Data Analysis tools to quickly and easily transform your data in IBM Integration Bus.

Resolving problems when using databases

Use the advice given here to help you to resolve problems that can arise when using databases.

Before you begin

Before you start:

- Read “Is there a problem with a database?” on page 659

Procedure

- “DB2 error message SQL0443N is issued” on page 810
- “DB2 error message SQL0805N is issued” on page 810
- “DB2 error message SQL0998N is issued on Linux” on page 810
- “DB2 error message SQL0998N or SQL1248N is issued” on page 811
- “DB2 error message SQL1040N is issued” on page 811
- “DB2 error message SQL1224N is issued when you connect to DB2” on page 811
- “DB2 or ODBC error messages are issued on z/OS” on page 813
- “You do not know how many database connections a broker requires” on page 813
- “You want to use XA with DB2 databases” on page 813
- “XA coordination fails if the database restarts while the broker is running” on page 814
- “Error message BIP2322 is issued when you access DB2 on z/OS” on page 814
- “Error message BIP2322 IM004 is issued when you connect to an Informix database” on page 814
- “On Oracle, a database operation fails to return any rows, even though the rows exist” on page 815
- “Broker commands fail when the Oracle 10g Release 2 client runs on Linux on POWER with Red Hat Enterprise Linux Advanced Server V4.0” on page 815
- “Error message BIP2322 Driver not capable is issued when you use an Informix database” on page 816
- “Database updates are not committed as expected” on page 816
- “You want to list the database connections that the broker holds” on page 816
- “You want to know whether the password that is set for a database is as expected” on page 817
- “The queue manager finds the XA resource manager is unavailable when configured for XA with DB2 on Windows” on page 817
- “Error messages are received when you are trying to remove a DB2 database on Windows when you are using a sample” on page 817
- “DB2 error message SQL7008N is issued” on page 818
- “SQLCODE -981 is issued when you access DB2 on z/OS” on page 818

DB2 error message SQL0443N is issued

Procedure

- **Scenario:** After you upgrade your DB2 server to a new fix pack level, a DB2 error message SQL0443N is issued if you invoke a DB2 Call Level Interface (CLI) catalog function, such as `SQLTables()`, `SQLColumns()`, or `SQLStatistics()`. An example of the error message is:

```
SQL0443N Routine "SYSIBM.SQLTABLES" (specific name "TABLES") has returned an error SQLSTATE with diagnostic text SYSIBM:CLI:-805". SQLSTATE=38553.
```

- **Solution:** Bind the `db2schema.bnd` file against each database by entering the following commands at a command prompt:

```
db2 terminate
db2 connect to database-name
db2 bind path\db2schema.bnd blocking all grant public sqlerror continue
db2 terminate
```

where *database-name* is the name of the database to which the utilities must be bound, and *path* is the full path name of the directory where the bind files are located. For example, the default location on Windows is `C:\Program Files\IBM\SQLLIB\bnd\` (on Windows 32-bit editions) or `C:\Program Files(x86)\IBM\SQLLIB\bnd\` (on Windows 64-bit editions).

To list all the names of databases for a particular DB2 instance, run the DB2 CLI command **db2 list database directory**. For further information, see the DB2 documentation.

DB2 error message SQL0805N is issued

Procedure

- **Scenario:** When a message flow that includes a Database node runs, SQL error SQL0805N NULLID.SQLLF000 is issued.
- **Solution:** Open a DB2 Command Line Processor window and issue a bind command to the database.

Linux **UNIX** On Linux and UNIX systems, enter the commands:

```
connect to db
bind ~/sqllib/bnd/@db2cli.lst grant public CLIPKG 5
connect reset
```

where *db* is the database name.

Windows On Windows systems, enter the commands:

```
connect to db
bind x:\sqllib\bnd\@db2cli.lst blocking all grant public
connect reset
```

where *x*: identifies the drive onto which you installed DB2, and *db* is the database name.

DB2 error message SQL0998N is issued on Linux

Procedure

- **Scenario:** You are trying to use a globally coordinated message flow with DB2 on Linux and error message SQL0998N is issued with Reason Code 09 and Subcode " ".
- **Solution:** Check that the `LD_ASSUME_KERNEL` environment variable is not set. If it is set, use the **unset** command to remove it from your environment and ensure that you modify your profile scripts so that it remains unset.

DB2 error message SQL0998N or SQL1248N is issued

Procedure

- **Scenario:** When you try to use a globally coordinated message flow with a DB2 database, you get one of the following error messages:
 - SQL0998N with Reason Code 09 and Subcode ""
 - SQL1248N with a message indicating that the database is not defined with the transaction manager
- **Solution:** Use the instructions in Configuring global coordination with DB2 to configure the database and XAResourceManager stanza.

DB2 error message SQL1040N is issued

Procedure

- **Scenario:** You are using a DB2 database, and error message BIP2322 is issued with error SQL1040N.
- **Explanation:** The following DB2 message indicates that the value of the DB2 database configuration parameter **maxappls** has been reached:

```
"SQL1040N The maximum number of applications is already connected to the database.
SQLSTATE=57030"
```

DB2 has rejected the attempt to connect.

- **Solution:**
 1. Stop all brokers that connect to the affected database.
 2. Increase the value of the **maxappls** configuration parameter. Also, check the value of the associated parameter **maxagents**, and increase it in line with **maxappls**.
 3. Restart the DB2 database.
 4. Restart the brokers.

DB2 error message SQL1224N is issued when you connect to DB2

Procedure

- **Scenario:** DB2 error message SQL1224N is issued when you connect to a DB2 database. This error indicates that a database agent could not be started, or was ended because of a database shutdown or force command.
- **Solution:** On AIX, use TCP/IP to connect to DB2 databases, to avoid the shared memory limit of 10 connections. To set up AIX and DB2 loop-back to use a TCP/IP connection:

1. Configure DB2 to use TCP/IP, and to start the TCP/IP listener. On the database server machine, log in as the DB2 instance owner, typically `db2inst1`, and issue the following commands:

```
db2set DB2COMM=tcPIP
db2stop
db2start
```

2. If the DB2 connection port is not defined in `/etc/services`, edit the services file to add the DB2 connection and interrupt ports. You must use unique names, and port numbers that are not already defined in the services file; for example:

```
db2svc1    3700/tcp          # DB2 Connection Service
db2isvc1   3701/tcp          # DB2 Interrupt Service
```

3. Update the DB2 configuration; for example:
`db2 update dbm cfg using svcname db2svc1`

where *db2svc1* is the name of the DB2 Connection port service in */etc/services*.

Alternatively, you can specify a port number directly.

4. Stop and restart the database by using the following commands:

```
db2stop
db2start
```

5. Catalog a new TCP/IP connection node:

```
db2 catalog tcpip node NODENAME remote HOSTNAME server db2svc1
```

where:

NODENAME

is the name of the new TCP/IP connection node. You can use *local* as your node name, if it is a unique identifier.

HOSTNAME

is the name of your computer.

db2svc1

is the name of the DB2 connection port service in */etc/services*.

Message DB20000I is displayed when the command completes successfully.

6. Catalog the database with a new alias name; for example:

```
db2 catalog database DATABASE as DBALIAS at node NODENAME
```

where:

DATABASE

is the physical name of the database.

DBALIAS

is the database alias name that you want to use.

Specify the new alias name in all subsequent references to the local database.

7. Stop and start DB2:

```
db2 terminate
db2stop
db2start
```

8. Log on with the user ID under which the broker is running.

9. Update the ODBC configuration file for each broker to add definitions for the database:

- a. At the top of the file, add a definition for the database alias name:

```
DBALIAS=IBM DB2 ODBC Driver
```

- b. Add a new stanza for the database alias:

```
[DBALIAS]
Driver=INSTHOME/sql1lib/lib/libdb2.a
Description=Database Alias
Database=DBALIAS
```

where *INSTHOME* is the path to your DB2 Instance directory.

10. Update your message flows to specify the alias database name, redeploy the BAR file to the broker, and test the flows.



DB2 or ODBC error messages are issued on z/OS Procedure

- **Scenario:** DB2 or ODBC messages are issued on z/OS indicating one or more of the following errors:
 - An exception was caught while issuing the database SQL connect command.
 - A database error occurred with an ODBC return code of -1, an SQL state of 58004 and a native error code of -99999.
- **Solution:** If an ODBC message is displayed:
 1. Turn ODBC application tracing on to produce the traceodbc file.
 2. Locate the traceodbc file, which is written to the /output subdirectory. For example, the full path might be /u/argo/VCP0BRK/output/traceodbc.
 3. Go to the bottom of this file and search for previous instances of SQLerror.Common DB2 problems include:
 - ODBC return code -1, SQL state 58004, Native error code -99999
These codes might be returned for the following reasons:
 - No SQL code. The DB2 subsystem is not started
 - RRS is not started.
 - SQLCODE 922.
The user ID of the started task is not authorized to use plan DSNACLI.
 - ODBC return code -1, SLQ state 42503, Native error code -553
These codes might be returned if the user ID of the started task is not authorized to use the current SQL ID. Reconfigure the broker and specify DB2_TABLE_NAME as a valid name, or create a RACF group, and connect the started task user ID to this group.

You do not know how many database connections a broker requires Procedure

- **Scenario:** You do not know how many database connections to set up for your broker.
- **Solution:** Determine the number of database connections required by a broker for capacity and resource planning. On DB2, the default action taken is to limit the number of concurrent connections to a database to the value of the **maxapps** configuration parameter; the default value for **maxapps** is 40. The associated parameter **maxagents** also affects the current connections.
The connection requirements for a single broker are:
 - Five are required by internal broker threads.
 - One is required for each database access node to separate ODBC data source names for each message flow thread (that is, if the same DSN is used by a different node, the same connection is used).

You want to use XA with DB2 databases Procedure

- **Scenario:** You want to use XA with one or more DB2 databases.
- **Solution:** Ensure that your queue manager is configured to use **ThreadOfControl=THREAD**.
 -   On Linux and UNIX systems, configure this parameter in the XAResourceManager stanza in the file qm.ini for the broker queue manager.

- **Windows** On Windows systems, configure this parameter in WebSphere MQ Explorer.

XA coordination fails if the database restarts while the broker is running

Procedure

- **Scenario:** XA global coordination fails, and you get an error like the following example, which is from a DB2 database:

```
Database error: SQL State '40003'; Native Error Code '-900'; Error Text '[IBM]
[CLI Driver] SQL0900N The application state is in error. A database connection
does not exist.SQSTATE=08003'.
```

- **Explanation:** A globally coordinated message flow cannot automatically reconnect to a database if the database is restarted while the broker is still running.
- **Solution:** Stop and restart the broker if the database goes down, or is brought down for a scheduled maintenance.

Error message BIP2322 is issued when you access DB2 on z/OS

Procedure

- **Scenario:** You are running a message flow in which a node attempts to access a table on a DB2 data-sharing group. If ODBC tracing is turned on, an error message is written to the traceodbc file:

```
SQLError( hEnv=0, hDbc=0, hStmt=1, pszSqlState=&302f8ecc, pfNativeError=&302f8ec8,
pszErrorMsg=&28f6a6d0, cbErrorMsgMax=1024, pcbErrorMsg=&302f8eb4 )
SQLError( pszSqlState="51002", pfNativeError=-805, pszErrorMsg="{DB2 for OS/390}
{ODBC Driver}{DSN06011}
DSNT408I SQLCODE = -805, ERROR: DBRM OR PACKAGE NAME DSN610GH..DSNCLICS.16877-
BE5086005F4 NOT FOUND IN PLAN DSNACLI. REASON 02
DSNT418I SQLSTATE = 51002 SQLSTATE RETURN CODE
DSNT415I SQLERRP = DSNXEPM SQL PROCEDURE DETECTING ERROR
DSNT416I SQLERRD = -350 0 0 -1 0 0 SQL DIAGNOSTIC INFORMATION
DSNT416I SQLERRD = X'FFFFFFEA2' X'00000000' X'00000000' X'FFFFFFF'
X'00000000' X'00000000' SQL DIAGNOSTIC INFORMATION
```

This error is accompanied by the following BIP2322 error message in the syslog:
BIP2322E: DATABASE ERROR: SQL STATE '51002'; NATIVE ERROR CODE '-805'

- **Explanation:** This error occurs when the DSNACLI plan has not been bound in the correct way.
- **Solution:** Ensure that the DSNACLI plan is bound correctly. See Binding a DB2 plan to use data-sharing groups on z/OS for information about how to complete this task.

Error message BIP2322 IM004 is issued when you connect to an Informix database

Procedure

- **Scenario:** You are using the `mqsicvp` command to connect to an Informix® database, and you see the following error message.

```
BIP2322E: Database error: SQL State 'IM004'; Native Error Code '0'; Error Text '[DataDirect][ODBC lib]
Driver's SQLAllocHandle on SQL_HANDLE_ENV failed'. The error has the following diagnostic information:
SQL State 'IM004' SQL Native Error Code '0' SQL Error Text '[DataDirect][ODBC lib] Driver's SQLAllocHandle
on SQL_HANDLE_ENV failed'.
```

- **Explanation:** This error can be caused when the environment for the database has not been initialized.
- **Solution:** Check the documentation for the client on your broker system for details of the actions that you must take. For example, you might have to specify the following environment variables:

```

export INFORMIXDIR=installation_directory_of_informix_client_software
export PATH=${INFORMIXDIR}/bin:${PATH}
export INFORMIXSERVER=server_name
export INFORMIXSQLHOSTS=${INFORMIXDIR}/etc/sqlhosts
export TERMCAP=${INFORMIXDIR}/etc/termcap
export TERM=vt100
export LIBPATH=${INFORMIXDIR}/lib:${INFORMIXDIR}/lib/esql:
                ${INFORMIXDIR}/lib/cli:$LIBPATH

```

where *server_name* is defined in the file `sqlhosts` (the required value is typically the machine name), and the location of the file `sqlhosts` is set up as part of the installation process.

To configure your system to run this setup at the start of every session, add these statements to the login profile of the user that is going to run the broker.

For more information, see *Command environment: Linux and UNIX systems*.

On Oracle, a database operation fails to return any rows, even though the rows exist

Procedure

- **Scenario:** You are using Oracle databases in your message flows, and ESQL binds against columns that are declared as data type CHAR, and those parameter markers are referenced in a WHERE clause. The database operation fails to return any rows, even though the rows exist.
- **Explanation:** Fixed-length character strings must be padded with blank characters on Oracle for this type of comparison to succeed.
- **Solution:** Define the CHAR columns as VARCHAR2 columns, or pad the ESQL variable with blank characters to the required column length, so that the comparison locates the required rows from the table.

Broker commands fail when the Oracle 10g Release 2 client runs on Linux on POWER with Red Hat Enterprise Linux Advanced Server V4.0

Procedure

- **Scenario:** Broker commands fail in an environment where an Oracle 10g Release 2 client runs on Linux on POWER® with Red Hat Enterprise Linux Advanced Server V4.0. Abend files might be created, with contents like the following data:

```

/opt/mqsi/lib/libCommonServices.so(.ImbAbendSignalHandler+0x10) [0x800017d3c0]
[0x80022b33f0]
/lib64/tls/libpthread.so.0[0x80cc2339d8]
/lib64/tls/libpthread.so.0[0x80cc230e4c]
/lib64/tls/libpthread.so.0[0x80cc22af54]
bipservice[0x10005e38]
/lib64/tls/libpthread.so.0[0x80cc22a1d8]
/lib64/tls/libc.so.6[0x80cc0dd3e4]

```

or

```

/opt/mqsi/lib/libImbCmdLib.so(.ZN15ImbCreateTables15createAllTablesEv+0x38)
[0x800014c9bc]mqsicreatebroker[0x10023324]mqsicreatebroker[0x1002c0c4]
/opt/mqsi/lib/libImbCmdLib.so(.ZN10ImbCmdBase20processCommandStringEv+0x6c)
[0x80001dcad4]mqsicreatebroker[0x1000d728]/lib64/tls/libc.so.6[0x80cc02423c]
/lib64/tls/libc.so.6[0x80cc0243c4]

```

- **Explanation:** The Oracle installation library contains copies of `libgcc` library files. The Oracle user profile adds the directory containing these files to the environment variable `LD_LIBRARY_PATH`. This action causes the `libgcc` library files to be found before the system libraries, and leads to the failure of broker commands and the production of abend files.

- **Solution:** Ensure that you add /lib64 to the environment variable LD_LIBRARY_PATH before the Oracle library path. Before you run `mqsiprofile`, include a string like:

```
/lib64::/usr/lib:/oracle/app/oracle/product/10.2.0/db_1/lib
```

which shows /lib64 preceding the Oracle library directory.

Error message BIP2322 Driver not capable is issued when you use an Informix database

Procedure

- **Scenario:** When you try to access an Informix database from a node in a message flow, the following error message is issued:
BIP2322E: Database error: SQL State 'HYC00'; Native Error Code '-11092'; Error Text '[Informix][Informix ODBC Driver]Driver not capable.'.
- **Explanation:** The broker uses transaction statements, therefore the database must be created, and configured to enable logging.
- **Solution:** Consult your database administrator to ensure that transaction logging has been enabled on the Informix database that the broker is trying to access. For example, create the database with a buffered log:
create database with [buffered] log;

Database updates are not committed as expected

Procedure

- **Scenario:** You have included a Database node, Compute node, or Filter node in a message flow, and you have set the **Transaction** property to Commit. The message flow has raised an exception and has rolled back, but the database updates have not been committed.
- **Explanation:** When you set **Transaction** to Commit, the database updates performed by the node are committed when the node completes successfully. If an exception is raised before the node has completed, and causes the message flow to be rolled back, the commit is not issued and the database updates are also rolled back. The conditions under which this situation can occur are:
 - The node itself causes an exception to be raised. The commit is never performed.
 - The ESQL contains a PROPAGATE statement. This statement does not complete until all processing along the path taken by the propagated message has completed, and control returns to the node. Only then can the commit be performed. If an exception is raised along this path, control is not returned and the database updates are rolled back as part of the message flow.
- **Solution:** Review the operation of the node that performs the database updates. For example, you might be able to split the work between two nodes, with the first updating the database, and the second propagating the output message. Consider changing the ESQL code to process the message in a different way.

You want to list the database connections that the broker holds

Procedure

- **Scenario:** You want to list the database connections that the broker holds.
- **Solution:** The broker does not have any functionality to list the connections that it has to a database. Use the facilities that your database supplies to list connections. Refer to the documentation for your database to find out how to perform this task.

You want to know whether the password that is set for a database is as expected

Procedure

- **Scenario:** You want to check the password that is set for a database that is associated with a broker is the password that you expect.
- **Solution:** Use the `mqsi reportdbparms` command with the broker name, user ID, and password. The command reports whether the entered password was correct or not. For more information, see “Checking the security credentials that are used by an integration node” on page 392.

The queue manager finds the XA resource manager is unavailable when configured for XA with DB2 on Windows

Procedure

- **Scenario:** You have configured a queue manager for XA coordination with DB2 on your Windows computer. When you restart the queue manager, it reports error AMQ7604 in the queue manager error log. All subsequent attempts at XA coordination between WebSphere MQ and DB2 fail.

The error message has the following content, or similar:

```
23/09/2008 15:43:54 - Process(5508.1) User(MUSR_MQADMIN) Program(amqzma0.exe)
AMQ7604: The XA resource manager 'DB2 MQBankDB database' was not available
when called for xa_open.
The queue manager is continuing without this resource manager.
```

- **Explanation:** The user ID that runs the WebSphere MQ Services process `amqmsrvn.exe`, which has a default value of `MUSR_MQADMIN`, is running with an access token that does not contain group membership information for the group `DB2USERS`.
- **Solution:** Check that the WebSphere MQ Services user ID is a member of the group `DB2USERS`, stop the WebSphere MQ service (for example, by issuing the command `net stop "IBM MQSeries"`), and all other processes that are running under the same user ID, and then restart these processes. You can restart your computer to stop and restart these processes after you have checked the user ID status, but this action is typically not required.

Error messages are received when you are trying to remove a DB2 database on Windows when you are using a sample

Procedure

- **Scenario:** You are running a sample, and you are trying to remove a DB2 database on your Windows computer, and you receive a BIP9835E error message with the error code `SQLSTATE=57019`.

The error message has content like the following data:

```
BIP9830I: Deleting the DB2 Database <Your database name>.
BIP9835E: The DB2 batch command failed with the error code SQLSTATE=57019.
The database <Your database name> could not be created/deleted.
The error code SQLSTATE=57019 was returned from the DB2 batch command.
```

- **Explanation:** If you use the DB2 Control Center to perform a query, a connection is opened against the database. This connection stays open until the DB2 Control Center is closed, at which point the connection is ended.
- **Solution:** Close the DB2 Control Center application, and try to run the sample again.

DB2 error message SQL7008N is issued

Procedure

- **Scenario:** You are using DB2 and encounter error SQL7008N when updating or inserting into tables on iSeries®.
- **Explanation:** SQL7008N is a common error when the tables being accessed on an iSeries server are not being journaled.
- **Solution:** To correct the error, take one of the following steps:
 - Journal your tables.
 - Change the isolation level to No Commit. You can change the isolation level of the database alias referenced by your ODBC data source to No Commit by using the following DB2 command:

```
db2 update cli cfg for section <db_alias> using TxnIsolation 32
```

SQLCODE -981 is issued when you access DB2 on z/OS

Procedure

- **Scenario:** You are running a message flow that uses ODBC database interaction. When a commit or rollback is attempted, DB2 reports an error with SQLCODE=-981 and SQLSTATE=57015. An error message is seen similar to: {DB2 FOR OS/390}{ODBC DRIVER}{DSN09015} DSNT408I SQLCODE = -981, ERROR: THE SQL STATEMENT FAILED BECAUSE THE RRSAF CONNECTION IS NOT IN A STATE THAT ALLOWS SQL OPERATIONS, REASON 00C12219
- **Explanation:** You can choose for ODBC database operations to be committed or rolled back irrespective of the success or failure of the message flow transaction as a whole. This error can be seen if you attempt to use more than one uncoordinated ODBC database connection on a single message flow thread.
- **Solution:** Only one uncoordinated ODBC database connection per thread is supported. Update your message flow to perform ODBC database operations outside of the message flow transaction on only one database. Any number of different databases are supported as part of a coordinated message flow transaction.

Related concepts:

“Security identities for integration nodes connecting to external systems” on page 289

You can access external systems from the message flows that you deploy to your integration nodes, and you must therefore consider the steps that you might want to take to secure that access.

ESQL procedures

An ESQL procedure is a subroutine that has no return value. It can accept input parameters from, and return output parameters to, the caller.

Related tasks:

“Is there a problem with a database?” on page 659

If you have database problems, complete a set of initial checks to identify errors.

“ODBC trace” on page 864

You can use various methods to trace for ODBC activity, depending on the operating system that you are using.

Invoking stored procedures

To invoke a procedure that is stored in a database, use the ESQL CALL statement. The stored procedure must be defined by a CREATE PROCEDURE statement that has a Language clause of DATABASE and an EXTERNAL NAME clause that identifies the name of the procedure in the database and, optionally, the database schema to which it belongs.

Related reference:

`mqsicreatebroker` command

Use the `mqsicreatebroker` command to create a broker and its associated resources.

`mqsdeletebroker` command

Use the `mqsdeletebroker` command to delete a named broker. The command also deletes the queues on the associated queue manager (created when the broker was created). You can also specify that the queue manager is to be deleted.

`mqsichangetrace` command

Use the `mqsichangetrace` command to set the tracing characteristics for a broker.

Database facilities

The database products used by IBM Integration Bus also record information that might be useful if you have any problems with their access.

Resolving problems when using publish/subscribe

Use the advice given here to help you to resolve common problems that can occur when you run publish/subscribe applications.

About this task

Procedure

- “Application responses are not received”
- “Your application is not receiving publications” on page 820
- “Publishing a message causes a filter error” on page 820
- “Symbols in subscription filters cause problems” on page 821
- “There are performance problems on AIX when the JIT compiler is not loaded” on page 821
- “The Publication node fails with MQRC 2035” on page 821

Application responses are not received

Procedure

- **Scenario:** Application responses are not received.
- **Explanation:** Depending on the application code, a publisher or subscriber might request confirmation that its message was processed successfully. Responses can make debugging client problems much easier, because a response code is given if a problem occurs.

Typically, a response is always returned to a subscriber. However, for the publishing side, a message might be published without the knowledge of the originating application (for example, by using the default topic property on the input node, or by using a Compute node to modify this property in a message flow). The results of processing that message are still logged in user trace, which might give clues as to what is happening.

If a response is not received, it is typically due to one of the following causes:

- The system is busy. Messages might build up on the input queue, and the client might not be waiting long enough for its response.
- WebSphere MQ expiry is being used. There are cases where this option is what is required, but the expiry of the input message is copied to the response. As a result, the message might expire on the input queue, or it might expire on the way back to the client. This situation is not an error, even with a persistent message.
- The input message or response might have been put to the dead-letter queue, if one is configured. Look on this queue to see if any new messages are there. This situation is typically accompanied by error messages in the log written by the broker that describe the problem. For example, the reply-to queue might have been specified incorrectly in the input message, therefore the reply message has been put to the dead-letter queue.
- **Solution:** If your application is not asking for responses (that is, not using messages of type MQMT_REQUEST) consider doing so, particularly when developing applications.

Your application is not receiving publications

Procedure

- **Scenario:** Your application is not receiving publications.
- **Explanation:** If an application has subscribed successfully (that is, has received an OK response to a RegSub message), it receives publications that match its subscription.

Subscribers are sent messages only if they match the topic, the subscription point, *and* the filter. Because the subscription point is specified in the message flow, not in the publication message, an incorrect message flow setting can cause unexpected failures.

A user trace of the flow that contains the Publication node shows you whether publications are matching anything.

- **Solution:** If a filter is being used, a user trace shows you whether this message is being evaluated as expected.

The case with multiple integration servers, or multiple brokers, is more complex. A response is sent to a subscriber after the message has been processed by the target integration server. Other integration servers (and brokers) are updated asynchronously. As a result, there might be a delay before publications made elsewhere are received. If the broker is busy, there can be a delay before messages are processed fully. In a multi-broker setup, if communications have been suspended, subscription changes are propagated through the network of brokers. Check the channels.

With multiple integration servers or brokers, it might be possible to fill intermediate WebSphere MQ queues if the load is high. This situation might be reported in the syslog (if a broker cannot put to a queue because it is full) or in the WebSphere MQ log (if a message coming across a channel cannot be put to the target queue because it is full). If you see messages of this type, display the queue depths on all your queue managers to see if any are almost full.

Publishing a message causes a filter error

Procedure

- **Scenario:** When you publish a message, you receive an error response message with reason text MQRCCF_FILTER_ERROR.
- **Explanation:** A broker returns this message to a publication when subscriptions have been registered that specify filter expressions (for Content Based Routing)

and an error has been encountered when the broker attempts to filter the published message. This situation can occur, for example, if a message is published that includes unsupported data types, or if the message body is corrupted.

Symbols in subscription filters cause problems

Procedure

- **Scenario:** If you specify certain symbols when you use filters in a subscription, the filter does not work. Sometimes your subscription messages are put to the dead-letter queue, and a number of error messages are written to the local error log indicating MQRFH2 parsing errors.
- **Explanation:** The MQRFH2 header employs standard XML encoding, so that its parser interprets some symbols in a special way.
- **Solution:** If you want to include these symbols in your filters, use the appropriate escape character to ensure that they are parsed correctly:

Symbol	Escape character
<	<
>	>
"	"
'	'
&	&

For example, if you want to use:

```
<Filter>Body.e_ALERT_BODY.eqnum<6/</Filter>
```

specify:

```
<Filter>Body.e_ALERT_BODY.eqnum&lt;6/</Filter>
```

The Publication node fails with MQRC 2035

Procedure


- **Scenario:** The Publication node fails with MQRC 2035.
- **Explanation:** IBM Integration Bus publishes messages with the user ID in the original message, not the broker service ID.
- **Solution:** You can force IBM Integration Bus to use the broker service ID in all circumstances by setting the environment variable *MQSI_PUBSUB_USE_BROKER_USERID* to any value. If there is no MQMD header, or if there is an MQMD header but its *UserIdentifier* field is blank, IBM Integration Bus continues to use the broker's user ID.

There are performance problems on AIX when the JIT compiler is not loaded

Procedure

- **Scenario:** There are performance problems on AIX when the JIT compiler is not loaded.
- **Explanation:** The environment variable LIBPATH can affect the loading of the Java JIT (just-in-time) compiler on AIX. IBM Integration Bus for AIX runs correctly without the JIT compiler, but publish/subscribe performance is adversely affected.
- **Solution:** Ensure that the JIT compiler runs.

The JIT compiler loads and runs correctly if LIBPATH is not set, therefore do not set LIBPATH. You can make libraries available by linking them into

`/var/wmqi/lib` (for all IBM Integration Bus for AIX processes) or `/usr/lib` (for all processes on the system).  IBM Integration Bus for AIX configuration does this action for DB2 libraries.

If it is necessary to set `LIBPATH`, update it to include the directory `/usr/java130/bin`. For example, you can use the following command to start the broker:

```
LIBPATH=/usr/local/lib:/usr/java130/bin mqsistart mybroker
```

Related concepts:

Publish/Subscribe

Publish/subscribe is a style of messaging application in which the providers of information (publishers) are decoupled from the consumers of that information (subscribers).

Trace

If you cannot get enough information about a particular problem from the entries that are available in the various logs, the next troubleshooting method to consider is using trace. Trace provides more details about what is happening while code runs. The information produced from trace is sent to a specified trace record, so that you or IBM support personnel can analyze it to discover the cause of your problem.

Routing using publish/subscribe applications

You can route your messages to applications using the publish/subscribe method of messaging.

Related tasks:

“Viewing Administration log information” on page 190

You can view Administration log information by using either the IBM Integration Explorer, or the CMP.

“Interpreting trace” on page 859

Use the information in a formatted trace file to identify unexpected behavior.

Related reference:

Publish/subscribe

Use the reference information in this section to help you develop publish/subscribe applications.

User trace

User trace is one of two types of optional trace that are available in IBM Integration Bus and provides more information than that provided by the entries that are written to the Administration log. User trace is inactive by default; you must activate it explicitly by using a command, or by selecting options in the IBM Integration Toolkit.

WebSphere MQ logs

WebSphere MQ messages are written to the local error log in the same way as IBM Integration Bus messages.

`mqsicreatebroker` command

Use the `mqsicreatebroker` command to create a broker and its associated resources.

`mqsichangebroker` command

Use the `mqsichangebroker` command to change one or more of the configuration

parameters of the broker.

Resolving problems with performance

Use the advice given here to help you to resolve common problems with performance.

About this task

- **Scenario:** You are experiencing problems with performance, such as:
 - Poor response times in the IBM Integration Toolkit when developing message flows
 - Poor response time at deployment
 - Individual messages taking a long time to process
 - Poor overall performance, or performance that does not scale well
- **Solution:** Possible solutions are:
 - Tune the broker
 - Speed up WebSphere MQ persistent messaging by optimizing the I/O (input/output)
 - Speed up database access by optimizing I/O
 - Increase system memory
 - Use additional instances or multiple integration servers
 - Optimize ESQL statements for best performance

A good source of information about performance is the set of reports in WebSphere MQ Family Category 2 (freeware) SupportPacs, available for download from the WebSphere MQ SupportPacs web page.

For more information, read “Do you have a component that is running slowly?” on page 663.

This topic contains advice for dealing with some common performance problems that can arise:

- “A WHILE loop in a large XML array takes a long time to process”
- “Message flow performance is reduced when you access message trees with many repeating records” on page 824
- “You are experiencing poor performance in the IBM Integration Toolkit when working with large projects” on page 825
- “Performance is reduced when you run Web Services with small message sizes” on page 826

A WHILE loop in a large XML array takes a long time to process Procedure

- **Scenario:** A WHILE loop in a large XML array takes a long time to process.
- **Explanation:** This problem arises when you use the CARDINALITY() function to determine the size of the array in the WHILE statement. With large arrays, the cost of determining the number of elements is proportional to the size of the array. The CARDINALITY function is invoked each time the WHILE statement is executed. The message has many elements, therefore takes a long time to process when running the loop in this way.
- **Solution:** Unless you have a message in which the number of elements of the array grows dynamically, determine the size of the array before entering the WHILE loop. Use code like the following example:

```

DECLARE i,c INTEGER;
SET i = 1;
SET c=CARDINALITY(OutputRoot.XMLNS.MyMessage.Array[ ]);
WHILE i <= c DO
    . . .
    . . . loop processing
    . . .
END WHILE;

```

Message flow performance is reduced when you access message trees with many repeating records

Procedure

- **Scenario:** Message flow performance is reduced when the following conditions are true:
 - You are using ESQL processing to manipulate a large message tree.
 - The message tree consists of repeating records or many fields.
 - You have used explicit SET statements with field reference paths to access or create the fields.
 - You have observed a gradual slowing of message flow processing as the ESQL processes more fields or repetitions.
- **Explanation:** This problem occurs when you use field references, rather than reference variables, to access or create consecutive fields or records.

Consider the following example, in which independent SET statements use field reference paths to manipulate the message tree. The SET statement takes a source and target parameter, where either or both parameters are field references:

```
SET OutputRoot.XMLNS.TestCase.StructureA.ParentA.field = '1';
```

The problem arises when the SET statement is used to create many more fields, as shown in the following example:

```

SET OutputRoot.XMLNS.TestCase.StructureA.ParentA.field1 = '1';
SET OutputRoot.XMLNS.TestCase.StructureA.ParentA.field2 = '2';
SET OutputRoot.XMLNS.TestCase.StructureA.ParentA.field3 = '3';
SET OutputRoot.XMLNS.TestCase.StructureA.ParentA.field4 = '4';
SET OutputRoot.XMLNS.TestCase.StructureA.ParentA.field5 = '5';

```

In this example, the five fields that are created are all children of ParentA. Before the specified field can be created or modified, the broker must navigate the named message tree to locate the point in the message tree that is to be altered. For example:

- To access field 1, the SET statement navigates to ParentA, then to the first field, therefore involving two navigations.
- To access field 5, the SET statement navigates to ParentA, then traverses each of the previous fields until it reaches field 5, therefore involving six navigations.

Navigating over all the fields that precede the specified field causes the loss in performance.

Now consider a scenario that accesses repeating fields in an input message tree; for example:

```

DECLARE myChar CHAR;
DECLARE thisRecord INT 0;
WHILE thisRecord < 10000 DO
  SET thisRecord = thisRecord + 1;
  SET myChar = InputRoot.MRM.myParent.myRepeatingRecord[thisRecord];
END WHILE;

```

When index notation is used, as the count increases, the processing needs to navigate over all the preceding fields to get the one it wants; that is, it has to count over the previous records to get to the one that is represented by the current indexed reference.

- When accessing `InputRoot.MRM.myParent.myRepeatingRecord[1]`, one navigation takes place to get to the first record.
- When accessing `InputRoot.MRM.myParent.myRepeatingRecord[2]`, two navigations take place to get to the second record.
- When accessing `InputRoot.MRM.myParent.myRepeatingRecord[N]`, N navigations take place to get to the N -th record.

Therefore, the total number of navigations for this WHILE loop is: $1 + 2 + 3 + \dots + N$, which is not linear.

- **Solution:** If you are accessing or creating consecutive fields or records, use reference variables. When you use reference variables, the statement navigates to the main parent, which maintains a pointer to the field in the message tree. The following example shows the ESQL that can be used to reduce the number of navigations when creating new output message tree fields:

```

SET OutputRoot.XMLNS.TestCase.StructureA.ParentA.field1 = '1';
DECLARE outRef REFERENCE TO OutputRoot.XMLNS.TestCase.StructureA.ParentA;
SET outRef.field2 = '2';
SET outRef.field3 = '3';
SET outRef.field4 = '4';
SET outRef.field5 = '5';

```

When referencing repeating input message tree fields, you could use the following ESQL:

```

DECLARE myChar CHAR;
DECLARE inputRef REFERENCE TO InputRoot.MRM.myParent.myRepeatingRecord[1];
WHILE LASTMOVE(inputRef) DO
  SET myChar = inputRef;
  MOVE inputRef NEXTSIBLING NAME 'myRepeatingRecord';
END WHILE;

```

For further information, see [Creating dynamic field references](#).

You are experiencing poor performance in the IBM Integration Toolkit when working with large projects

Procedure

- **Scenario:** You are experiencing poor performance in the IBM Integration Toolkit when working with large or complex projects.
- **Explanation:** Performance is reduced because of frequent project changes, such as adding and removing projects, or using **Project > Clean**. Complete project updates use large amounts of memory due to the size, number, and connections between files.
- **Solution:** Increase your system memory.

Performance is reduced when you run Web Services with small message sizes

Procedure

- **Scenario:** You see poor response times and throughput rates when you run Web Services using HTTP, and send smaller messages sizes (typically less than 32 KB). Throughput rates can fluctuate with message size. IBM Integration Bus running on the AIX platform might be affected.
- **Explanation:** The default configuration of HTTP enables the Nagle algorithm, which seeks to improve the efficiency of Internet Protocol networks by reducing the number of packets sent. It works by buffering small packets together, creating a smaller number of large packets. The HTTPRequest node uses the platform default for the **tcpnodelay** setting of its sockets. You can disable the Nagle algorithm at either the operating system level (system wide) or through IBM Integration Bus (affecting only the IBM Integration Bus HTTP sockets).
- **Solution:** Use the following commands to disable the Nagle algorithm:

HTTP Request node

```
mqsichangeproperties <BrokerName> -e <IntegrationServerName>
-o ComIbmSocketConnectionManager -n tcpNoDelay -v true|false
mqsichangeproperties <BrokerName> -e <IntegrationServerName>
-o ComIbmSocketConnectionManager -n tcpNoDelaySSL -v true|false
```

HTTP Listener / Tomcat servlet engine

```
mqsichangeproperties <BrokerName> -b httplistener
-o HTTPConnector -n tcpNoDelay -v true|false
mqsichangeproperties <BrokerName> -b httplistener
-o HTTPSConnector -n tcpNoDelay -v true|false
```

To determine the value set, take the following steps:

HTTP Request node

Use the following command:

```
mqsireportproperties <BrokerName> -e <IntegrationServerName>
-o ComIbmSocketConnectionManager -r
```

HTTP Listener / Tomcat servlet engine

Check the IBM Integration Bus registry.

Related concepts:

ESQL procedures

An ESQL procedure is a subroutine that has no return value. It can accept input parameters from, and return output parameters to, the caller.

Related tasks:

“Using logs” on page 839

There are a variety of logs that you can use to help with problem determination and troubleshooting.

“Clearing Administration log information” on page 193

Clear Administration log information to reduce the size of the log by using either the IBM Integration Explorer or the CMP.

Invoking stored procedures

To invoke a procedure that is stored in a database, use the ESQL CALL statement. The stored procedure must be defined by a CREATE PROCEDURE statement that has a Language clause of DATABASE and an EXTERNAL NAME clause that identifies the name of the procedure in the database and, optionally, the database schema to which it belongs.

Creating dynamic field references

You can use a variable of type REFERENCE as a dynamic reference to navigate a message tree. This acts in a similar way to a message cursor or a variable pointer.

Related reference:

WHILE statement

The WHILE statement evaluates a condition expression, and if it is TRUE executes a sequence of statements.

CARDINALITY function

Message Sets: Performance considerations when using regular expressions

Take care when specifying regular expressions: some forms of regular expression can involve a large amount of work to find the best match, which might degrade performance.

Resolving problems when you monitor message flows

Follow the advice for dealing with common problems that can arise when you are monitoring IBM Integration Bus flows.

You are not receiving monitoring events from IBM Integration Bus

Procedure

- **Scenario:** You are not receiving monitoring events from IBM Integration Bus.
- **Explanation:** You can configure IBM Integration Bus flows to emit event messages. These messages can be used to support transaction monitoring and business monitoring. The events that are published by IBM Integration Bus can be written to a transaction repository, creating an audit trail of the transactions that are processed by a broker. You can also use WebSphere Business Monitor to monitor events. To receive monitoring events, you must configure and enable event sources, activate monitoring, and subscribe to the topic for the flow.
- **Solution:** If you are not receiving monitoring events, check that the following resources are configured correctly.
 1. Check that event sources are configured on the flow by using monitoring properties or a monitoring profile. For more information, see “Deciding how to configure monitoring events for message flows” on page 573.
 2. Check that event sources are enabled by setting the **-i** parameter on the **mqsichangeflowmonitoring** command. For more information, see “Enabling and disabling event sources” on page 585.
 3. Check that monitoring is activated for the flow by setting the **-c** parameter on the **mqsichangeflowmonitoring** command. For more information, see “Activating monitoring” on page 583.
 4. Check that you have subscribed to the topic for the flow. For example:
`$SYS/Broker/brokerName/Monitoring/integrationServerName/flowName`
 5. If you are writing events to a transaction repository, complete the following extra checks.
 - Check that events are configured for your transactions.
 - Check that you have subscribed to the event topic and written events to a repository.
For more information, see “Monitoring scenarios” on page 571.
 6. If you are using WebSphere Business Monitor to monitor events, complete the following extra checks.

- Check that WebSphere Business Monitor is installed, and that a monitor server is created and started. For more information, see IBM WebSphere Business Monitor Version 7.0 product documentation.
 - Check that the WebSphere Business Monitor sample has been imported and deployed. For more information, see WebSphere Business Monitor.
 - Check that the monitoring model application and message-driven bean (MDB) are installed and configured. For more information, see Running the WebSphere Business Monitor sample.
- To report monitoring settings, use the **mqsireportflowmonitoring** command. For more information, see **mqsireportflowmonitoring** command.

Related concepts:

“Monitoring basics” on page 568

Message flows can be configured to emit events. The events can be read and used by other applications for transaction monitoring, transaction auditing, and business process monitoring.

Related tasks:

“Business-level monitoring” on page 565

You can configure your message flow to emit event messages that can be used to support transaction monitoring and auditing, and business process monitoring.

Related reference:



Monitoring message flows

The following reference topics on monitoring message flows are available:



mqsichangeflowmonitoring command

Use the **mqsichangeflowmonitoring** command to enable monitoring of message flows.



mqsireportflowmonitoring command

Use the **mqsireportflowmonitoring** command to display the current options for monitoring that have been set using the **mqsichangeflowmonitoring** command.

Resolving problems when developing CMP applications

Use the advice given here to help you to resolve problems that can arise when developing IBM Integration API (also known as the CMP) applications.

About this task

- “Your CMP application hangs if the broker is not available”
- “You set a property of an object and query its value, but the value has not changed” on page 829
- “You cannot connect to a broker when using a .broker file” on page 829

Your CMP application hangs if the broker is not available Procedure

- **Scenario:** When the broker is unavailable, the CMP application hangs.
- **Explanation:** Communication between the CMP and the broker is asynchronous, therefore the CMP hangs because it is waiting for a message from the broker.
- **Solution:** Configure the maximum amount of time that the CMP waits by using the following method:

```
// Wait for a maximum of 10 seconds
BrokerProxy.setRetryCharacteristics(10000);
```

The value specified represents the time in milliseconds that the CMP will wait for information before throwing the `BrokerProxyPropertyNotInitializedException` exception.

If you set this timeout value too low, an exception is thrown, even if the broker is available.

You set a property of an object and query its value, but the value has not changed

Procedure

- **Scenario:** You have set a property of an object, then queried its value; the value has not changed.
- **Explanation:** Methods that change properties of broker objects are not processed immediately. If you call a property change method on a CMP object, the CMP sends a message that requests the specified change to the broker. The broker processes the request asynchronously, and notifies all `AdministeredObjectListeners` of the affected object when the change has been attempted.
- **Solution:** Methods that change state typically return to the calling program as soon as the request has been put to the queue manager of the broker, or, following a call to `BrokerProxy.beginUpdates()`, as soon as the request has been added to the current batch. If the property has still not been updated after the action's response to the request has been returned to the application, consult the response message for more details.

You cannot connect to a broker when using a .broker file

Procedure

- **Scenario:** You cannot connect to a broker when you use a `.broker` file.
- **Explanation:** If your CMP applications use the `MQPropertyFileBrokerConnectionParameters` class, they can connect to a broker by using a connection file that has a `.broker` extension. However, this file can be parsed only if an XML parser is available.
- **Solution:** Ensure that a supported parser is available on the CLASSPATH. A supported parser is shipped with IBM Integration Bus.
Alternatively, your application can use the `MQBrokerConnectionParameters` class instead of the `MQPropertyFileBrokerConnectionParameters` class. This class connects to a broker by specifying the host name, queue manager name, and queue manager port of the target broker directly. This method does not require an XML parser.

Resolving problems with user-defined extensions

Advice for dealing with some common problems that can arise when you work with user-defined extensions

About this task

Procedure

- “You cannot deploy one of your user-defined nodes, despite having a plug-in LIL in the correct directory.” on page 830
- “You cannot deploy a flow with one of your user-defined nodes in it.” on page 830
- “You get problems when nodes try to use the ESQL path interface in the plug-in API” on page 830
- “After migration your custom property editor does not work” on page 831

- “Interpreting problems in user-defined extensions” on page 831
- “You want to debug classloading” on page 833
- “An error is issued when you deploy a user-defined extension on z/OS” on page 833
- “You cannot determine which user-defined extensions have been loaded by the broker on startup” on page 834
- “You are migrating a C user-defined node and `cniDefineNodeClass` returns `CCI_INV_IMPL_FUNCTION`.” on page 834

You cannot deploy one of your user-defined nodes, despite having a plug-in LIL in the correct directory.

Procedure

- **Scenario:** You cannot deploy one of your user-defined nodes, despite having a plug-in LIL in the correct directory.
- **Explanation:** You have `memset()` the data area to zero and have not initialized the `CNI_VFT` structure with the initialization constant `{CNI_VFT_DEFAULT}`.
- **Solution:** Initialize by copying a predefined initialization structure over the function table area, as follows:

```
static CNI_VFT virtualFunctionTable = {CNI_VFT_DEFAULT};
```

In addition, implement logging from your user-defined node so that you can see if the plug-in API is producing error codes; the broker does not log these to its own log, unless you take a service trace.

You cannot deploy a flow with one of your user-defined nodes in it.

Procedure

- **Scenario:** You cannot deploy a flow with one of your user-defined nodes in it.
- **Explanation:** Your LIL file has failed to load.
- **Solution:** Check system log (syslog or Eventviewer) of broker startup; did you see a BIP2308 message saying a LIL file failed to load? If there are any problems loading a LIL file, a BIP2308 message appears in the system log.

You get problems when nodes try to use the ESQL path interface in the plug-in API

Procedure

- **Scenario:** When you attempt to use the ESQL path interface in the plug-in API, the return value is `CCI_PATH_NOT_NAVIGABLE`.
- **Explanation:** The plug-in API allows you to specify a path in the form of an ESQL path expression and navigate to that element, returning a handle to it if it exists. It also allows you to create elements along the path to the requested element.

The navigate path utility function (`cniSqlNavigatePath`) executes the `SQLPathExpression` created with the `cniSqlCreateReadOnlyPathExpression` or `cniSqlCreateModifiablePathExpression` utility functions, as defined by the `sqlPathExpression` argument.

If the path is not navigable, the return code is set to `CCI_PATH_NOT_NAVIGABLE`. This might be returned when embedding a path expression in another path expression. The input `cciMessage*` functions must not be NULL; however, any of the output `cciMessage*` functions can be NULL. If you embed a path expression that can be NULL inside a path expression that cannot be NULL, `CCI_PATH_NOT_NAVIGABLE` is returned.

- **Solution:** If the return code is set to CCI_PATH_NOT_NAVIGABLE, ensure that if a correlation name is specified in a path, the respective parameter is not NULL.

After migration your custom property editor does not work Procedure

- **Scenario:** You have migrated to a new version of IBM Integration Bus and your custom property editor no longer works.
- **Explanation:** Custom property editors can use Eclipse or RAD APIs. If any of those APIs are changed in a new version of IBM Integration Bus, your property editor might not work.
- **Solution:** Update your property editor code to comply with the changed API.

Interpreting problems in user-defined extensions Procedure

- **Scenario:** You want to debug problems in user-defined nodes and parsers.
- **Solution:** Start user trace at debug level. In order to see BIP4142, BIP4144, BIP4145, and BIP4146 messages, this must be done at the integration server level. For example, use the **mqsichangetrace** command without the **-f** parameter.

The following debug messages are available to help you to understand the execution of your user-defined nodes and parsers:

- BIP2233 and BIP2234: a pair of messages that are traced before and after a user-defined extension implementation function is invoked. These messages report the input parameters and returned value. For example:

```
BIP2233 Invoking user-defined extension function [function name]
([function call parameters])
```

```
BIP2234 Returned from user-defined extension function [function name]
with result: [result of call]
```

Note: In these messages, an *implementation function* can be interpreted as either a C implementation function or a Java implementation method.

- BIP2308: a message that is logged when the broker fails to load a LIL file.
BIP2308 File [name of LIL file] could not be loaded; operating system return code [error code return from operating system]
- BIP3904: a message that is traced before invoking the Java evaluate() method of a user-defined node. For example:
BIP3904 (for Java): Invoking the evaluate() method of node (class=[node class name], name=[label of node in flow]) where *node class name* is the name of the Java user-defined extension class.
- BIP3905: a message that is traced before invoking the C cniEvaluate implementation function (iFpEvaluate member of CNI_VFT) of a user-defined node. For example:
BIP3905 (for C): Invoking the cniEvaluate() implementation function of node (class=[node class name], name=[label of node in flow]) where *node class name* is the name of the user-defined extension class that is provided by the user-defined extension while calling C cniDefineNodeClass.
- BIP4142: a debug message that is traced when invoking a user-defined node utility function, where the utility function alters the state of a syntax element. This includes all utility functions that start with cniSetElement*, where * represents all nodes with that stem. For example:

BIP4142 Evaluating cniSetElement [element identifier type]. Changing value from [value before user's change] to [value after user's change]"

- BIP4144 and BIP4145: a pair of messages that are traced by certain implementation functions that, when invoked by a user-defined extension, can modify the internal state of a message broker's object. Possible message broker objects include syntax element, node, and parser. These messages report the input parameter provided to the invoked method and the returned value. For example:

BIP4144 Entered function [function name] ([function call parameters])

BIP4145 Exiting function [function name] with result: [result to be returned]

In these messages, an *implementation function* can be interpreted as either a C implementation function or a Java implementation method.

The C implementation functions that invoke messages BIP4144 and BIP4145 include:

For user-defined parsers	For user-defined nodes
cpiCreateParserFactory	cniCreateElement*
cpiDefineParserClass	cniDeleteMessage
cpiAppendToBuffer	cniAdd*
cpiCreateElement	cniDetach
cpiCreateAndInitializeElement	cniCopyElementTree
cpiAddBefore	cniFinalize
cpiAddAfter	cniWriteBuffer
cpiAddAsFirstChild	cniSql*
cpiAddAsLastChild	cniSetInputBuffer
cpiSetNameFromBuffer	cniDispatchThread

(* represents all nodes with that stem; for example, cniAdd* includes cniAddAfter, cniAddasFirstChild, cniAddasLastChild, and cniAddBefore.)

The Java methods that invoke messages BIP4144 and BIP4145 are:

For user-defined nodes
com.ibm.broker.plugin.MbElement.CreateElement*
com.ibm.broker.plugin.MbElement.add*
com.ibm.broker.plugin.MbElement.detach
com.ibm.broker.plugin.MbElement.copyElementTree

- BIP4146: a debug message that is traced when invoking a user-defined parser utility function, where the utility function alters the state of a syntax element. This includes all utility functions that start with cpiSetElement*, where * represents all nodes with that stem. For example:

BIP4146 Evaluating cpiSetElement [element identifier type]. Changing value from [value before user's change] to [value after user's change]

For information on the C user-defined API, see the C language user-defined parser API and the C language user-defined node API.

- BIP4147: an error message that is traced when a user-defined extension passes an invalid input object to a user-defined extension utility API function. For Example:

- BIP4147 User-defined extension input parameter failed debug validation check. Input parameter [parameter name] passed into function [function name] is not a valid object.
- BIP4148: an error message that is traced when a user-defined extension damages a broker's object. For Example:
BIP4148 User-defined extension damaged broker's object. Function [function name] has damaged broker's object passed as parameter [parameter name].
 - BIP4149: an error message that is traced when a user-defined extension passes an invalid input data pointer to a user-defined extension utility API function. For Example:
BIP4149 User-defined extension input parameter failed debug validation check. Input parameter [parameter name] passed into function [function name] is a NULL pointer.
 - BIP4150: an error message that is traced when a user-defined extension passes invalid input data to a user-defined extension utility API function. For example:
BIP4150 User-defined extension input parameter failed debug validation check. Input parameter [parameter name] passed into function [function name] does not have a valid value.
 - BIP4151: a debug message that is traced when `cnigetAttribute2` or `cnigetAttributeName2` sets the return code to an unexpected value. Expected values are `CCI_SUCCESS`, `CCI_ATTRIBUTE_UNKNOWN`, and `CCI_BUFFER_TOO_SMALL`. Any other value is an unexpected value. For example:
BIP4151 An unexpected value was returned from User-defined extension implementation function [function name].
 - BIP4152: a debug message that is traced when `cnigetAttribute2` or `cnigetAttributeName2` sets the return code to `CCI_BUFFER_TOO_SMALL`, and then `cnigetAttribute2` or `cnigetAttributeName2` is called again, this time with the correct size buffer, however the return code is still set to `CCI_BUFFER_TOO_SMALL`. For example:
BIP4152 User-defined extension Implementation function [function name] returned `CCI_BUFFER_TOO_SMALL` on 2nd attempt.

You want to debug classloading Procedure

- **Scenario:** You want to debug classloading.
- **Solution:** Classes and the location from which they are loaded are written to user trace. Use this information to check that the correct classes are being loaded.

An error is issued when you deploy a user-defined extension on z/OS Procedure

- **Scenario:** When you deploy a user-defined extension on z/OS, Linux, or UNIX, an error is displayed in the log of each integration server, stating that there is insufficient authority to open the LIL file.
- **Explanation:** On Linux and UNIX, the user-defined extension must have group read permission. On z/OS, the user-defined extension must have group execute permission.
- **Solution:**

- On Linux and UNIX, set the file permissions of the user-defined extension to group read by issuing the command **chmod a+r**.
- On z/OS, set the file permissions of the user-defined extension to group read and execute by issuing the command **chmod a+rx**.

You cannot determine which user-defined extensions have been loaded by the broker on startup

Procedure

- **Scenario:** You cannot determine which user-defined extensions have been loaded by the broker on startup.
- **Solution:** Use the **mqsireportproperties** command for each type of user-defined extension.
 - For a Java user-defined extension, issue the command:

```
mqsireportproperties WBRK_BROKER -e default -o ComIbmJavaPluginNodeFactory -r
```

You see a report similar to this example:

```
ComIbmJavaPluginNodeFactory
  uuid='ComIbmJavaPluginNodeFactory'
  userTraceLevel='none'
  traceLevel='none'
  userTraceFilter='none'
  traceFilter='none'
  NodeClassName='ComIbmJMSSClientInputNode'
  NodeClassName='ComIbmJMSSClientOutputNode'
  NodeClassName='ComIbmJavaComputeNode'
  NodeClassName='ComIbmXslMqsiNode'
  NodeClassName='SearchFilterNode'
```

BIP8071I: Successful command completion.

The user-defined extension called SearchFilter has a NodeClassName of SearchFilterNode.

- For a C user-defined extension (assuming that **CONST_PLUGIN_NODE_FACTORY** was set to **ComIbmSamplePluginNodeFactory** in the **NodeFactory.h** file, as in the sample **NumComputeNode**), issue the command:

```
mqsireportproperties WBRK_BROKER -e default -o ComIbmSamplePluginNodeFactory -r
```

You see a report similar to this example:

```
ComIbmSamplePluginNodeFactory
  uuid='ComIbmSamplePluginNodeFactory'
  userTraceLevel='none'
  traceLevel='none'
  userTraceFilter='none'
  traceFilter='none'
  NodeClassName='NumComputeNode'
```

BIP8071I: Successful command completion.

The user-defined extension called NumCompute has a NodeClassName of NumComputeNode.

You are migrating a C user-defined node and **cniDefineNodeClass** returns **CCI_INV_IMPL_FUNCTION**.

Procedure

- **Scenario:** When you attempt to migrate a C user-defined node, **cniDefineNodeClass** returns **CCL_INV_IMPL_FUNCTION**.

- **Explanation:** New fields have been added to the CNI_VFT struct. CNI_VFT_DEFAULT has been updated to initialize these new fields in the header file BipCci.h. If you initialize your CNI_VFT with CNI_VFT_DEFAULT, you should not need to make any code changes. However, if you do not initialize CNI_VFT with CNI_VFT_DEFAULT, these new fields are initialized with random values.
- **Solution:** Initialize your CNI_VFT with CNI_VFT_DEFAULT.

Related concepts:

 User-defined extensions overview

A user-defined extension is an optional component that you design and create to extend the functionality of IBM Integration Bus.


 User-defined message processing nodes

A user-defined message processing node is a node that you can create to complement the supplied built-in node types.

Related tasks:

 Installing user-defined extension runtime files on a broker

Install the compiled runtime files for your user-defined extension on the broker on which you want to test its function. If your user-defined node uses a custom compiler, install the user-defined node plug-in to the broker on which you want to deploy the node.

 Starting user trace

Use user trace for debugging your applications; you can trace brokers, integration servers, and deployed message flows. Start user trace facilities using the **mqsi changetrace** command or the IBM Integration Explorer.

“Is there a problem with a database?” on page 659

If you have database problems, complete a set of initial checks to identify errors.

“Has the message flow run successfully before?” on page 656

Sometimes a problem appears in a message flow that has previously run successfully.

Related reference:

 cciGetLastExceptionData

Gets diagnostic information about the last exception generated. Information about the last exception generated on the current thread is returned in a CCI_EXCEPTION_ST output structure. The user-defined extension can use this function to determine whether any recovery is required when a utility function returns an error code.

 Utility function return codes and values

By convention, the return code output parameter of all utility functions is set to indicate successful completion, or an error. The table lists all return codes with their meanings.

Resolving problems when you convert WebSphere Enterprise Service Bus resources

Use the advice that is given here to help you resolve common problems that can arise when you convert WebSphere Enterprise Service Bus resources by using the WebSphere ESB conversion tool.

About this task

The following list describes known problems with a corresponding solution or workaround:

- “Avoiding the launch of the Workspace Migration wizard”
- “Correctly displaying the project icon types in the IBM Integration Bus workspace” on page 837
- “Resolving unbound classpath container errors reported in the IBM Integration Bus Problems view” on page 838
- “Correctly displaying WebSphere Enterprise Service Bus resources in your WebSphere ESB conversion session” on page 838

Related concepts:



WebSphere ESB conversion tool limitations

A number of constraints apply when you convert WebSphere Enterprise Service Bus resources into IBM Integration Bus resources by using the WebSphere ESB conversion tool.

Related tasks:



Importing WebSphere Enterprise Service Bus resources

You must import your WebSphere Enterprise Service Bus resources before you can convert them to IBM Integration Bus resources by using the WebSphere ESB conversion tool.



Running the WebSphere ESB conversion tool

Start the WebSphere ESB conversion tool, then create and configure a WebSphere ESB conversion session to convert WebSphere Enterprise Service Bus resources into IBM Integration Bus resources.

Related information:



IBM Business Process Manager Version 8 product documentation

Avoiding the launch of the Workspace Migration wizard

About this task

Scenario 1:

You import a WebSphere Enterprise Service Bus PI file into the IBM Integration Bus workspace. A Workspace Migration wizard opens when you:

- Import a WebSphere Enterprise Service Bus PI file directly from the file system.
- Import a WebSphere Enterprise Service Bus PI file by using the IBM Integration Bus **Project Interchange** import option.
- Import a WebSphere Enterprise Service Bus PI file:
 - By using the WebSphere ESB Project Interchange import option, and
 - The PI file includes project types for which IBM Integration Bus does not have server runtime support, for example, Web and App projects.
- Import a WebSphere Enterprise Service Bus PI file that contains a Java project.

Scenario 2:

You open an IBM Integration Bus workspace that includes WebSphere Enterprise Service Bus projects. You imported these projects by using the **Project Interchange** import option. A Workspace Migration wizard opens.

Explanation on the Workspace Migration wizard options:

When you get the first Workspace Migration wizard page, you have two options:

- Click **Next** to continue.
- Click **Cancel** to exit.

If you click **Next**:

1. Continue to click **Next** until the option **Finish** is enabled.
2. If you have project types that are not a WebSphere Enterprise Service Bus library project or a WebSphere Enterprise Service Bus mediation module project, the Undefined Server Runtime page opens. The window displays the following message:

No existing server run time supports the selected list of projects. The migration wizard will not change the target runtime for these projects.

Click **Next** to continue. The WebSphere ESB conversion tool does not convert all WebSphere Enterprise Service Bus project types. For more information, see WebSphere ESB conversion tool limitations.

3. Click **Finish**. The Migration Validation window opens. The window displays the following message: Migration validation completed successfully. See the Migration Results view for additional information.
4. Click **Ok** to finish.

Procedure

Solution:

Complete the following tasks in order:

1. In IBM Integration Designer, export the WebSphere Enterprise Service Bus projects that you want to convert to IBM Integration Bus projects by selecting the **Business Integration** export option **Integration Modules and Libraries**. For more information, see Exporting WebSphere Enterprise Service Bus projects for conversion.
2. Import WebSphere Enterprise Service Bus resources by using the **WebSphere ESB Project Interchange** import option. For more information, see Importing WebSphere Enterprise Service Bus resources.
3. Optional: If your WebSphere Enterprise Service Bus PI file includes a Java project, complete the following steps in order:
 - In the Application Development view, complete the following steps:
 - a. Expand **Independent resources**. Select *YourJavaProject*, where *YourJavaProject* is the name of your Java project.
 - b. Right-click the *YourJavaProject* and select **Properties**.
 - Click **Targeted Runtimes**. In the list of runtimes, clear any selected entries, for example `wps.esb.v75`. Click **Apply**.
 - Click **Java Build Path**, then click JRE System Library. Click **Edit**, and choose the option **Workspace default JRE**. Click **Finish**.
 - Click **Ok**.

Correctly displaying the project icon types in the IBM Integration Bus workspace

About this task

Scenario 1:

WebSphere Enterprise Service Bus resources that are imported into IBM Integration Bus workspace display incorrect project icon types and are marked in error.

Projects are represented in IBM Integration Bus with the wrong icon type when you import WebSphere Enterprise Service Bus files directly from the file system or by using the standard **Project Interchange** import option. Icons display in error.

Procedure

Solution:

Complete the following tasks in order:

1. Import WebSphere Enterprise Service Bus resources by using the **WebSphere ESB Project Interchange** import option.
2. Resolve all unbound classpath container errors that result from importing WebSphere Enterprise Service Bus resources into the IBM Integration Bus workspace. For more information, see “Resolving unbound classpath container errors reported in the IBM Integration Bus Problems view.”

Resolving unbound classpath container errors reported in the IBM Integration Bus Problems view

About this task

Scenario 1:

When you import WebSphere Enterprise Service Bus resources by using the option **Project Interchange**, the following errors are reported in the Problems view:

- Error 1: The project cannot be built until build path errors are resolved.
- Error 2: Unbound classpath container: JRE System Library [IBM Process Server v8.0 JRE] in project *project name*. *Project name* corresponds to the name of the WebSphere Enterprise Service Bus imported project in IBM Integration Bus workspace.

Procedure

Solution:

Complete the following tasks in order:

1. Right-click the message that describes error 2 in the Problems tab. Select **Quick Fix**. The window Quick Fix opens.
2. Select the option **Select a system library to use when building *project name***, where *project name* is the name of the WebSphere Enterprise Service Bus imported project in IBM Integration Bus workspace.
3. Click **Finish**. The window Edit Library opens.
4. Select the option **Workspace default JRE (jdk)**.
5. Click **Finish**.

Correctly displaying WebSphere Enterprise Service Bus resources in your WebSphere ESB conversion session

About this task

Scenario 1:

When you reimport a WebSphere Enterprise Service Bus project into your WebSphere ESB conversion session, the WebSphere Enterprise Service Bus

resources included in the WebSphere Enterprise Service Bus project are not refreshed automatically.

Procedure

Solution:

To view your WebSphere Enterprise Service Bus project resources in your WebSphere ESB conversion session, complete the following tasks in order:

1. Close your WebSphere ESB conversion session.
2. Open your WebSphere ESB conversion session.
 - a. In the Application Development view, select **Independent resources > WESB_conversions**.
 - b. Double-click **YourConversionSessionName**.

Results

Your WebSphere Enterprise Service Bus resources display correctly.

Using logs

There are a variety of logs that you can use to help with problem determination and troubleshooting.

About this task

This section describes how to view the various logs available to you with IBM Integration Bus, and how to interpret the information in those logs. It contains the following topic areas:

Local error log

- “Windows: Viewing the local error log” on page 840
- “Linux and UNIX systems: Configuring the syslog daemon” on page 841
- “z/OS: Viewing broker job logs” on page 843

Eclipse log

- “Viewing the Eclipse error log” on page 844
- “Viewing Activity logs for message flows and resource types” on page 626

There is also a section of reference topics about the various types of log.

Related concepts:



Logs

If an error is reported by an IBM Integration Bus component, start your investigations into its causes by looking at the product and systems logs to which information is written during component operation.

Related tasks:

Chapter 4, “Troubleshooting and support,” on page 647

If you are having problems with your message flow applications, use the techniques described in this section to help you to diagnose and solve the problems.

Related reference:



Local error logs

IBM Integration Bus components use the local error log (also known as the system log) to record information about major activities within the system. When an error occurs, check the local error log first.



IBM Integration Bus logs

IBM Integration Bus writes information to a number of product-specific logs to report the results of actions that you take.

Windows: Viewing the local error log

The Windows Event Viewer is where IBM Integration Bus writes records to the local system. Use Windows system facilities to view this log.

Viewing the system log

About this task

The system log contains events logged by the Windows system components. For example, the failure of a driver or other system component to load during startup is recorded in the system log. To view the system log:

Procedure

1. Open a command prompt.
2. At the prompt, type **eventvwr**. This opens the Windows Event Viewer.

Viewing the application log

About this task

The application log contains events that are logged by applications or programs. For example, a database program might record a file error in the application log. To view the application log:

Procedure

1. Open a command prompt.
2. At the prompt, type **eventvwr**. This opens the Windows Event Viewer.

Interpreting log information

About this task

In both logs, each event is displayed on a separate row, in date and time order (most recent first), with the following information:

- **Type:** The event type, which can be information, a warning, or an error.
- **Date and time:** The date and time when the event was written to the log.
- **Source:** What action has caused the event.
- **Category:** The category of the event. The default category is none.
- **Event:** The event number.
- **User:** The name of the user at the time of the event.
- **Computer:** The name of the local machine.

To view an individual log entry:

Procedure

1. Within the system or application log, find the log entry.
2. Right-click the entry.
 - On Windows, click **Events** to open the Event Properties window.
The window shows a description of the event. Select the Details tab to view bytes or words that were parsed when the record was written to the log.
3. Use the up and down arrows to move through the events of the log.
4. To close the window, click **OK** to return to the system or application log.

Related concepts:



Logs

If an error is reported by an IBM Integration Bus component, start your investigations into its causes by looking at the product and systems logs to which information is written during component operation.

Related tasks:

Chapter 4, “Troubleshooting and support,” on page 647

If you are having problems with your message flow applications, use the techniques described in this section to help you to diagnose and solve the problems.

“Using logs” on page 839

There are a variety of logs that you can use to help with problem determination and troubleshooting.

“Linux and UNIX systems: Configuring the syslog daemon”

On Linux and UNIX systems, all IBM Integration Bus messages (other than messages that are generated by the command-line utilities) are sent to the syslog subsystem.

“z/OS: Viewing broker job logs” on page 843

On z/OS, the broker writes messages to the appropriate z/OS system log and job logs. These messages might include information, warning, error, and severe messages to indicate various situations and events.

Related reference:



Local error logs

IBM Integration Bus components use the local error log (also known as the system log) to record information about major activities within the system. When an error occurs, check the local error log first.

Linux and UNIX systems: Configuring the syslog daemon

On Linux and UNIX systems, all IBM Integration Bus messages (other than messages that are generated by the command-line utilities) are sent to the syslog subsystem.

About this task

UNIX You must configure this subsystem so that all diagnostic messages that enable you to monitor the performance and behavior of your broker environment are displayed.

The configuration steps you make to ensure that all relevant messages are displayed depend on the version of Linux and UNIX that you are using. Refer to your operating system documentation relating to syslog (or syslog-ng for some versions of Linux) for information about how to configure the syslog subsystem.

IBM Integration Bus processes call the syslog commands on the operating system but only those messages that correspond to the filter defined for the output destination are displayed. IBM Integration Bus messages have:

- A facility of user.
- A level of `err`, `warn`, or `info`, depending on the severity of the situation causing the message to be issued.

To record all IBM Integration Bus messages, create a filter on the user facility for messages of level `info` or greater. It is good practice to write these messages to a separate file; there might be a high number of them and they are more likely to be of interest to broker administrators rather than to system administrators.

The following line in a `syslog.conf` file causes all IBM Integration Bus events to be written to a file `/var/log/user.log`

```
user.info /var/log/user.log
```

Many UNIX systems provide a command-line utility, known as `logger`, to help you test and refine your configuration of the syslog subsystem.

On UNIX, syslog entries are restricted in length and messages that are sent to the syslog are truncated by the new line character. To record a large amount of data in a log on UNIX, set the `Destination` property on the `Trace` node to `File` or `User Trace` instead of `Local Error Log`.

What to do next

See the documentation for your operating system.

Related concepts:



Logs

If an error is reported by an IBM Integration Bus component, start your investigations into its causes by looking at the product and systems logs to which information is written during component operation.

Related tasks:

Chapter 4, “Troubleshooting and support,” on page 647

If you are having problems with your message flow applications, use the techniques described in this section to help you to diagnose and solve the problems.

“Using logs” on page 839

There are a variety of logs that you can use to help with problem determination and troubleshooting.

“Windows: Viewing the local error log” on page 840

The Windows Event Viewer is where IBM Integration Bus writes records to the local system. Use Windows system facilities to view this log.

“z/OS: Viewing broker job logs” on page 843

On z/OS, the broker writes messages to the appropriate z/OS system log and job logs. These messages might include information, warning, error, and severe messages to indicate various situations and events.

Related reference:



Local error logs

IBM Integration Bus components use the local error log (also known as the system log) to record information about major activities within the system. When an error occurs, check the local error log first.

z/OS: Viewing broker job logs

On z/OS, the broker writes messages to the appropriate z/OS system log and job logs. These messages might include information, warning, error, and severe messages to indicate various situations and events.

Understanding the broker address spaces

About this task

The broker runs in multiple address spaces. A single control address space is always running when the broker is active, and is responsible for communicating with the IBM Integration Toolkit.

Each integration server within the broker is mapped to its own address space, and as these integration servers start and stop, the corresponding address spaces are started and stopped. The control address space is assigned a JOBNAME and STEPNAME, which is the same as the broker name. The integration servers have a JOBNAME that is also the same as the broker name, and a STEPNAME that matches the last seven characters of the integration server name.

Viewing the z/OS system console log

About this task

The broker writes all its messages to the z/OS system console log. You can see messages from all address spaces running on the z/OS system in this log. It is easy to identify jobs that are associated with the broker in the console log, because of the naming of broker address spaces. By using the console log, you can see the order of event reporting for different products. This information can be helpful for cross-product problem determination.

Viewing the broker job logs

About this task

The broker control address space, and each of the integration server address spaces, has its own job log. When you select the job log for the appropriate address space, you can see all messages relating to that address space. Use this option in a busy system where the system console log might have many messages from different products obscuring the information in which you are interested.

Interpreting log information

About this task

In both logs, each message is displayed on its own, in date and time order; it might span multiple lines, if necessary. For each message, the following information is available:

- **Date and time:** The exact date and time when the event was written to the log.
- **JOBID:** The started task job identifier of the address space.
- **Message Number:** The message number that identifies whether the event is information, a warning, or an error, with diagnostic text.
- **JOBNAME:** The JOBNAME of the address space issuing the message. This name is always the same as the broker name.
- **STEPNAME:** The STEPNAME of the address space that is issuing the message. For the control address space, this name is the same as the broker name; for integration servers, it is the same as the last eight characters of the integration server that is issuing the message.

- **PROCSTEP:** The procedure STEPNAME, which equals BROKER for the main broker control address space. For integration server address spaces, this is either EGNOENV, if there is not an integration server specific profile, or EGENV if there is an integration server specific profile.

Related concepts:



Logs

If an error is reported by an IBM Integration Bus component, start your investigations into its causes by looking at the product and systems logs to which information is written during component operation.

Related tasks:

“Using logs” on page 839

There are a variety of logs that you can use to help with problem determination and troubleshooting.

“Linux and UNIX systems: Configuring the syslog daemon” on page 841

On Linux and UNIX systems, all IBM Integration Bus messages (other than messages that are generated by the command-line utilities) are sent to the syslog subsystem.

“Windows: Viewing the local error log” on page 840

The Windows Event Viewer is where IBM Integration Bus writes records to the local system. Use Windows system facilities to view this log.

Related reference:



Local error logs

IBM Integration Bus components use the local error log (also known as the system log) to record information about major activities within the system. When an error occurs, check the local error log first.



IBM Integration Bus logs

IBM Integration Bus writes information to a number of product-specific logs to report the results of actions that you take.

Viewing the Eclipse error log

The Eclipse error log captures internal errors that are caused by the operating system or your code.

About this task

To view the Eclipse error log:

Procedure

1. Switch to the Plug-in Development perspective.
2. From the main menu, select **Window > Show view > Other**. Then select **General > Error Log**.

The error log is displayed, showing the following information for each error:

- The status of the error (for example, error or warning)
 - A brief description of the error
 - From which plug-in the error derived
 - The date and time that the error was produced
3. If an error has a plus sign (+) at the start of it, it is a complex problem, and there are a number of errors contributing to it. Click the plus sign to view the individual errors.

4. To see the details of a particular problem, double-click the entry in the Problems view. A separate window is displayed, showing more details of the error.

Related concepts:



Logs

If an error is reported by an IBM Integration Bus component, start your investigations into its causes by looking at the product and systems logs to which information is written during component operation.

Related tasks:

“Using logs” on page 839

There are a variety of logs that you can use to help with problem determination and troubleshooting.

Related reference:



IBM Integration Bus logs

IBM Integration Bus writes information to a number of product-specific logs to report the results of actions that you take.

Using trace

You can use different types of trace to help you with problem determination and troubleshooting.

About this task

How to use the optional trace.

For user trace:

- Starting user trace
- Checking user trace options
- Changing user trace options
- Retrieving user trace
- Stopping user trace

For service trace:

- “Starting service trace” on page 847
- “Checking service trace options” on page 849
- “Changing service trace options” on page 851
- “Retrieving service trace” on page 854
- “Stopping service trace” on page 852

For both types of trace:

- “Formatting trace” on page 856
- “Interpreting trace” on page 859
- “Clearing old information from trace files” on page 861
- “Changing trace settings from the IBM Integration Explorer” on page 862

Other types of trace:

- “ODBC trace” on page 864
- “IBM Integration API (CMP) trace” on page 867
- “Switching Trace nodes on and off” on page 868

You can also use the “IBM Support Assistant Data Collector” on page 879 to help with data collection.

Related concepts:



Trace

If you cannot get enough information about a particular problem from the entries that are available in the various logs, the next troubleshooting method to consider is using trace. Trace provides more details about what is happening while code runs. The information produced from trace is sent to a specified trace record, so that you or IBM support personnel can analyze it to discover the cause of your problem.

Related tasks:

Chapter 4, “Troubleshooting and support,” on page 647

If you are having problems with your message flow applications, use the techniques described in this section to help you to diagnose and solve the problems.

“Making initial checks” on page 650

Before you start problem determination in detail, consider whether there is an obvious cause of the problem, or an area of investigation that is likely to give useful results. This approach to diagnosis can often save a lot of work by highlighting a simple error, or by narrowing down the range of possibilities.

“Contacting your IBM Support Center” on page 876

If you cannot resolve problems that you find when you use IBM Integration Bus, or if you are directed to do so by an error message generated by IBM Integration Bus, you can request assistance from your IBM Support Center.

“IBM Support Assistant Data Collector” on page 879

You can collect diagnostic documents by using IBM Support Assistant Data Collector, and submit a problem report to IBM. IBM Support Assistant Data Collector is included with your IBM Integration Bus installation.

Related reference:



User trace

User trace is one of two types of optional trace that are available in IBM Integration Bus and provides more information than that provided by the entries that are written to the Administration log. User trace is inactive by default; you must activate it explicitly by using a command, or by selecting options in the IBM Integration Toolkit.



Service trace

Service trace is one of two types of optional trace that are available in IBM Integration Bus and provides more information than that provided by the entries that are written to the Event Log or to user trace. Service trace is inactive by default; you must activate it explicitly by using a command.



mqsichangetrace command

Use the **mqsichangetrace** command to set the tracing characteristics for a broker.



mqsiformatlog command

Use the **mqsiformatlog** command to process the XML log created by **mqsireadlog**. The command retrieves and formats any messages that the XML log contains into a form suitable for the locale of the user who runs the command.



mqsireadlog command

Use the **mqsireadlog** command to retrieve trace records for the specified component.

mqsireporttrace command

Use the **mqsireporttrace** command to display the trace options currently in effect. Trace can be run only against a broker, or any of its resources.

Database facilities

The database products used by IBM Integration Bus also record information that might be useful if you have any problems with their access.

Starting service trace

Service trace is used to get detailed information about your environment for use by your IBM Support Center.

About this task

Activate service traces only when you receive an error message that instructs you to start service trace, or when directed to do so by your IBM Support Center

Use the **mqsichangetrace** command to start IBM Integration Bus service trace facilities.

You can select only one broker on each invocation of the command, but you can activate concurrent traces for more than one broker by invoking the command more than once.

To limit the scope of a trace, you must specify the individual broker that you want to trace.

If the trace cannot be associated with a specific component, the component name part of the file name is set to `utility`; for example, when tracing a command such as **mqsilist**, when no arguments are used.

To trace the **mqsichangeresourcestats**, **mqsicreateexecutiongroup**, **mqsideleteexecutiongroup**, **mqsideploy**, **mqsilist**, **mqsimode**, **mqsireloadsecurity**, **mqsireportresourcestats**, **mqsistartmsgflow**, and **mqsistopmsgflow** commands, use the **-v** parameter. This parameter takes an argument that is the name of the file to which trace records are written. For example, the following command outputs trace of the **mqsistartmsgflow** command to the specified file:

```
mqsistartmsgflow IB9NODE -e eg1 -m simpleflow -v .\trace.txt
```

If you want to trace the command executable files themselves, set the environment variables `MQSI_UTILITY_TRACE` and `MQSI_UTILITY_TRACESIZE` before you run the command.

MQSI_UTILITY_TRACE

Set this variable to `normal` for a basic level of trace, or `debug` for a fuller trace.

MQSI_UTILITY_TRACESIZE

The size, in kilobytes, of the binary trace file. See the **-c** parameter of the **mqsichangetrace** command.

Ensure that you reset these variables when the command that you are tracing has completed. If you do not do so, all subsequent commands are also traced, and their performance is therefore degraded.

To enable service trace of your CMP applications, take one of the following steps:

- Call the method `BrokerProxy.enableAdministrationAPITracing(String filename)`.
- Before running your CMP application, set the environment variable `MQSI_CMP_TRACE` to the name of the file to which trace is sent.

You can also use the IBM Integration Explorer to activate service trace on integration servers and message flows.

To enable service trace for integration servers or messages flows in the IBM Integration Explorer:

1. In the Navigator view, expand the Integration Nodes folder and right-click the integration server or message flow with which you want to work.
2. Click **Service Trace > Normal** or **Service Trace > Debug** to select the level of service trace that you require.

Example: starting service trace for the broker

About this task

To start debug level service tracing for the broker A on distributed systems, enter the following command:

```
mqsichangetrace BrokerA -t -b -l debug
```

where:

- t specifies service trace
- b specifies that trace for the agent subcomponent of the specified component is to be started
- l specifies the level of trace (in this case, debug)

z/OS On z/OS, enter the following command:

```
F MQPIBRK,ct t=yes, b=yes, l=debug
```

Example: starting service trace for an integration server

About this task

To start debug level service tracing for an integration server IS1 on broker A on distributed systems, enter the following command:

```
mqsichangetrace BrokerA -t -e IS1 -l debug -m fast -c 200000 -r
```

where:

- t specifies service trace
- l specifies the level of trace (in this case, debug)
- m specifies the way trace information is to be buffered (in this case, fast)
- c specifies the size of the trace file in KB (in this case, 200000)
- r specifies that the trace file is reset

z/OS On z/OS, enter the following command:

```
F MQPIBRK,ct t=yes, l=debug, m=fast, c=20000, r=yes
```

Related concepts:



IBM Integration Explorer

The IBM Integration Explorer is a graphical user interface based on the Eclipse platform for administering your integration nodes (brokers).



Trace

If you cannot get enough information about a particular problem from the entries that are available in the various logs, the next troubleshooting method to consider is using trace. Trace provides more details about what is happening while code runs. The information produced from trace is sent to a specified trace record, so that you or IBM support personnel can analyze it to discover the cause of your problem.

Related tasks:

Chapter 4, “Troubleshooting and support,” on page 647

If you are having problems with your message flow applications, use the techniques described in this section to help you to diagnose and solve the problems.

“Using trace” on page 845

You can use different types of trace to help you with problem determination and troubleshooting.



Starting user trace

Use user trace for debugging your applications; you can trace brokers, integration servers, and deployed message flows. Start user trace facilities using the **mqsichangetrace** command or the IBM Integration Explorer.

“Checking service trace options”

Use the **mqsireporttrace** command or the IBM Integration Explorer to check what tracing options are currently active for your brokers.

“Stopping service trace” on page 852

Use the **mqsichangetrace** command or the IBM Integration Explorer to stop an active service trace.

“Contacting your IBM Support Center” on page 876

If you cannot resolve problems that you find when you use IBM Integration Bus, or if you are directed to do so by an error message generated by IBM Integration Bus, you can request assistance from your IBM Support Center.

Related reference:



Service trace

Service trace is one of two types of optional trace that are available in IBM Integration Bus and provides more information than that provided by the entries that are written to the Event Log or to user trace. Service trace is inactive by default; you must activate it explicitly by using a command.



mqsichangetrace command

Use the **mqsichangetrace** command to set the tracing characteristics for a broker.

Checking service trace options

Use the **mqsireporttrace** command or the IBM Integration Explorer to check what tracing options are currently active for your brokers.

About this task

Specify the component for which the check is required. The command responds with the current trace status for the component that you have specified.

Example: checking service trace options for an integration server using the IBM Integration Explorer

About this task

To check what options are currently set for the integration server by using the IBM Integration Explorer:

Procedure

1. In the Navigator view, expand the Integration Nodes folder and right-click the integration server with which you want to work.
2. Click **Service** to view user trace settings for your integration server.

Example: checking service trace options for a broker

About this task

To check what options are currently set for the broker, on distributed systems, enter the following command:

```
mqsireporttrace brokerA -t
```

where **-t** specifies service trace.

z/OS

On z/OS, enter the following command:

```
F MQP1BRK,reporttrace t=yes
```

Results

If you have started tracing by following the example in “Starting service trace” on page 847, the response to the **mqsireporttrace** command is:

```
BIP8098I: Trace level: debug, mode: safe, size: 1024 KB  
BIP8071I: Successful command completion
```

Related concepts:



IBM Integration Explorer

The IBM Integration Explorer is a graphical user interface based on the Eclipse platform for administering your integration nodes (brokers).



Trace

If you cannot get enough information about a particular problem from the entries that are available in the various logs, the next troubleshooting method to consider is using trace. Trace provides more details about what is happening while code runs. The information produced from trace is sent to a specified trace record, so that you or IBM support personnel can analyze it to discover the cause of your problem.

Related tasks:

Chapter 4, “Troubleshooting and support,” on page 647

If you are having problems with your message flow applications, use the techniques described in this section to help you to diagnose and solve the problems.

“Using trace” on page 845

You can use different types of trace to help you with problem determination and troubleshooting.



Checking user trace options

Use the **mqsireporttrace** command or the IBM Integration Explorer to check what tracing options are currently active for your brokers, integration servers and message flows.

“Starting service trace” on page 847

Service trace is used to get detailed information about your environment for use by your IBM Support Center.

“Changing service trace options”

Use the **mqsichangetrace** command or the IBM Integration Explorer to change the service trace options that you have set.

Related reference:



Service trace

Service trace is one of two types of optional trace that are available in IBM Integration Bus and provides more information than that provided by the entries that are written to the Event Log or to user trace. Service trace is inactive by default; you must activate it explicitly by using a command.



mqsireporttrace command

Use the **mqsireporttrace** command to display the trace options currently in effect. Trace can be run only against a broker, or any of its resources.

Changing service trace options

Use the **mqsichangetrace** command or the IBM Integration Explorer to change the service trace options that you have set.

Example: changing service trace from debug to normal on an integration server using the IBM Integration Explorer

About this task

To change from a debug level of trace to a normal level on the integration server using the IBM Integration Explorer:

Procedure

1. In the Navigator view, expand the Integration Nodes folder and right-click the integration server with which you want to work.
2. Click **Service > Normal**.

Example: changing service trace from debug to normal

About this task

To change from a debug level of trace to a normal level on the broker, on distributed systems, enter the following command:

```
mqsichangetrace BrokerA -t -b -l normal
```

where:

-t specifies service trace

-b specifies that tracing for the agent subcomponent of the specified component is to be changed

-l specifies the level of trace (in this case, changing it to normal)

z/OS

On z/OS, enter the following command:

F MQP1BRK,ct t=yes, b=yes, l=normal

Related concepts:

IBM Integration Explorer

The IBM Integration Explorer is a graphical user interface based on the Eclipse platform for administering your integration nodes (brokers).

Trace

If you cannot get enough information about a particular problem from the entries that are available in the various logs, the next troubleshooting method to consider is using trace. Trace provides more details about what is happening while code runs. The information produced from trace is sent to a specified trace record, so that you or IBM support personnel can analyze it to discover the cause of your problem.

Related tasks:

Chapter 4, “Troubleshooting and support,” on page 647

If you are having problems with your message flow applications, use the techniques described in this section to help you to diagnose and solve the problems.

“Using trace” on page 845

You can use different types of trace to help you with problem determination and troubleshooting.

Changing user trace options

Use the **mqsi changetrace** command to change the trace options that you have set. You can also use the IBM Integration Explorer to change the trace options for integration servers and assigned message flows.

“Starting service trace” on page 847

Service trace is used to get detailed information about your environment for use by your IBM Support Center.

“Checking service trace options” on page 849

Use the **mqsi reporttrace** command or the IBM Integration Explorer to check what tracing options are currently active for your brokers.

Related reference:

Service trace

Service trace is one of two types of optional trace that are available in IBM Integration Bus and provides more information than that provided by the entries that are written to the Event Log or to user trace. Service trace is inactive by default; you must activate it explicitly by using a command.

mqsi changetrace command

Use the **mqsi changetrace** command to set the tracing characteristics for a broker.

Stopping service trace

Use the **mqsi changetrace** command or the IBM Integration Explorer to stop an active service trace.

Procedure

- Use the **mqsi changetrace** command with a trace level of none to stop an active trace. This action stops the trace activity for the component that you specify on the command. It does not affect active traces on other components. For example, if you stop tracing on the integration server test, an active trace on another integration server continues.

- Click **Service Trace > None** on an integration server or message flow, in the IBM Integration Explorer, to stop an active service trace on the selected object.

Results

If you redeploy a component from the IBM Integration Toolkit, trace for that component is returned to its default setting of none.

Example: stopping service trace on an integration server, using the IBM Integration Explorer **About this task**

To stop the service trace using the IBM Integration Explorer:

Procedure

1. In the Navigator view, expand the Integration Nodes folder and right-click the integration server with which you want to work.
2. Click **Service TraceNone**.

Example: stopping service trace on an integration server, using the mqsichangetrace command **About this task**

To stop debug level service tracing for an integration server IS1 on broker A on distributed systems, enter the command

```
mqsichangetrace BrokerA -t -e IS1 -l none
```

where:

- t specifies service trace
- l specifies the level of trace (in this case, none)

z/OS On z/OS, enter the command
F MQPIBRK,ct t=yes, b=yes, l=none

Example: stopping service trace on the broker **About this task**

To stop the trace started by the command shown in “Starting service trace” on page 847, on distributed systems, enter the following command:

```
mqsichangetrace BrokerA -t -b -l none
```

where:

- t specifies service trace
- b specifies that trace for the agent subcomponent of the specified component is to be stopped
- l specifies the level of trace (in this case, none)

z/OS On z/OS, enter the following command:
F MQPIBRK,ct t=yes, b=yes, l=none

Related concepts:



IBM Integration Explorer

The IBM Integration Explorer is a graphical user interface based on the Eclipse platform for administering your integration nodes (brokers).



Trace

If you cannot get enough information about a particular problem from the entries that are available in the various logs, the next troubleshooting method to consider is using trace. Trace provides more details about what is happening while code runs. The information produced from trace is sent to a specified trace record, so that you or IBM support personnel can analyze it to discover the cause of your problem.

Related tasks:

Chapter 4, “Troubleshooting and support,” on page 647

If you are having problems with your message flow applications, use the techniques described in this section to help you to diagnose and solve the problems.

“Using trace” on page 845

You can use different types of trace to help you with problem determination and troubleshooting.



Stopping user trace

Use user trace for debugging your applications; you can trace brokers, integration servers, and deployed message flows. Stop user trace facilities by using the **mqsichangetrace** command or the IBM Integration Explorer.

“Starting service trace” on page 847

Service trace is used to get detailed information about your environment for use by your IBM Support Center.

“Clearing old information from trace files” on page 861

If the component that you are tracing has stopped, you can delete its trace files from the log subdirectory of the IBM Integration Bus home directory.

Related reference:



Service trace

Service trace is one of two types of optional trace that are available in IBM Integration Bus and provides more information than that provided by the entries that are written to the Event Log or to user trace. Service trace is inactive by default; you must activate it explicitly by using a command.



mqsichangetrace command

Use the **mqsichangetrace** command to set the tracing characteristics for a broker.

Retrieving service trace

Use the **mqsireadlog** command to access the trace information recorded by the service trace facilities.

Procedure

This command retrieves the trace details according to parameters that you specify on the command, and writes the requested records to a file, or to the command-line window, in XML format.

Example: retrieving service trace information in XML format Procedure

To retrieve information for the service trace activated with the `mqsichangetrace` command, and write it to an output file, on distributed systems, enter the following command:

```
mqsireadlog brokerName -t -b agent -f -o /path/to/strace.xml
```

where:

- t** specifies service trace.
- b** *agent* specifies that trace for the agent subcomponent of the specified component is to be retrieved.
- f** specifies that the log file is to be read directly from the file system (this flag is mandatory for service trace).
- o** specifies the output file (in this case, `/path/to/strace.xml`.) If you do not specify a path to the file, the file is created in the current directory. Ensure that there is enough space for the output file.

Results

This command sends a log request to the broker to retrieve the service trace log, and stores the responses in the specified file. You can view this file using a plain text editor.

Example: retrieving service trace information in XML format for an integration server Procedure

To retrieve information for the service trace activated with the `mqsichangetrace` command and associated with integration server IS1 on broker A, and write it to an output file, on distributed systems, enter the following command:

```
mqsireadlog BrokerA -t -e IS1 -f -o /path/to/strace.xml
```

where:

- t** specifies service trace.
- f** specifies that the log file is to be read directly from the file system (this flag is mandatory for service trace).
- o** specifies the output file (in this case, `/path/to/strace.xml`.) If you do not specify a path to the file, the file is created in the current directory. Ensure that there is enough space for the output file.

Results

This command sends a log request to the broker to retrieve the service trace log, and stores the responses in the specified file. You can view this file using a plain text editor.

Related concepts:



Trace

If you cannot get enough information about a particular problem from the entries that are available in the various logs, the next troubleshooting method to consider is using trace. Trace provides more details about what is happening while code runs. The information produced from trace is sent to a specified trace record, so that you or IBM support personnel can analyze it to discover the cause of your problem.

Related tasks:

Chapter 4, “Troubleshooting and support,” on page 647

If you are having problems with your message flow applications, use the techniques described in this section to help you to diagnose and solve the problems.

“Using trace” on page 845

You can use different types of trace to help you with problem determination and troubleshooting.



Retrieving user trace

Use the **mqsireadlog** command to access the trace information that is recorded by the user trace facilities.

“Formatting trace”

Use the **mqsiformatlog** command to format trace information.

“Interpreting trace” on page 859

Use the information in a formatted trace file to identify unexpected behavior.

Related reference:



Service trace

Service trace is one of two types of optional trace that are available in IBM Integration Bus and provides more information than that provided by the entries that are written to the Event Log or to user trace. Service trace is inactive by default; you must activate it explicitly by using a command.



mqsireadlog command

Use the **mqsireadlog** command to retrieve trace records for the specified component.

Formatting trace

Use the **mqsiformatlog** command to format trace information.

About this task

The trace information that is generated by the **mqsireadlog** command is not easy to read unless you use an XML viewer (such as an Internet browser) or an XML editor that understands the document type descriptor (DTD) in the file.

Procedure

IBM Integration Bus provides the command **mqsiformatlog** to format the trace information to a flat file, so that you can view it using a text editor.

Results

The **mqsiformatlog** command takes a file generated by the **mqsireadlog** command as input, and flattens the XML log into structured records. It also retrieves the inserts for the XML message in your current locale. You can direct the formatted output to a file, or view it in the command line window.

Each trace entry contains a time stamp and an IBM Integration Bus message that contains a number (for example, BIP2622) and a text string containing variable inserts.

Example: formatting service trace information for an integration server

About this task

To format the service trace file for an integration server, enter the command

```
mqsiformatlog -i </path/to/output.xml> -o </path/to/output.txt>
```

where:

- i specifies the path to the input file that contains the unformatted service trace information for the integration server
- o specifies the path to the output file that is to contain the formatted service trace information

Example: formatting user trace information on Windows

About this task

Windows On Windows, to format the trace file that is created in Starting user trace, enter the command

```
mqsiformatlog -i trace.xml -o formattrace.log
```

where:

- i specifies the input file (in this case, trace.xml)
- o specifies the output file (in this case, formattrace.log)

This command reads the trace information in the file trace.xml, formats it, and writes it to the file formattrace.log. The following example shows a portion of the output of the **mqsiformatlog** command for a normal level trace file.

Timestamps are formatted in local time, 330 minutes past GMT.

```
2003-06-19 11:30:29.795999      2852  UserTrace  BIP2632I: Message received and
propagated to 'out' terminal of MQ Input node 'Video_Test.VIDEO_XML_IN'.
2003-06-19 11:30:29.795999      2852  UserTrace  BIP6060I: Parser type 'Properties'
created on behalf of node 'Video_Test.VIDEO_XML_IN' to handle portion of incoming
message of length
0 bytes beginning at offset '0'.
2003-06-19 11:30:29.795999      2852  UserTrace  BIP6061I: Parser type 'MQMD'
created on behalf of node 'Video_Test.VIDEO_XML_IN' to handle portion of incoming
message of length '364' bytes beginning at offset '0'. Parser type selected based
on value 'MQHMD' from previous parser.
2003-06-19 11:30:29.795999      2852  UserTrace  BIP6061I: Parser type 'MRM'
created on behalf of node 'Video_Test.VIDEO_XML_IN' to handle portion of incoming
message of length '650' bytes beginning at offset '364'. Parser type selected based
on value 'MRM' from previous parser.
2003-06-19 11:30:29.795999      2852  UserTrace  BIP2537I: Node 'Video_Test.Extract
Fields':
Executing statement 'BEGIN ... END;' at (.Video_Test_Compute.Main, 2.2).
2003-06-19 11:30:29.795999      2852  UserTrace  BIP2537I: Node 'Video_Test.Extract
Fields':
Executing statement 'SET OutputRoot = InputRoot;' at (.Video_Test_Compute.Main, 7.3).
2003-06-19 11:30:29.795999      2852  UserTrace  BIP2538I: Node 'Video_Test.Extract
Fields':
Evaluating expression 'InputRoot' at (.Video_Test_Compute.Main, 7.20).
2003-06-19 11:30:29.795999      2852  UserTrace  BIP2568I: Node 'Video_Test.Extract
Fields':
```

Performing tree copy of 'InputRoot' to 'OutputRoot'.

...

```
2003-06-19 11:30:29.827999    2852  UserTrace  BIP4124I: Message propagated to
'out' terminal of Compute node 'Video_Test.Extract Fields'.
2003-06-19 11:30:29.827999    2852  UserTrace  BIP2638I: The MQ Output node
'Video_Test.VIDEO_OUT' attempted to write a message to queue 'VIDEO_OUT' connected
to queue manager ''. The MQCC was '0' and the MQRC was '0'.
2003-06-19 11:30:29.827999    2852  UserTrace  BIP2622I: Message successfully
output by output node 'Video_Test.VIDEO_OUT' to queue 'VIDEO_OUT' on queue manager ''.
```

Threads encountered in this trace:
2852

Related concepts:



Trace

If you cannot get enough information about a particular problem from the entries that are available in the various logs, the next troubleshooting method to consider is using trace. Trace provides more details about what is happening while code runs. The information produced from trace is sent to a specified trace record, so that you or IBM support personnel can analyze it to discover the cause of your problem.

Related tasks:

Chapter 4, “Troubleshooting and support,” on page 647

If you are having problems with your message flow applications, use the techniques described in this section to help you to diagnose and solve the problems.

“Using trace” on page 845

You can use different types of trace to help you with problem determination and troubleshooting.



Retrieving user trace

Use the **mqsireadlog** command to access the trace information that is recorded by the user trace facilities.

“Retrieving service trace” on page 854

Use the **mqsireadlog** command to access the trace information recorded by the service trace facilities.

“Interpreting trace” on page 859

Use the information in a formatted trace file to identify unexpected behavior.

“Clearing old information from trace files” on page 861

If the component that you are tracing has stopped, you can delete its trace files from the log subdirectory of the IBM Integration Bus home directory.

Related reference:



User trace

User trace is one of two types of optional trace that are available in IBM Integration Bus and provides more information than that provided by the entries that are written to the Administration log. User trace is inactive by default; you must activate it explicitly by using a command, or by selecting options in the IBM Integration Toolkit.



Service trace

Service trace is one of two types of optional trace that are available in IBM Integration Bus and provides more information than that provided by the entries that are written to the Event Log or to user trace. Service trace is inactive by default; you must activate it explicitly by using a command.



`mqsiformatlog` command

Use the `mqsiformatlog` command to process the XML log created by `mqsireadlog`. The command retrieves and formats any messages that the XML log contains into a form suitable for the locale of the user who runs the command.

Interpreting trace

Use the information in a formatted trace file to identify unexpected behavior.

About this task

A formatted log file, like the one in “Formatting trace” on page 856, contains a sequence of IBM Integration Bus messages. These messages record the activity in a specific part of the system (the part that you identify when you start the trace). You can use this sequence to understand what is happening, and to check that the behavior that is recorded is what you are expecting.

For example, message flow trace records the path that a message takes through the message flow. You can see why decisions result in this path (where a choice is available).

Procedure

1. Verify that the trace file is complete.

If the size of the trace log is too small to contain all events, trace output continues at the start of the trace log, overwriting existing entries. This is known as *wrapping*.

An indication that a trace has wrapped is the relationship between the first and last timestamp in it, and the time that trace was enabled. For example, assume that you start tracing at 10:15, and collect the trace at 10:30. If the trace timestamps run from 10:20 to 10:30, it is probable that trace wrapped. Of course it could mean that nothing happened between 10:15 and 10:20.

Examine the trace and decide whether the beginning of it makes sense, and whether it looks complete. For example, if you want to trace the passage of three messages through a flow, and trace starts half way through the second message, it could have wrapped, or trace might not have been enabled early enough.

2. If trace has wrapped, increase the size of the trace file, and rerun trace. For information on trace settings, see `mqsichangetrace` command.
3. If you see unexpected behavior in a message flow or integration server, use this trace information to check the actions that have been taken and identify the source of an error or other discrepancy.

Results

The messages contain identifiers for the resources that are being traced, for example the integration servers and message flows. The identifier that is given is typically the label (the name) that you gave to the resource when you defined it.

Here is an extract from a user trace file. In the example, each column has been labeled:

Timestamp	Thread ID	Trace type	Message
2005-07-12 16:17:18.242605	5344	UserTrace	BIP2537I: Node 'Reply.MapToRequestor': Executing statement ''SET I = I + 1;' at ('.MapToRequestor.CopyMessageHeaders', '6.4').
2005-07-12 16:17:18.242605	5344	UserTrace	BIP2539I: Node 'Reply.MapToRequestor': Evaluating expression ''I'' at ('.MapToRequestor.CopyMessageHeaders', '6.12'). This resolved to ''I''. The result was ''1''.
2005-07-12 16:17:18.242605	5344	UserTrace	BIP2539I: Node 'Reply.MapToRequestor': Evaluating expression ''I + 1'' at ('.MapToRequestor.CopyMessageHeaders', '6.14'). This resolved to ''1 + 1''. The result was ''2''.
2005-07-12 16:17:18.242605	5344	UserTrace	BIP2566I: Node 'Reply.MapToRequestor': Assigning value ''2'' to field / variable ''I''.

References such as '6.12' apply to the row and column number within a function that specify the location of the command that is being executed; in this case, row 6, column 12.

Related concepts:



Trace

If you cannot get enough information about a particular problem from the entries that are available in the various logs, the next troubleshooting method to consider is using trace. Trace provides more details about what is happening while code runs. The information produced from trace is sent to a specified trace record, so that you or IBM support personnel can analyze it to discover the cause of your problem.

Related tasks:

Chapter 4, “Troubleshooting and support,” on page 647

If you are having problems with your message flow applications, use the techniques described in this section to help you to diagnose and solve the problems.

“Using trace” on page 845

You can use different types of trace to help you with problem determination and troubleshooting.



Retrieving user trace

Use the **mqsi readlog** command to access the trace information that is recorded by the user trace facilities.

“Retrieving service trace” on page 854

Use the **mqsi readlog** command to access the trace information recorded by the service trace facilities.

“Formatting trace” on page 856

Use the **mqsi formatlog** command to format trace information.

“Resolving problems with user-defined extensions” on page 829

Advice for dealing with some common problems that can arise when you work with user-defined extensions

Related reference:



User trace

User trace is one of two types of optional trace that are available in IBM Integration Bus and provides more information than that provided by the entries that are written to the Administration log. User trace is inactive by default; you must activate it explicitly by using a command, or by selecting options in the IBM Integration Toolkit.



Service trace

Service trace is one of two types of optional trace that are available in IBM Integration Bus and provides more information than that provided by the entries that are written to the Event Log or to user trace. Service trace is inactive by default; you must activate it explicitly by using a command.

Clearing old information from trace files

If the component that you are tracing has stopped, you can delete its trace files from the log subdirectory of the IBM Integration Bus home directory.

About this task

If you are tracing an integration server, you can use the **-r** parameter of the **mqsichangetrace** command to reset (clear) the trace log (the **-r** parameter can be specified only if you specify the **-e** parameter). You might clear the log when you start a new trace, to ensure that all the records on the log are unique to the new trace.

Example: clearing the user trace log for the default integration server

About this task

To clear the user trace log for the default integration server, on distributed systems, enter the command: **mqsichangetrace WBRK_BROKER -u -e default -r** where:

WBRK_BROKER specifies the name of the broker

-u specifies user trace

-e specifies the integration server (in this case the default integration server)

-r clears the trace log

z/OS

On z/OS, enter the command **F MQP1BRK,ct u=yes, e='default', r=yes**

Related concepts:



Trace

If you cannot get enough information about a particular problem from the entries that are available in the various logs, the next troubleshooting method to consider is using trace. Trace provides more details about what is happening while code runs. The information produced from trace is sent to a specified trace record, so that you or IBM support personnel can analyze it to discover the cause of your problem.

Related tasks:

Chapter 4, “Troubleshooting and support,” on page 647

If you are having problems with your message flow applications, use the techniques described in this section to help you to diagnose and solve the problems.

“Using trace” on page 845

You can use different types of trace to help you with problem determination and

troubleshooting.



Starting user trace

Use user trace for debugging your applications; you can trace brokers, integration servers, and deployed message flows. Start user trace facilities using the **mqsichangetrace** command or the IBM Integration Explorer.

“Starting service trace” on page 847

Service trace is used to get detailed information about your environment for use by your IBM Support Center.

Related reference:



Local error logs

IBM Integration Bus components use the local error log (also known as the system log) to record information about major activities within the system. When an error occurs, check the local error log first.



User trace

User trace is one of two types of optional trace that are available in IBM Integration Bus and provides more information than that provided by the entries that are written to the Administration log. User trace is inactive by default; you must activate it explicitly by using a command, or by selecting options in the IBM Integration Toolkit.



Service trace

Service trace is one of two types of optional trace that are available in IBM Integration Bus and provides more information than that provided by the entries that are written to the Event Log or to user trace. Service trace is inactive by default; you must activate it explicitly by using a command.



mqsichangetrace command

Use the **mqsichangetrace** command to set the tracing characteristics for a broker.

Changing trace settings from the IBM Integration Explorer

Collect trace information, in addition to user and service trace, by selecting options in the IBM Integration Explorer.

About this task

- Trace for the integration node and IBM Integration Explorer components.
- Trace for the IBM Integration API (also known as the CMP).

The following sections tell you how to change the settings for these types of trace.

Changing the WebSphere MQ Java Client trace settings

About this task

You control tracing of the WebSphere MQ Java Client from the IBM Integration Explorer:

Procedure

1. Click **Window > Preferences**.
2. In the left menu, expand **IBM Integration**.
3. Click **Service Trace**.
4. In the **WebSphere MQ Java Client** section, select the check box to enable tracing on the WebSphere MQ Java Client. The default file to which trace is

- written is C:\Documents and Settings*userid*\Application Data\IBM\MQ Explorer\.metadata\MQClientTraceEnabled.log where *userid* is your user ID.
5. Click **OK** to apply your changes and to close the Preferences window.

Changing the CMP trace settings

About this task

You control tracing of the CMP from the IBM Integration Explorer:

Procedure

1. Click **Window > Preferences**.
2. In the left menu, expand **IBM Integration**.
3. Click **Service Trace**.
4. In the **Integration API** section, select the check box to enable tracing on the CMP. The default file to which trace is written is C:/Documents and Settings/*userid*/Application Data/IBM/MQ Explorer/.metadata/AdminAPITrace.log where *userid* is your user ID.
5. Click **OK** to apply your changes and to close the Preferences window.

Changing the IBM Integration Explorer trace settings

About this task

You can control tracing of the IBM Integration Explorer using the IBM Integration Preferences page:

Procedure

1. Click **Window > Preferences**.
2. In the left menu, expand **IBM Integration**.
3. Click **Service Trace**.
4. Select the check box for the component that you want to trace, and select a location for the log file for the trace. The default location for the trace files in is the WebSphere MQ Explorer workspace directory.
5. Set a value for the log file size in the Maximum Trace File Size field. When the size of the log file you set is exceeded, the original log file is copied to *filename.ext.bak*. A new log file with the selected name is started.
6. Click **OK** to apply your changes and to close the Preferences window.

Related concepts:



Trace

If you cannot get enough information about a particular problem from the entries that are available in the various logs, the next troubleshooting method to consider is using trace. Trace provides more details about what is happening while code runs. The information produced from trace is sent to a specified trace record, so that you or IBM support personnel can analyze it to discover the cause of your problem.



The IBM Integration API

The IBM Integration API is a programming interface that your applications can use to control brokers and their resources through a remote interface.

Related tasks:

“Using trace” on page 845

You can use different types of trace to help you with problem determination and

troubleshooting.



Starting user trace

Use user trace for debugging your applications; you can trace brokers, integration servers, and deployed message flows. Start user trace facilities using the **mqsichangetrace** command or the IBM Integration Explorer.

“IBM Integration API (CMP) trace” on page 867

Enable or disable service trace for the IBM Integration API (also known as the CMP).

Related reference:



User trace

User trace is one of two types of optional trace that are available in IBM Integration Bus and provides more information than that provided by the entries that are written to the Administration log. User trace is inactive by default; you must activate it explicitly by using a command, or by selecting options in the IBM Integration Toolkit.



Service trace

Service trace is one of two types of optional trace that are available in IBM Integration Bus and provides more information than that provided by the entries that are written to the Event Log or to user trace. Service trace is inactive by default; you must activate it explicitly by using a command.



IBM Integration Toolkit

The IBM Integration Toolkit provides your application development environment on Windows and Linux on x86.

ODBC trace

You can use various methods to trace for ODBC activity, depending on the operating system that you are using.

About this task

Windows

For Windows, use the **Tracing** tab of the ODBC function:

1. Click **Start > Settings > Control Panel > Administrative Tools**.
2. Double-click **Data Sources**.
3. Click the **Tracing** tab.
4. Click **Start Tracing Now**.
5. Click **OK**.

To stop ODBC tracing, on the **Tracing** tab, click **Stop Tracing Now**, then **OK**.

Linux

UNIX

For Linux and UNIX operating systems using IBM Integration ODBC Database Extender drivers:

- To initiate trace for ODBC activity, edit the [ODBC] stanza in the `odbcinst.ini` file in the directory pointed to by your `ODBCSYSINI` environment variable as follows:
 1. Change `Trace=no` to `Trace=yes`.
 2. Specify a path and file name for `TraceFile`
 3. Ensure that the `TraceFile` entry points to a file system that has enough space to receive the trace output

z/OS

For z/OS, to initiate application trace for ODBC activity:

1. Edit the BIPDSNA0 file in the component dataset and under the stanza entry [COMMON], change APPLTRACE=0 to APPLTRACE=1
2. If the [COMMON] stanza in the BIPDSNA0 member does not include the TRACEPIDTID parameter, then set TRACEPIDTID=1 to enable the process/thread IDs in the ODBC trace.
3. Remove the comment from the COMPDIR variable declaration and the APPLTRC DD from the steps EGN0ENV and EGENV, in the IBM Integration Bus started task JCL.
4. Stop and restart the broker after you have made all the changes to the BIPDSNA0 file and the started task JCL.

By default, the trace output file is written to <component_HFS>/output/, into a file called db2app1trace.. Each address space has a unique number, and the eight character integration server label appended to the end of db2app1trace.

This unique number, appended to the ODBC file, is the SE number in the integration server address space JOBLOG.

If the eight character integration server label is not unique across multiple integration servers, look for the value of SE in the JOBLOG for which you want to view the ODBC trace, and find the file that specifies this value.

Results

Example

DB2 on IBM Integration Bus for z/OS

The following sample ODBC trace files show the layout of a trace file, together with some examples of successful and error returns. The general layout of each group in an ODBC file is that:

- Each line is preceded by a process/thread ID and time stamp.
- The first line displays what the call does.
- The second line displays the return.
- The third line displays the result.

The first trace file shows a trace where a call fails because an object does not have the correct authority to perform an action:

```
[000207A9 0000000000000021] [2008-09-24 15:49:20.544123] SQLAllocStmt( hDbc=2, phStmt=&1c7f9554 )
[000207A9 0000000000000021] [2008-09-24 15:49:20.544156] SQLAllocStmt( phStmt=1 )
[000207A9 0000000000000021] [2008-09-24 15:49:20.544163] ----> SQL_SUCCESS

[000207A9 0000000000000021] [2008-09-24 15:49:20.544179] SQLFreeStmt( hStmt=1, fOption=SQL_CLOSE )
[000207A9 0000000000000021] [2008-09-24 15:49:20.544189] SQLFreeStmt( )
[000207A9 0000000000000021] [2008-09-24 15:49:20.544194] ----> SQL_SUCCESS

[000207A9 0000000000000021] [2008-09-24 15:49:20.544205] SQLPrepare( hStmt=1 )
[000207A9 0000000000000021] [2008-09-24 15:49:20.544212] ( pszSqlStr="SELECT TESTTABLE.ID FROM
WMQI77.TESTTABLE TESTTABLE", cbSqlStr=-3 )
[000207A9 0000000000000021] [2008-09-24 15:49:20.587083] SQLPrepare( )
[000207A9 0000000000000021] [2008-09-24 15:49:20.587101] ----> SQL_ERROR

[000207A9 0000000000000021] [2008-09-24 15:49:20.587157] SQLError( hEnv=0, hDbc=0, hStmt=1,
pszSqlState=&3902af28, pfNativeError=&3902af24, pszErrorMsg=&1b88b0b0,
cbErrorMsgMax=1024, pcbErrorMsg=&3902aefc )
[000207A9 0000000000000021] [2008-09-24 15:49:20.587190] SQLError( pszSqlState="42501", pfNativeError=-551,
pszErrorMsg="(DB2 FOR OS/390){ODBC DRIVER}{DSN09015}
DSNT408I SQLCODE = -551, ERROR: WMQI83 DOES NOT HAVE THE PRIVILEGE TO PERFORM
OPERATION SELECT ON OBJECT WMQI77.TESTTABLE
DSNT418I SQLSTATE = 42501 SQLSTATE RETURN CODE
DSNT415I SQLERRP = DSNXOSC SQL PROCEDURE DETECTING ERROR
DSNT416I SQLERRD = -100 0 0 -1 0 0 SQL DIAGNOSTIC INFORMATION
DSNT416I SQLERRD = X'FFFFFFFF9C' X'00000000' X'00000000' X'FFFFFFFF'
X'00000000' X'00000000' SQL DIAGNOSTIC INFORMATION
ERRLOC=1:13:2", pcbErrorMsg=623 )
```

```

[000207A9 0000000000000021] [2008-09-24 15:49:20.587666] ----> SQL_SUCCESS

[000207A9 0000000000000021] [2008-09-24 15:49:20.587725] SQLError( hEnv=0, hDbc=0, hStmt=1,
pszSqlState=&3902af28, pfNativeError=&3902af24, pszErrorMsg=&1b88b0b0,
cbErrorMsgMax=1024, pcbErrorMsg=&3902aefc )
[000207A9 0000000000000021] [2008-09-24 15:49:20.587752] SQLError( )
[000207A9 0000000000000021] [2008-09-24 15:49:20.587757] ----> SQL_NO_DATA_FOUND

[000207A9 0000000000000021] [2008-09-24 15:49:20.588049] SQLFreeStmt( hStmt=1, fOption=SQL_DROP )
[000207A9 0000000000000021] [2008-09-24 15:49:20.588075] SQLFreeStmt( )
[000207A9 0000000000000021] [2008-09-24 15:49:20.588080] ----> SQL_SUCCESS

[000207A9 0000000000000021] [2008-09-24 15:49:20.593800] SQLTransact( hEnv=1, hDbc=0, fType=SQL_COMMIT )
[000207A9 0000000000000021] [2008-09-24 15:49:20.593887] SQLTransact( )
[000207A9 0000000000000021] [2008-09-24 15:49:20.593893] ----> SQL_SUCCESS

```

The second trace file shows the same trace file with the operation working:

```

[000207A9 0000000000000021] [2008-09-24 16:00:25.287052] SQLAllocStmt( hDbc=1, phStmt=&1c7f8e54 )
[000207A9 0000000000000021] [2008-09-24 16:00:25.287068] SQLAllocStmt( phStmt=1 )
[000207A9 0000000000000021] [2008-09-24 16:00:25.287075] ----> SQL_SUCCESS

[000207A9 0000000000000021] [2008-09-24 16:00:25.287088] SQLFreeStmt( hStmt=1, fOption=SQL_CLOSE )
[000207A9 0000000000000021] [2008-09-24 16:00:25.287098] SQLFreeStmt( )
[000207A9 0000000000000021] [2008-09-24 16:00:25.287104] ----> SQL_SUCCESS

[000207A9 0000000000000021] [2008-09-24 16:00:25.287114] SQLPrepare( hStmt=1 )
[000207A9 0000000000000021] [2008-09-24 16:00:25.287121] ( pszSqlStr="SELECT TESTTABLE.ID FROM
WMQ177.TESTTABLE TESTTABLE", cbSqlStr=-3 )
[000207A9 0000000000000021] [2008-09-24 16:00:25.302484] SQLPrepare( )
[000207A9 0000000000000021] [2008-09-24 16:00:25.302510] ----> SQL_SUCCESS

[000207A9 0000000000000021] [2008-09-24 16:00:25.302539] SQLFreeStmt( hStmt=1,
fOption=SQL_CLOSE )
[000207A9 0000000000000021] [2008-09-24 16:00:25.302555] SQLFreeStmt( )
[000207A9 0000000000000021] [2008-09-24 16:00:25.302560] ----> SQL_SUCCESS

[000207A9 0000000000000021] [2008-09-24 16:00:25.302573] SQLExecute( hStmt=1 )
[000207A9 0000000000000021] [2008-09-24 16:00:25.302622] SQLExecute( )
[000207A9 0000000000000021] [2008-09-24 16:00:25.302628] ----> SQL_SUCCESS

[000207A9 0000000000000021] [2008-09-24 16:00:25.302660] SQLNumResultCols( hStmt=1,
pcCol=&3902c7fa )
[000207A9 0000000000000021] [2008-09-24 16:00:25.302672] SQLNumResultCols( pcCol=1 )
[000207A9 0000000000000021] [2008-09-24 16:00:25.302679] ----> SQL_SUCCESS

[000207A9 0000000000000021] [2008-09-24 16:00:25.302697] SQLDescribeCol( hStmt=1, iCol=1,
pszColName=&3902cb10, cbColNameMax=200, pcbColName=&3902c804,
pfSQLType=&3902c802, pcbColDef=&3902c858, pibScale=&3902c800,
pfNullable=&3902c7fe )
[000207A9 0000000000000021] [2008-09-24 16:00:25.302733] SQLDescribeCol( pszColName="ID",
pcbColName=2, pfSQLType=SQL_CHAR, pcbColDef=10, pibScale=0,
pfNullable=SQL_NULLABLE )
[000207A9 0000000000000021] [2008-09-24 16:00:25.302819] ----> SQL_SUCCESS

[000207A9 0000000000000021] [2008-09-24 16:00:25.302826] SQLColAttribute( hStmt=1, iCol=1,
fDescType=SQL_DESC_OCTET_LENGTH, rgbDesc=NULL, cbDescMax=0,
pcbDesc=NULL, pfDesc=&3902c864 )
[000207A9 0000000000000021] [2008-09-24 16:00:25.302850] SQLColAttribute( pfDesc=10 )
[000207A9 0000000000000021] [2008-09-24 16:00:25.302857] ----> SQL_SUCCESS

[000207A9 0000000000000021] [2008-09-24 16:00:25.302866] SQLBindCol( hStmt=1, iCol=1,
fCType=SQL_C_CHAR, rgbValue=&1b48829c, cbValueMax=12,
pcbValue=&1b488298 )
[000207A9 0000000000000021] [2008-09-24 16:00:25.302888] SQLBindCol( )
[000207A9 0000000000000021] [2008-09-24 16:00:25.302894] ----> SQL_SUCCESS

[000207A9 0000000000000021] [2008-09-24 16:00:25.302901] SQLSetStmtAttr( hStmt=1,
fAttribute=SQL_ATTR_ROW_BIND_TYPE, pvParam=&10, iStrLen=0 )
[000207A9 0000000000000021] [2008-09-24 16:00:25.302917] SQLSetStmtAttr( )
[000207A9 0000000000000021] [2008-09-24 16:00:25.302922] ----> SQL_SUCCESS

[000207A9 0000000000000021] [2008-09-24 16:00:25.302928] SQLSetStmtAttr( hStmt=1,
fAttribute=Unknown value 9, pvParam=&20, iStrLen=0 )
[000207A9 0000000000000021] [2008-09-24 16:00:25.302943] SQLSetStmtAttr( )
[000207A9 0000000000000021] [2008-09-24 16:00:25.302949] ----> SQL_SUCCESS

[000207A9 0000000000000021] [2008-09-24 16:00:25.302956] SQLExtendedFetch( hStmt=1,
fFetchType=SQL_FETCH_NEXT, iRow=0, pcRow=&1c7f6894,
rgfRowStatus=&1bca17d0 )

```

```
[000207A9 0000000000000021] [2008-09-24 16:00:25.317947] ( Row=1, iCol=1, fCType=SQL_C_CHAR,
rgbValue="TABLG      ", pcbValue=10 )
[000207A9 0000000000000021] [2008-09-24 16:00:25.317980] ( Row=2, iCol=1, fCType=SQL_C_CHAR,
rgbValue="TABLF      ", pcbValue=10 )
[000207A9 0000000000000021] [2008-09-24 16:00:25.318001] ( Row=3, iCol=1, fCType=SQL_C_CHAR, r
gbValue="TABLE      ", pcbValue=10 )
[000207A9 0000000000000021] [2008-09-24 16:00:25.318022] ( Row=4, iCol=1, fCType=SQL_C_CHAR,
rgbValue="TABLD      ", pcbValue=10 )
[000207A9 0000000000000021] [2008-09-24 16:00:25.318044] ( Row=5, iCol=1, fCType=SQL_C_CHAR,
rgbValue="TABLC      ", pcbValue=10 )
[000207A9 0000000000000021] [2008-09-24 16:00:25.318065] ( Row=6, iCol=1, fCType=SQL_C_CHAR,
rgbValue="TABLB      ", pcbValue=10 )
[000207A9 0000000000000021] [2008-09-24 16:00:25.318087] ( Row=7, iCol=1, fCType=SQL_C_CHAR,
rgbValue="TABLA      ", pcbValue=10 )
[000207A9 0000000000000021] [2008-09-24 16:00:25.318109] SQLExtendedFetch( pcRow=7 )
```

Related tasks:

Chapter 4, “Troubleshooting and support,” on page 647

If you are having problems with your message flow applications, use the techniques described in this section to help you to diagnose and solve the problems.

“Using trace” on page 845

You can use different types of trace to help you with problem determination and troubleshooting.

Related reference:



Supported databases

You can optionally configure databases to contain data that is accessed by your message flows. Databases from IBM and other suppliers are supported at specific versions on supported operating systems.



Database facilities

The database products used by IBM Integration Bus also record information that might be useful if you have any problems with their access.

IBM Integration API (CMP) trace

Enable or disable service trace for the IBM Integration API (also known as the CMP).

Enabling CMP trace

About this task

To enable tracing for the CMP for your application, set the environment variable MQSI_CMP_TRACE, where *<filename>* is the name of the file to which the trace is sent:

```
export MQSI_CMP_TRACE=<filename>
```

Or you can enable tracing by using the following API call in your code:

```
// Enable Administration service trace
BrokerProxy.enableAdministrationAPITracing("outputfile.txt");
```

This request logs all calls to the CMP to the outputfile.txt file in the current directory. All CMP activity in the entire Java virtual machine is logged.

You can also enable CMP service trace from the **File** menu of the CMP Exerciser.

In addition, because the CMP uses the WebSphere MQ Java client, you can enable WebSphere MQ Java client tracing.

Disabling CMP trace

About this task

To disable tracing for the CMP for your application, use the following API call in your code:

```
// Disable Administration service trace
BrokerProxy.disableAdministrationAPITracing();
```

You can also disable CMP service trace from the **File** menu of the CMP Exerciser.

Related concepts:



Trace

If you cannot get enough information about a particular problem from the entries that are available in the various logs, the next troubleshooting method to consider is using trace. Trace provides more details about what is happening while code runs. The information produced from trace is sent to a specified trace record, so that you or IBM support personnel can analyze it to discover the cause of your problem.



The IBM Integration API

The IBM Integration API is a programming interface that your applications can use to control brokers and their resources through a remote interface.

Related tasks:

“Using trace” on page 845

You can use different types of trace to help you with problem determination and troubleshooting.

“Resolving problems when developing CMP applications” on page 828

Use the advice given here to help you to resolve problems that can arise when developing IBM Integration API (also known as the CMP) applications.

Related reference:



Service trace

Service trace is one of two types of optional trace that are available in IBM Integration Bus and provides more information than that provided by the entries that are written to the Event Log or to user trace. Service trace is inactive by default; you must activate it explicitly by using a command.

Switching Trace nodes on and off

Use the `mqsi changetrace` command or the IBM Integration Explorer to switch Trace nodes on and off.

About this task

When an integration server is created, or a message flow is deployed, its Trace node switch is set to on by default. Integration servers and message flows that you have migrated from a previous version are also handled in this way. The message flow level Trace node switch setting is not changed on redeployment. You can significantly improve the performance of a flow that includes Trace nodes by switching Trace nodes off.

Restriction: If you switch off Trace nodes in a flow that was migrated from a version earlier than 6.1, you can no longer revert that flow to its earlier state.

If the Trace node setting for an integration server is off, all Trace nodes in its flows are disabled. You can change the settings for Trace nodes in individual message flows; the settings are applied when you turn on Trace nodes for the integration server. If the Trace node setting for an integration server is on, the Trace node switch setting of each message flow determines the effective settings. Use the `mqsireporttrace` command to check the settings for message flows and integration servers.

Example: switching off Trace nodes for an integration server on distributed systems, using the command line Procedure

To disable the Trace node switch settings of all message flows in an integration server of a broker called IB9NODE, (that is, stop all the Trace nodes in any of the message flows deployed to the integration server from executing), enter the following command:

```
mqsichangetrace IB9NODE -n off -e default
```

In this example, `-n off` switches off Trace nodes in the default integration server (`-e default`).

All Trace nodes are switched off, even if the message flow that contains them has its Trace node switch set to *on*.

Example: switching off Trace nodes for an integration server on z/OS systems, using the command line Procedure

To disable the Trace node switch settings of all message flows in an integration server of a broker called MQP1BRK, (that is, stop all the Trace nodes in any of the message flows deployed to the integration server from executing), enter the following command:

```
F MQP1BRK,ct n='off', e='default'
```

In this example, `n='off'` switches off Trace nodes in the default integration server (`e='default'`).

All Trace nodes are switched off, even if the message flow that contains them has its Trace node switch set to *on*.

Example: switching off Trace nodes for a message flow on distributed systems, using the command line About this task

Trace nodes for the default integration server in broker IB9NODE are switched on. You want to turn off Trace nodes for the myFlow message flow.

Procedure

Enter the following command:

```
mqsichangetrace IB9NODE -n off -e default -f myFlow
```

Example: switching on Trace nodes for a message flow and an integration server on distributed systems, using the command line

About this task

Trace nodes for the default integration server in broker IB9NODE are switched off. You want to turn on Trace nodes for the myFlow message flow, then turn on Trace nodes for the integration server.

Procedure

1. To turn on Trace nodes for the message flow, enter the following command:

```
mqsichangetrace IB9NODE -n on -e default -f myFlow
```

Trace nodes for the myFlow message flow are turned on, but this setting is not applied until you turn on trace nodes for the integration server.

2. To turn on Trace nodes for the integration server, enter the following command:

```
mqsichangetrace IB9NODE -n on -e default
```

Trace nodes are turned on for the default integration server. Within that integration server, Trace nodes are enabled in all message flows that have Trace nodes turned on, including the myFlow message flow.

Example: switching off Trace nodes for a message flow on z/OS systems, using the command line

About this task

Trace nodes for the default integration server in broker MQP1BRK are switched on. You want to turn off Trace nodes for one of the message flows (myFlow).

Procedure

Enter the following command:

```
F MQP1BRK,ct n='off', e='default', f='myFlow'
```

Example: switching on Trace nodes for an integration server, using the IBM Integration Explorer

About this task

Trace nodes are enabled by default. If you have switched them off, follow these steps to enable Trace nodes for an integration server from the IBM Integration Explorer:

Procedure

1. In the Navigator view, expand the Integration Nodes folder and right-click the integration server with which you want to work.
2. Click **Trace Nodes All Flows > Enable**. An alert saying Trace nodes are switched on is displayed in the Alert Viewer. The Trace node switch setting of each message flow determines the effective settings.

Example: switching on Trace nodes for a message flow, using the IBM Integration Explorer

Before you begin

If the Trace node setting for an integration server is off, all Trace nodes in its flows are disabled. You can change the settings for Trace nodes in individual message flows; the settings are applied when you turn on Trace nodes for the integration server. If the Trace node setting for an integration server is on, the Trace node switch setting of each message flow determines the effective settings.

About this task

Follow these steps to enable Trace nodes for one of your message flows from the IBM Integration Explorer:

Procedure

1. In the Navigator view, expand the Integration Nodes folder and right-click the message flow with which you want to work.
2. Click **Trace Nodes > Enable**. An alert is displayed in the Alert Viewer.

Related concepts:



IBM Integration Explorer

The IBM Integration Explorer is a graphical user interface based on the Eclipse platform for administering your integration nodes (brokers).

Related reference:



mqsichangetrace command

Use the **mqsichangetrace** command to set the tracing characteristics for a broker.



mqsireporttrace command

Use the **mqsireporttrace** command to display the trace options currently in effect. Trace can be run only against a broker, or any of its resources.



Trace node

Use the Trace node to generate trace records that you can use to monitor the behavior of a message flow.

Using dumps and abend files

A dump or an abend file might be produced when a problem occurs. Dumps and abend files can be used by IBM to resolve the problem.

About this task

This section contains the following topics:

Procedure

- “Checking for dumps” on page 872
- “Using the DUMP command on z/OS” on page 873
- “Checking for abend files” on page 875

What to do next

You can also use the “IBM Support Assistant Data Collector” on page 879 to help with data collection.

Related tasks:

Chapter 4, “Troubleshooting and support,” on page 647

If you are having problems with your message flow applications, use the techniques described in this section to help you to diagnose and solve the problems.

“Contacting your IBM Support Center” on page 876

If you cannot resolve problems that you find when you use IBM Integration Bus, or if you are directed to do so by an error message generated by IBM Integration Bus, you can request assistance from your IBM Support Center.

“IBM Support Assistant Data Collector” on page 879

You can collect diagnostic documents by using IBM Support Assistant Data Collector, and submit a problem report to IBM. IBM Support Assistant Data Collector is included with your IBM Integration Bus installation.

Related reference:



Abend files

When a process does not end normally an abend file is generated.



Dumps

Under exceptional circumstances, Windows MiniDumps, UNIX core dumps, or z/OS SVC or core dumps might be produced.

Checking for dumps

If a dump occurs on your system, an error message is produced.

Procedure

• On Windows

BIP2111 error message (message broker internal error). The error message contains the path to the MiniDump file in your errors directory.

• On UNIX

BIP2060 error message (integration server terminated unexpectedly). Look in the directory where the broker was started, or in the service user ID's home directory, to find the core dump file.

• On z/OS

– BIP2060 error message (integration server ended unexpectedly) from the main Broker Address Space. This message should be accompanied by one of the following messages and dump.

– IEF450I message in the syslog, or component's joblog, showing an abend code followed by a reason code, for example:

```
IEF450I MQ83BRK DEFAULT - ABEND=S2C1 U0000 REASON=000000C4
```

Look in the system's dump dataset hlq for the dump dataset, or search the syslog for the appropriate IEA611I message to find out the dump dataset name.

– IEA993I message in the syslog for a SYSMDUMP. Look in the started task user's directory for the coredump.pid file, as specified in the syslog:

```
IEA993I SYSMDUMP TAKEN TO coredump.00500319
```

- An error message for an SVC dump; see Dumps on IBM Integration Bus for z/OS for further information on SVC dumps.

Related tasks:

Chapter 4, “Troubleshooting and support,” on page 647

If you are having problems with your message flow applications, use the techniques described in this section to help you to diagnose and solve the problems.

“Using dumps and abend files” on page 871

A dump or an abend file might be produced when a problem occurs. Dumps and abend files can be used by IBM to resolve the problem.

“Checking for abend files” on page 875

Abend files are produced when a process ends abnormally. The information contained in an abend file helps the IBM Support Center to diagnose and fix the problem.

“Using the DUMP command on z/OS”

Follow the steps in this task to use the DUMP command on z/OS.

Related reference:



Dumps

Under exceptional circumstances, Windows MiniDumps, UNIX core dumps, or z/OS SVC or core dumps might be produced.



Abend files

When a process does not end normally an abend file is generated.

Using the DUMP command on z/OS

Follow the steps in this task to use the DUMP command on z/OS.

About this task

z/OS You might be asked to dump any or several of the following address spaces for IBM to resolve the problem:

- Control address space
- DataFlowEngine address space
- OMVS kernel address space

The following procedure demonstrates how to dump the DataFlowEngine address space. This procedure is the same for any of the address spaces.

Procedure

1. Find the address space ID of the address space that you want to dump using the display command on the z/OS syslog:

```
D OMVS,U=your started task user ID
```

This command displays the address spaces of all the processes that are running from your started task user ID, for example:

```
D OMVS,U=MQ01BRK
BPX0040I 16.14.30 DISPLAY OMVS 237
OMVS      000D ACTIVE          OMVS=(14)
USER      JOBNAME  ASID        PID          PPID STATE   START      CT_SECS
MQ01BRK  MQ01BRK  009D    67306064    84083282 HRI--- 15.41.55   48.37
LATCHWAITPID=          0 CMD=bipservice MQ01BRK AUTO
```

```

MQ01BRK MQ01BRK 009D 84083282          1 1WI--- 15.41.55    48.37
  LATCHWAITPID= 0 CMD=/argoinst/S000_L30307_P/usr/lpp/mqsi/bin
MQ01BRK MQ01BRK 009D 16974444 67306064 HRI--- 15.42.01    48.37
  LATCHWAITPID= 0 CMD=bipbroker MQ01BRK
MQ01BRK MQ01BRK 009F 16974445          1 1W---- 15.42.05   2914.22
  LATCHWAITPID= 0 CMD=/argoinst/S000_L30307_P/usr/lpp/mqsi/bin
MQ01BRK MQ01BRK 009F 33751662 16974445 HR---- 15.42.05   2914.22
  LATCHWAITPID= 0 CMD=DataFlowEngine MQ01BRK ca614eec-f300-000

```

The infrastructure main program bipimain is the first process in every address space. For a control address space, bipimain starts the bipservice process, which starts the bipbroker process, which might also start the biphttplistener process, depending on the configuration. For a DataFlowEngine address space, bipimain starts the DataFlowEngine process. For each integration server, an additional DataFlowEngine address space is started. In this example, only one integration server is deployed.

2. Use the z/OS DUMP command to dump the DataFlowEngine address space, which is shown in the above example as 9F.

- a. Enter the following command:

```
DUMP TITLE=(DFE)
```

The console returns:

```
*^15 IEE094D SPECIFY OPERAND(S) FOR DUMP COMMAND
```

- b. Enter:

```
R 15,ASID=9F,CONT
```

The console returns:

```
*16 IEE094D SPECIFY OPERAND(S) FOR DUMP COMMAND
```

- c. Enter:

```
R 16,SDATA=(CSA,RGN,PSA,SQA,LSQA,LPA,TRT,GRSQ,SUM),END
```

The console returns:

```

IEE600I REPLY TO 16 IS;SDATA=(CSA,RGN,PSA,SQA,LSQA,LPA,TRT,GRSQ,SUM),END
IEA794I SVC DUMP HAS CAPTURED: 356
DUMPID=014 REQUESTED BY JOB (*MASTER*)
DUMP TITLE=DFE
IEF196I IGD101I SMS ALLOCATED TO DDNAME (SYS00018)
IEF196I          DSN (SYS3.DUMP.ARG5.#MASTER#.T142958.S00014      )
IEF196I          STORCLAS (SMS) MGMTCLAS (DUMP) DATACLAS (      )
IEF196I          VOL SER NOS= ARGSMR

```

The dump is stored in either a pre-allocated dump data set called SYS1.DUMPxx, or an automatically allocated dump data set named according to an installation-specified pattern.

3. In some scenarios, all address spaces for a given broker, that is, all those listed in the example above, plus the OMVS address space and dataspace, are needed by IBM to resolve the problem. Use the z/OS DUMP command to dump all MQ01BRK address spaces.

- a. Enter the command:

```
DUMP TITLE=(ALL)
```

The console returns:

```
^15 IEE094D SPECIFY OPERAND(S) FOR DUMP COMMAND
```

- b. Enter:


```
R 15,JOBNAME=(OMVS,MQ01BRK),DSPNAME=('OMVS'.*),SDATA=(PSA,SQA,LSQA,RGN,TRT,
LPA,CSA,GRSQ,SUM,NUC)
```

The console returns:

```
IEE600I REPLY TO 15 IS;JOBNAME=(OMVS,MQ01BRK),DSPNAME=('OMVS'.*),S
IEA794I SVC DUMP HAS CAPTURED: 303
DUMPID=040 REQUESTED BY JOB (*MASTER*)
DUMP TITLE=ALL
IEE853I 13.40.40 SYS1.DUMP TITLES 306
SYS1.DUMP DATA SETS AVAILABLE=000 AND FULL=000
CAPTURED DUMPS=0001, SPACE USED=00000447M, SPACE FREE=00001053M
DUMP.MVK4.#MASTER#.D030415.T134007.S00039 DATA UNAVAILABLE WHILE
BEING DUMPED TO
IEA611I COMPLETE DUMP ON DUMP.MVK4.#MASTER#.D030415.T134007.S00039 309
DUMPID=040 REQUESTED BY JOB (*MASTER*)
FOR ASIDS(000D,009D,009F)
```

Results

You can also find information on the individual thread by issuing the **DISPLAY** z/OS console command, as in the example:

```
D OMVS,PID=83886535
```

Related tasks:

Chapter 4, “Troubleshooting and support,” on page 647

If you are having problems with your message flow applications, use the techniques described in this section to help you to diagnose and solve the problems.

“Using dumps and abend files” on page 871

A dump or an abend file might be produced when a problem occurs. Dumps and abend files can be used by IBM to resolve the problem.

“Checking for dumps” on page 872

If a dump occurs on your system, an error message is produced.

“Contacting your IBM Support Center” on page 876

If you cannot resolve problems that you find when you use IBM Integration Bus, or if you are directed to do so by an error message generated by IBM Integration Bus, you can request assistance from your IBM Support Center.

Related reference:



Dumps

Under exceptional circumstances, Windows MiniDumps, UNIX core dumps, or z/OS SVC or core dumps might be produced.



Abend files

When a process does not end normally an abend file is generated.

Checking for abend files

Abend files are produced when a process ends abnormally. The information contained in an abend file helps the IBM Support Center to diagnose and fix the problem.

About this task

The following list contains examples of what might cause the broker to produce an abend file:

Procedure

- The broker runs out of memory.
- A user-defined extension causes an instruction in the broker process that is not valid.
- An unrecoverable error occurs in the broker.

Results

Abend files are never produced during normal operation. If an abend file is produced, contact the IBM Support Center for assistance.

Related tasks:

“Contacting your IBM Support Center”

If you cannot resolve problems that you find when you use IBM Integration Bus, or if you are directed to do so by an error message generated by IBM Integration Bus, you can request assistance from your IBM Support Center.

Chapter 4, “Troubleshooting and support,” on page 647

If you are having problems with your message flow applications, use the techniques described in this section to help you to diagnose and solve the problems.

“Checking for dumps” on page 872

If a dump occurs on your system, an error message is produced.

“Using dumps and abend files” on page 871

A dump or an abend file might be produced when a problem occurs. Dumps and abend files can be used by IBM to resolve the problem.

“Using the DUMP command on z/OS” on page 873

Follow the steps in this task to use the DUMP command on z/OS.

Related reference:



Dumps

Under exceptional circumstances, Windows MiniDumps, UNIX core dumps, or z/OS SVC or core dumps might be produced.



Abend files

When a process does not end normally an abend file is generated.

Contacting your IBM Support Center

If you cannot resolve problems that you find when you use IBM Integration Bus, or if you are directed to do so by an error message generated by IBM Integration Bus, you can request assistance from your IBM Support Center.

About this task

Before you contact your Support Center, use the checklist shown here to gather important information. Some items might not be relevant in every situation, but provide as much information as you can to enable the IBM Support Center to re-create your problem. You can also use the “IBM Support Assistant Data Collector” on page 879 to help with data collection.

For IBM Integration Bus:

- The product version.
- Any fix packs applied.
- Any interim fixes applied.

- All current trace and error logs, including relevant Windows Event log or Linux/UNIX operating system syslog entries, and any abend or dump files from the *install_dir*\errors directory on Windows, or the */var/mqsi/errors* directory on Linux/UNIX. Obtain user trace log files at debug level for all relevant message flows and preferably format them. Also include any requested service trace files.

To send files from distributed systems, create a compressed file using any compression utility.

To send a file from the file system to IBM, use **tar** to compress the file. For example `tar -cx -f coredump.0002009E coredump.toibm`. To send MVS data sets to IBM, terse them using TRSMAIN, which you can download from z/OS tools download.

- A list of the components installed. Include details of the number of computers and their operating systems, the number of brokers and the computers on which they are running, and the existence and details of any User Name Servers.
- The compressed file obtained by exporting your workspace and appropriate message flows and message sets. This action is performed from the IBM Integration Toolkit.
- Details of the operation that you were performing, the results that occurred, and the results that you were expecting.
- A sample of the messages that were being used when the problem arose
- If relevant, the report file from the C or COBOL importer. This file is located in the directory from which the file import was attempted.
- If you are using tagged delimited wire format on message sets, the TDS log files.

For WebSphere MQ:

- The product version.
- Any fix packs applied.
- Any interim fixes applied.
- All current trace and error logs, including relevant Windows Event log or Linux and UNIX operating system syslog entries and First Failure Support Technology™ (FFST™) output files. You can find these files, which have the extension *.fdc*, in the errors subdirectory in the WebSphere MQ home directory.
- Details of WebSphere MQ client software, if appropriate.

For each database that you are using:

- The product and release level (for example, DB2 9.1).
- Any fix packs applied.
- Any interim fixes applied.
- All current trace and error logs, including relevant Windows Event log or Linux and UNIX operating system syslog entries, for example the *db2diag.log* file on DB2. Check the database product documentation for details of where to find these files.
- Definitions of any database tables.
- Any ODBC traces.

Windows

For Windows:

- The version.
- The Service Pack level.

- The environment settings.

UNIX

For Linux and UNIX operating systems:

- The product version. You can find the version installed by using the `uname -a` command.
- Any service level and patches that have been applied.
- The environment settings.

z/OS

For z/OS:

- The product version
- The list of PTFs that have been applied
- The environment settings
- The job logs from all address spaces

Related tasks:

Chapter 4, “Troubleshooting and support,” on page 647

If you are having problems with your message flow applications, use the techniques described in this section to help you to diagnose and solve the problems.

“Windows: Viewing the local error log” on page 840

The Windows Event Viewer is where IBM Integration Bus writes records to the local system. Use Windows system facilities to view this log.

“Linux and UNIX systems: Configuring the syslog daemon” on page 841

On Linux and UNIX systems, all IBM Integration Bus messages (other than messages that are generated by the command-line utilities) are sent to the syslog subsystem.

“IBM Support Assistant Data Collector” on page 879

You can collect diagnostic documents by using IBM Support Assistant Data Collector, and submit a problem report to IBM. IBM Support Assistant Data Collector is included with your IBM Integration Bus installation.

Related reference:



User trace

User trace is one of two types of optional trace that are available in IBM Integration Bus and provides more information than that provided by the entries that are written to the Administration log. User trace is inactive by default; you must activate it explicitly by using a command, or by selecting options in the IBM Integration Toolkit.



Service trace

Service trace is one of two types of optional trace that are available in IBM Integration Bus and provides more information than that provided by the entries that are written to the Event Log or to user trace. Service trace is inactive by default; you must activate it explicitly by using a command.



`mqsichangetrace` command

Use the `mqsichangetrace` command to set the tracing characteristics for a broker.



`mqsiformatlog` command

Use the `mqsiformatlog` command to process the XML log created by `mqsireadlog`. The command retrieves and formats any messages that the XML log contains into a form suitable for the locale of the user who runs the command.



`mqsireadlog` command

Use the `mqsireadlog` command to retrieve trace records for the specified

component.



mqsiporttrace command

Use the **mqsiporttrace** command to display the trace options currently in effect. Trace can be run only against a broker, or any of its resources.

IBM Support Assistant Data Collector

You can collect diagnostic documents by using IBM Support Assistant Data Collector, and submit a problem report to IBM. IBM Support Assistant Data Collector is included with your IBM Integration Bus installation.

About this task

You might have other versions of IBM Support Assistant, but if you run the commands provided in the following topics, you will collect the appropriate information for IBM Integration Bus.

Procedure

- “Collecting data in console mode with IBM Support Assistant Data Collector” on page 880
- “Selecting a problem collector for IBM Support Assistant Data Collector” on page 881

What to do next

Note: The IBM Support Assistant Data Collector is not supported for z/OS.

Related concepts:

“Selecting a problem collector for IBM Support Assistant Data Collector” on page 881

You can use the problem collectors installed with IBM Support Assistant Data Collector to gather diagnostic information.

Related tasks:

“Collecting data in console mode with IBM Support Assistant Data Collector” on page 880

You can use the IBM Support Assistant Data Collector in console mode to collect diagnostic documents for submission to IBM.

“Contacting your IBM Support Center” on page 876

If you cannot resolve problems that you find when you use IBM Integration Bus, or if you are directed to do so by an error message generated by IBM Integration Bus, you can request assistance from your IBM Support Center.

“Making initial checks” on page 650

Before you start problem determination in detail, consider whether there is an obvious cause of the problem, or an area of investigation that is likely to give useful results. This approach to diagnosis can often save a lot of work by highlighting a simple error, or by narrowing down the range of possibilities.

Chapter 4, “Troubleshooting and support,” on page 647

If you are having problems with your message flow applications, use the techniques described in this section to help you to diagnose and solve the problems.

Collecting data in console mode with IBM Support Assistant Data Collector

You can use the IBM Support Assistant Data Collector in console mode to collect diagnostic documents for submission to IBM.

Before you begin

Before you start:

Before contacting IBM Software Support, ensure that your company has an active IBM software subscription and support contract, and that you are authorized to submit problems to IBM. See “Contacting IBM Software Support” on page 885 for more details.

About this task

Procedure

To run the IBM Support Assistant Data Collector and collect diagnostic documents, complete the following steps:

1. Ensure that the IBM Integration Bus run time variables are set correctly for your environment by using the **mqsiprofile** command. See Environment variables after installation for more information.
2. At a command prompt, enter the **mqsidc** command.

Note: You can ensure that the script is executable by entering the following command to change the file permissions: **chmod 755 mqsidc**

The IBM Support Assistant Data Collector starts in console mode.

3. Start the data collection. Available options are presented as numbered lists.
 - a. Type a file name for saving the collected data, or press Enter to generate a unique file name.
 - b. At the input field prompts, type the number of the required option and press Enter.
4. Choose a data transfer method.
 - a. Type the number of the required option for transferring the diagnostic documents to IBM and press Enter. The available options are:
 - 1) Send the documents to IBM Software Support using secure transfer (HTTPS). You require a problem management record (PMR) number obtained through IBM Software Support.
 - 2) FTP the documents to IBM Software Support (unencrypted). You require a PMR number obtained through IBM Software Support. This option is less secure than option i.
 - 3) FTP the documents to another location (unencrypted). You are required to provide a target FTP location and appropriate authentication to access the documents.
 - 4) End the collection without sending.If you choose any of the first three options above, you are required to enter additional information to complete the upload.
 - b. Type the number of the required option to confirm the data collection.

When the IBM Support Assistant Data Collector completes, a .zip file is created at the location specified during the collection. You can extract the compressed files and examine the collected data by using a suitable tool.

What to do next

Note: The IBM Support Assistant Data Collector is not supported for z/OS.

Related concepts:

“Selecting a problem collector for IBM Support Assistant Data Collector”

You can use the problem collectors installed with IBM Support Assistant Data Collector to gather diagnostic information.

Related tasks:

“IBM Support Assistant Data Collector” on page 879

You can collect diagnostic documents by using IBM Support Assistant Data Collector, and submit a problem report to IBM. IBM Support Assistant Data Collector is included with your IBM Integration Bus installation.

“Contacting your IBM Support Center” on page 876

If you cannot resolve problems that you find when you use IBM Integration Bus, or if you are directed to do so by an error message generated by IBM Integration Bus, you can request assistance from your IBM Support Center.

“Contacting IBM Software Support” on page 885

IBM Software Support provides assistance with product defects.

“Making initial checks” on page 650

Before you start problem determination in detail, consider whether there is an obvious cause of the problem, or an area of investigation that is likely to give useful results. This approach to diagnosis can often save a lot of work by highlighting a simple error, or by narrowing down the range of possibilities.

Chapter 4, “Troubleshooting and support,” on page 647

If you are having problems with your message flow applications, use the techniques described in this section to help you to diagnose and solve the problems.

Selecting a problem collector for IBM Support Assistant Data Collector

You can use the problem collectors installed with IBM Support Assistant Data Collector to gather diagnostic information.

Note: Before you can use the problem collectors, IBM Integration Bus must have installed successfully, and an **mqsiprofile** must exist which you can start successfully (see Environment variables after installation for more information).

Generic problem collector

The generic problem collector gathers configuration information about IBM Integration Bus, operating system details, WebSphere MQ levels, environment details, the application event log, and the syslog.

Broker problem collector

The broker problem collector gathers information about the broker's deployed configuration, the standard system logs (STDOUT and STDERR) for its components (admin agent, ExecutionGroup, and httpListener), and abend files. This problem collector also gathers the diagnostic data collected by the generic problem collector.

Related tasks:

“IBM Support Assistant Data Collector” on page 879

You can collect diagnostic documents by using IBM Support Assistant Data Collector, and submit a problem report to IBM. IBM Support Assistant Data Collector is included with your IBM Integration Bus installation.

“Collecting data in console mode with IBM Support Assistant Data Collector” on page 880

You can use the IBM Support Assistant Data Collector in console mode to collect diagnostic documents for submission to IBM.

“Contacting your IBM Support Center” on page 876

If you cannot resolve problems that you find when you use IBM Integration Bus, or if you are directed to do so by an error message generated by IBM Integration Bus, you can request assistance from your IBM Support Center.

“Making initial checks” on page 650

Before you start problem determination in detail, consider whether there is an obvious cause of the problem, or an area of investigation that is likely to give useful results. This approach to diagnosis can often save a lot of work by highlighting a simple error, or by narrowing down the range of possibilities.

Chapter 4, “Troubleshooting and support,” on page 647

If you are having problems with your message flow applications, use the techniques described in this section to help you to diagnose and solve the problems.

Searching knowledge bases

If you have a problem with your IBM software, you want it resolved quickly. Begin by searching the available knowledge bases to determine whether the resolution to your problem is already documented.

Procedure

1. Search the information center

IBM provides extensive documentation in the form of online information centers. An information center can be installed on your local machine or on a local intranet. An information center can also be viewed on the IBM Web site. You can use the powerful search function of the information center to query conceptual and reference information as well as detailed instructions for completing tasks.

2. Search the Internet

If you cannot find an answer to your question in the information center, search the Internet for the latest, most complete information that might help you resolve your problem, including:

- IBM technotes
- IBM downloads
- IBM Redbooks® publications
- IBM developerWorks
- Forums and newsgroups
- Internet search engines

Related concepts:

“Troubleshooting overview” on page 647

Troubleshooting is the process of finding and eliminating the cause of a problem. Whenever you have a problem with your IBM software, the troubleshooting process begins as soon as you ask yourself “what happened?”

Related tasks:

Chapter 4, “Troubleshooting and support,” on page 647

If you are having problems with your message flow applications, use the techniques described in this section to help you to diagnose and solve the problems.

“Making initial checks” on page 650

Before you start problem determination in detail, consider whether there is an obvious cause of the problem, or an area of investigation that is likely to give useful results. This approach to diagnosis can often save a lot of work by highlighting a simple error, or by narrowing down the range of possibilities.

“Dealing with problems” on page 666

Learn how to resolve some of the typical problems that can occur.

“Using logs” on page 839

There are a variety of logs that you can use to help with problem determination and troubleshooting.

“Using trace” on page 845

You can use different types of trace to help you with problem determination and troubleshooting.

“Using dumps and abend files” on page 871

A dump or an abend file might be produced when a problem occurs. Dumps and abend files can be used by IBM to resolve the problem.

“Getting product fixes”

A product fix might be available to resolve your problem. You can determine what fixes are available from the IBM support site.

“Contacting IBM Software Support” on page 885

IBM Software Support provides assistance with product defects.

“Recovering after failure” on page 887

Follow a set of procedures to recover after a serious problem.

Related reference:



Troubleshooting

Use the reference information in this section to help you diagnose errors in IBM Integration Bus.

Getting product fixes

A product fix might be available to resolve your problem. You can determine what fixes are available from the IBM support site.

About this task

To determine what fixes are available from the IBM support site:

1. Open the IBM Integration Bus support web page.
2. Click **Downloads**, then **Recommended fixes**. This web page provides links to the latest available maintenance for the in service IBM Integration Bus family of products.

To receive weekly email notifications about fixes and other news about IBM products, follow these steps.

Procedure

1. From the support site (IBM Integration Bus support web page), locate the **Notifications** box in the center of the page.

2. If you are not signed in, click **Sign in to create, manage or view your subscriptions**. If you have not registered, click **register now** on the sign-in page and follow the on-screen instructions.
3. Click **Manage my subscriptions**.
4. Click the **Subscribe** tab. A list of products families is shown.
5. In the Software column, click **WebSphere**. A list of products is shown.
6. Select the product for which you want to receive notifications (for example, **WebSphere Message Broker**), then click **Continue**.
7. Set options to determine what notifications you receive, how often you receive them, and to which folder they are saved, then click **Submit**.

Related concepts:

“Troubleshooting overview” on page 647

Troubleshooting is the process of finding and eliminating the cause of a problem. Whenever you have a problem with your IBM software, the troubleshooting process begins as soon as you ask yourself “what happened?”

Related tasks:

Chapter 4, “Troubleshooting and support,” on page 647

If you are having problems with your message flow applications, use the techniques described in this section to help you to diagnose and solve the problems.

“Making initial checks” on page 650

Before you start problem determination in detail, consider whether there is an obvious cause of the problem, or an area of investigation that is likely to give useful results. This approach to diagnosis can often save a lot of work by highlighting a simple error, or by narrowing down the range of possibilities.

“Dealing with problems” on page 666

Learn how to resolve some of the typical problems that can occur.

“Using logs” on page 839

There are a variety of logs that you can use to help with problem determination and troubleshooting.

“Using trace” on page 845

You can use different types of trace to help you with problem determination and troubleshooting.

“Using dumps and abend files” on page 871

A dump or an abend file might be produced when a problem occurs. Dumps and abend files can be used by IBM to resolve the problem.

“Getting product fixes” on page 883

A product fix might be available to resolve your problem. You can determine what fixes are available from the IBM support site.

“Contacting IBM Software Support” on page 885

IBM Software Support provides assistance with product defects.

“Recovering after failure” on page 887

Follow a set of procedures to recover after a serious problem.

Related reference:



Troubleshooting

Use the reference information in this section to help you diagnose errors in IBM Integration Bus.

Contacting IBM Software Support

IBM Software Support provides assistance with product defects.

About this task

Before you contact IBM Software Support, you must ensure that your company has an active IBM software subscription and support contract, and that you are authorized to submit problems to IBM. The type of software subscription and support contract that you need depends on the type of product that you have:

- For IBM distributed software products (including, but not limited to, Tivoli, Lotus®, and Rational products, as well as DB2 and WebSphere products that run on Windows or UNIX operating systems), enroll in Passport Advantage® in one of the following ways:
 - Online: Go to the Passport Advantage web page and click **How to Enroll**.
 - By telephone: For the telephone number to call in your country, go to the Software Support Handbook, click **Contacts**, then **Worldwide contacts**.
- For customers with Subscription and Support (S & S) contracts, go to the Open service request web page.
- For customers with IBMLink, CATIA, Linux, S/390®, iSeries, pSeries, zSeries, and other support agreements, go to the IBM Support Line web page.
- For IBM eServer™ software products (including, but not limited to, DB2 and WebSphere products that run in zSeries, pSeries, and iSeries environments), you can purchase a software subscription and support agreement by working directly with an IBM marketing representative or an IBM Business Partner. For more information about support for eServer software products, go to the Support for IBM Systems web page.

If you are not sure what type of software subscription and support contract you need, call 1-800-IBMSERV (1-800-426-7378) in the United States or, from other countries, go to the contacts page of the Software Support Handbook and click the name of your geographic region for telephone numbers of people who provide support for your location.

Follow the steps in this topic to contact IBM Software Support:

Procedure

1. “Determine the effect of the problem on your business”
2. “Describe your problem and gather background information” on page 886
3. “Submit your problem to IBM Software Support” on page 886

Determine the effect of the problem on your business

About this task

When you report a problem to IBM, you will be asked to supply a severity level. Therefore, you need to understand and assess the effect on your business of the problem that you are reporting. Use the following criteria:

Severity 1

Critical effect on business: You are unable to use the program, resulting in a critical effect on operations. This condition requires an immediate solution.

Severity 2

Significant effect on business: The program is usable but is severely limited.

Severity 3

Some effect on business: The program is usable with less significant features (not critical to operations) unavailable.

Severity 4

Minimal effect on business: The problem has little effect on operations, or a reasonable workaround to the problem has been implemented.

Describe your problem and gather background information

About this task

When you are explaining a problem to IBM, be as specific as possible. Include all relevant background information so that IBM Software Support specialists can help you to solve the problem efficiently. To save time, know the answers to these questions:

- What software versions were you running when the problem occurred?
- Do you have logs, traces, and messages that are related to the problem symptoms? IBM Software Support is likely to ask for this information.
- Can the problem be re-created? If so, what steps led to the failure?
- Have any changes been made to the system? (For example, hardware, operating system, networking software, and so on.)
- Are you currently using a workaround for this problem? If so, be prepared to explain it when you report the problem.

Submit your problem to IBM Software Support

About this task

You can submit your problem in one of two ways:

- **Online:** Go to the Software Support Handbook and enter your information into the appropriate problem submission tool.
- **By telephone:** For the telephone number to call in your country, go to the contacts page of the Software Support Handbook and click the name of your geographic region for telephone numbers of people who provide support for your location.

The IBM Support Assistant Data Collector can be used to submit diagnostic documents if you have an existing problem management record (PMR). For more information, see “IBM Support Assistant Data Collector” on page 879.

If the problem that you submit is for a software defect or for missing or inaccurate documentation, IBM Software Support might create an Authorized Program Analysis Report (APAR). The APAR describes the problem in detail. Whenever possible, IBM Software Support provides a workaround for you to implement until the APAR is resolved and a fix is delivered.

IBM publishes resolved APARs on the IBM product support Web pages daily, so that other users who experience the same problem can benefit from the same resolutions.

Related concepts:

“Troubleshooting overview” on page 647

Troubleshooting is the process of finding and eliminating the cause of a problem. Whenever you have a problem with your IBM software, the troubleshooting process begins as soon as you ask yourself “what happened?”

Related tasks:

Chapter 4, “Troubleshooting and support,” on page 647

If you are having problems with your message flow applications, use the techniques described in this section to help you to diagnose and solve the problems.

“Making initial checks” on page 650

Before you start problem determination in detail, consider whether there is an obvious cause of the problem, or an area of investigation that is likely to give useful results. This approach to diagnosis can often save a lot of work by highlighting a simple error, or by narrowing down the range of possibilities.

“Dealing with problems” on page 666

Learn how to resolve some of the typical problems that can occur.

“Using logs” on page 839

There are a variety of logs that you can use to help with problem determination and troubleshooting.

“Using trace” on page 845

You can use different types of trace to help you with problem determination and troubleshooting.

“Using dumps and abend files” on page 871

A dump or an abend file might be produced when a problem occurs. Dumps and abend files can be used by IBM to resolve the problem.

“Getting product fixes” on page 883

A product fix might be available to resolve your problem. You can determine what fixes are available from the IBM support site.

“Contacting IBM Software Support” on page 885

IBM Software Support provides assistance with product defects.

“Recovering after failure”

Follow a set of procedures to recover after a serious problem.

Related reference:



Troubleshooting

Use the reference information in this section to help you diagnose errors in IBM Integration Bus.

Recovering after failure

Follow a set of procedures to recover after a serious problem.

About this task

Use the recovery methods described here only if you cannot resolve the underlying problem by using the diagnostic techniques described throughout the Chapter 4, “Troubleshooting and support,” on page 647 section of the information center. If your problem cannot be resolved by using these techniques, contact your IBM Support Center.

This section contains the following topics:

Procedure

- “Recovering after the broker fails”
- “Recovering after an integration server fails” on page 889
- “Recovering after the broker queue manager fails” on page 890

Related tasks:

“Making initial checks” on page 650

Before you start problem determination in detail, consider whether there is an obvious cause of the problem, or an area of investigation that is likely to give useful results. This approach to diagnosis can often save a lot of work by highlighting a simple error, or by narrowing down the range of possibilities.

“Using logs” on page 839

There are a variety of logs that you can use to help with problem determination and troubleshooting.

“Using trace” on page 845

You can use different types of trace to help you with problem determination and troubleshooting.

“Dealing with problems” on page 666

Learn how to resolve some of the typical problems that can occur.

“Contacting your IBM Support Center” on page 876

If you cannot resolve problems that you find when you use IBM Integration Bus, or if you are directed to do so by an error message generated by IBM Integration Bus, you can request assistance from your IBM Support Center.

Recovering after the broker fails

Check what recovery procedures are available, according to what has failed.

Before you begin

Try to get to the root of the problem first, using the diagnosis techniques described throughout the Chapter 4, “Troubleshooting and support,” on page 647 section of the information center. If your problem cannot be resolved using these techniques, contact your IBM Support Center. Use the procedure in this section only as a last resort.

If you are able to recover, and can start the broker again, the broker attempts to recover and re-establish all sessions with CMP applications that were active at the time of failure. CMP applications include the IBM Integration Explorer, the IBM Integration Toolkit, and applications that you have written to this API.

About this task

If you cannot correct the current problem by using problem determination, complete the following sequence of operations to re-create the broker:

Procedure

1. Stop the broker by using the **mqsistop** command.
2. Stop the broker queue manager by using the **endmqm** command.
3. Delete the broker by using the **mqsdeletebroker** command.
4. Re-create the broker by using the **mqsicreatebroker** command.
5. Start the broker by using the **mqsistart** command.
6. Redeploy all resources to the broker.

Related tasks:



Creating a broker

You can create brokers on every platform that is supported by IBM Integration Bus. The broker runs as a 64-bit application on all platforms except Linux on x86 and Windows on x86.

“Deleting a broker” on page 29

Delete a broker using the command line on the system where the Integration Bus component is installed.

“Recovering after failure” on page 887

Follow a set of procedures to recover after a serious problem.

Related reference:



mqsicreatebroker command

Use the **mqsicreatebroker** command to create a broker and its associated resources.



mqsdeletebroker command

Use the **mqsdeletebroker** command to delete a named broker. The command also deletes the queues on the associated queue manager (created when the broker was created). You can also specify that the queue manager is to be deleted.



mqsistart command

Use the **mqsistart** command to start the specified broker if all initial verification tests complete successfully.



mqsistop command

Use the **mqsistop** command to stop the specified component.

Recovering after an integration server fails

Follow the steps in this task to recover if an integration server fails.

Before you begin

Try to get to the root of the problem first, using the diagnosis techniques described throughout the Chapter 4, “Troubleshooting and support,” on page 647 section of the information center. If your problem cannot be resolved using these techniques, contact your IBM Support Center. Use the procedure in this section only as a last resort.

About this task

If a single integration server fails, and the problem cannot be corrected using problem determination, or by the IBM Support Center, perform the following sequence of operations to re-create the integration server:

Procedure

1. Delete the integration server.
2. Create an integration server of the same name.
3. Redeploy the configuration.

What to do next

If more than one integration server fails, you might need to re-create the broker. See “Recovering after the broker fails” on page 888 for information on how to do this.

Related tasks:

“Recovering after failure” on page 887

Follow a set of procedures to recover after a serious problem.

“Recovering after the broker fails” on page 888

Check what recovery procedures are available, according to what has failed.

Recovering after the broker queue manager fails

Check what recovery procedures are available, according to what has failed.

Before you begin

Try to get to the root of the problem first, by using the diagnosis techniques described throughout the Chapter 4, “Troubleshooting and support,” on page 647 section of the information center. If your problem cannot be resolved by using these techniques, contact your IBM Support Center. Use the procedure in this section only as a last resort.

About this task

If the broker's queue manager fails and cannot be corrected by using problem determination, or by the IBM Support Center, perform the following sequence of operations to re-create the queue manager:

Procedure

1. Ensure that no IBM Integration Toolkit users are deploying to the broker. You must wait until any such actions have completed.
2. Stop the broker by using the **mqsistop** command.
3. Delete the broker by using the **mqsdeletebroker** command, with the **-q** parameter to remove the queue manager.
4. Re-create the broker by using the **mqsicreatebroker** command. The **mqsicreatebroker** command creates the queue manager and default queues automatically.
5. Re-create any specific queues that are needed for your message flows.
6. Start your brokers by using the **mqsistart** command.
7. Redeploy all resources to the broker to ensure that its configuration is consistent.

Related tasks:



Creating a broker

You can create brokers on every platform that is supported by IBM Integration Bus. The broker runs as a 64-bit application on all platforms except Linux on x86 and Windows on x86.

“Deleting a broker” on page 29

Delete a broker using the command line on the system where the Integration Bus component is installed.

“Recovering after failure” on page 887

Follow a set of procedures to recover after a serious problem.

Related reference:



mqsicreatebroker command

Use the **mqsicreatebroker** command to create a broker and its associated resources.



mqsdeletebroker command

Use the **mqsdeletebroker** command to delete a named broker. The command also deletes the queues on the associated queue manager (created when the broker was created). You can also specify that the queue manager is to be deleted.



mqsistart command

Use the **mqsistart** command to start the specified broker if all initial verification tests complete successfully.



mqsistop command

Use the **mqsistop** command to stop the specified component.

Diagnostic messages

Diagnostic messages are listed in this section in numeric order, grouped according to the component to which they relate.

You can also view the full content of BIP messages for runtime components by using the **mqsixplain** command. For more information, see **mqsixplain** command.

- Runtime components (BIP)
- Toolkit (BIP)
- WebSphere Adapters (CWY)
 - PeopleSoft Enterprise (CWYES)
 - SAP Software (CWYAP)
 - Siebel Business Applications (CWYEB)

Diagnostic messages: Runtime components

The diagnostic messages for runtime components are listed in this section in numeric order, grouped according to the component to which they relate.

You can also view the full content of a BIP message by using the **mqsixplain** command. For more information, see **mqsixplain** command.

- BIP1000-1999: Configuration
- BIP2000-2999: Broker
- BIP3000-3999: Built-in nodes
- BIP4000-4999: Built-in nodes
- BIP5000-5999: Parsers
- BIP6000-6999: WebSphere MQ parsers
- BIP7000-7999: Publish Subscribe
- BIP8000-8999: Commands
- BIP9000-9999: z/OS
- BIP11000-11999: Activity log
- BIP12000-12999: Activity log

Diagnostic messages: IBM Integration Toolkit

The diagnostic messages for the IBM Integration Toolkit are listed in this section in numeric order, grouped according to the component to which they relate.

- BIP0000-0999: Toolkit

- BIP01000-01999: Broker archive

Diagnostic messages: WebSphere Adapters

Diagnostic messages are available for the supported WebSphere Adapters.

- PeopleSoft Enterprise (CWYES)
- SAP Software (CWYAP)
- Siebel Business Applications (CWYEB)

Diagnostic messages: WebSphere Adapter for PeopleSoft Enterprise (CWYES)

The diagnostic messages for WebSphere Adapter for PeopleSoft Enterprise are listed in this section in numeric order, grouped according to the component to which they relate.

- CWYES0001-0999: Discovery
- CWYES1001-1999: Discovery connection
- CWYES2001-2999: Dynamic method invoker
- CWYES6001-6999: Managed connection
- CWYES7001-7999: ASIRetriever

Diagnostic messages: WebSphere Adapter for SAP Software (CWYAP)

The diagnostic messages for WebSphere Adapter for SAP Software are listed in this section in numeric order, grouped according to the component to which they relate.

Messages for the SAP Resource Adapter:

- CWYAP1001-1999: Common Components
- CWYAP2001-2999: BAPI
- CWYAP3001-3999: ALE IDoc processing
- CWYAP4001-4999: QISS Runtime module

Messages for the SAP Enterprise Metadata Discovery:

- CWYAP100001-100999: EMD
- CWYAP101001-101999: EMD ALE metadata import
- CWYAP102001-102999: EMD BAPI metadata import
- CWYAP103001-103999: EMD QISS module

Diagnostic messages: WebSphere Adapter for SAP Software - Resource Adapter (CWYAP):

The diagnostic messages for WebSphere Adapter for SAP Software Resource Adapter are listed in this section in numeric order, grouped according to the component to which they relate.

- CWYAP1001-1999: Common Components
- CWYAP2001-2999: BAPI
- CWYAP3001-3999: ALE IDoc processing
- CWYAP4001-4999: QISS Runtime module

Diagnostic messages: WebSphere Adapter for SAP Software - EMD (CWYAP):

The diagnostic messages for WebSphere Adapter for SAP Software Enterprise Metadata Discovery (EMD) are listed in this section in numeric order, grouped according to the component to which they relate.

- CWYAP100001-100999: EMD
- CWYAP101001-101999: EMD ALE metadata import
- CWYAP102001-102999: EMD BAPI metadata import

- CWYAP103001-103999: EMD QISS module

Diagnostic messages: WebSphere Adapter for Siebel Business Applications (CWYEB)

The diagnostic messages for WebSphere Adapter for Siebel Business Applications are listed in this section in numeric order, grouped according to the component to which they relate.

- CWYEB0000-0999: Discovery
- CWYEB1000-1999: Discovery connection
- CWYEB2000-2999: SiebelApp Analyzer
- CWYEB3000-3999: Adapter Classes

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing 2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Software Interoperability Coordinator, Department 49XA
3605 Highway 52 N
Rochester, MN 55901
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Programming interface information

Programming interface information, if provided, is intended to help you create application software for use with this program.

However, this information may also contain diagnosis, modification, and tuning information. Diagnosis, modification and tuning information is provided to help you debug your application software.

Important: Do not use this diagnosis, modification, and tuning information as a programming interface because it is subject to change.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol ([®] or [™]), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at Copyright and trademark information (www.ibm.com/legal/copytrade.shtml).



Figure 26. Java compatibility logo

Sending your comments to IBM

If you especially like or dislike anything about this book, please use one of the methods listed below to send your comments to IBM.

Feel free to comment on what you regard as specific errors or omissions, and on the accuracy, organization, subject matter, or completeness of this book.

Please limit your comments to the information in this book and the way in which the information is presented.

To make comments about the functions of IBM products or systems, talk to your IBM representative or to your IBM authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate, without incurring any obligation to you.

You can send your comments to IBM in any of the following ways:

- By mail, to this address:

User Technologies Department (MP095)
IBM United Kingdom Laboratories
Hursley Park
WINCHESTER,
Hampshire
SO21 2JN
United Kingdom

- By fax:
 - From outside the U.K., after your international access code use 44-1962-816151
 - From within the U.K., use 01962-816151
- Electronically, use the appropriate network ID:
 - IBM Mail Exchange: GBIBM2Q9 at IBMMAIL
 - IBMLink: HURSLEY(IDRCF)
 - Internet: idrcf@hursley.ibm.com

Whichever method you use, ensure that you include:

- The publication title and order number
- The topic to which your comment applies
- Your name and address/telephone number/fax number/network ID.



Printed in USA