WebSphere Message Broker

Migration

Version 6 Release 0

WebSphere Message Broker

Migration

Version 6 Release 0

Note

Before using this information and the product it supports, read the information in the Notices appendix.

Fifth Edition (December 2006)

This edition applies to IBM® WebSphere® Message Broker Version 6.0 and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

About this topic collection

This PDF has been created from the WebSphere Message Broker Version 6.0.0.3 (with Message Brokers Toolkit Version 6.0.2.0 update, December 2006) information center topics. Always refer to the WebSphere Message Broker online information center to access the most current information. The information center is periodically updated on the document update site and this PDF and others that you can download from that Web site might not contain the most current information.

The topic content included in the PDF does not include the ″Related Links″ sections provided in the online topics. Links within the topic content itself are included, but are active only if they link to another topic in the same PDF collection. Links to topics outside this topic collection are also shown, but these attempt to link to a PDF that is called after the topic identifier (for example, ac12340_.pdf) and therefore fail. Use the online information to navigate freely between topics.

Feedback: do not provide feedback on this PDF. Refer to the online information to ensure that you have access to the most current information, and use the Feedback link that appears at the end of each topic to report any errors or suggestions for improvement. Using the Feedback link provides precise information about the location of your comment.

The content of these topics is created for viewing online; you might find that the formatting and presentation of some figures, tables, examples, and so on are not optimized for the printed page. Text highlighting might also have a different appearance.

Part 1. Migrating

Migrating and upgrading

This section describes how to migrate a broker domain to WebSphere Message Broker Version 6.0 from the following products:

- WebSphere MQ Event Broker Version 2.1
- WebSphere MQ Integrator Broker Version 2.1 (CSD02 or later)[1]
- WebSphere MQ Integrator Version 2.1 (CSD02 or later)[1]
- WebSphere Business Integration Event Broker Version 5.0 (Fix Pack 4 or later)
- WebSphere Business Integration Message Broker Version 5.0 (Fix Pack 4 or later)
- WebSphere Business Integration Message Broker with Rules and Formatter Extension Version 5.0 (Fix Pack 4 or later)

Note:

1. If you have applied CSD02, CSD03, CSD04, or CSD05 to Version 2.1, you must apply an additional APAR (IY45459) in order for migration to be successful. APAR IY45459 is included in CSD06.

The instructions apply to all operating systems supported by the Version 2.1 and Version 5.0 products.

Version 5.1 Message Brokers Toolkit

If you are using the Version 5.1 Message Brokers Toolkit, follow the instructions in this section, replacing all references to ″Version 5.0 Message Brokers Toolkit″ with ″Version 5.1 Message Brokers Toolkit″.

Upgrading

This section also shows you how to upgrade a broker domain from WebSphere Event Broker Version 6.0.

If you are already using WebSphere MQ Publish/Subscribe, migrate your applications to use the publish/subscribe functions provided by WebSphere Message Broker.

This section contains the following topics:

Concept topics

- “Coexistence with previous versions and other products” on page 4

Task topics

- “Migrating from Version 2.1 products” on page 11
- “Migrating from Version 5.0 products” on page 51
- “Upgrading from WebSphere Event Broker Version 6.0 to WebSphere Message Broker Version 6.0” on page 72
- “Restoring migrated components to previous versions” on page 72
- “Migrating publish/subscribe applications” on page 77

Coexistence with previous versions and other products

The topics in this section explain the extent to which WebSphere Message Broker Version 6.0 can coexist with previous versions and other products. This topic contains advice for the following scenarios:

- "Coexistence with previous versions of the product installed on the same computer"
- "Coexistence of Version 6.0 components with components from previous versions" on page 5
- "Coexistence with other products" on page 6

Coexistence with previous versions of the product installed on the same computer

WebSphere Message Broker Version 6.0 can coexist with either a Version 2.1 or a Version 5.0 product on the same computer.

You cannot install both a Version 2.1 and a Version 5.0 product on the same computer, and you cannot install several instances of either Version 2.1 or Version 5.0 products on a single computer.

When you migrate from a Version 2.1 or Version 5.0 product to WebSphere Message Broker Version 6.0, you do not need to uninstall the Version 2.1 or Version 5.0 product before installing Version 6.0. You can install WebSphere Message Broker Version 6.0 in a different location on the same computer, migrate your components and resources to Version 6.0, and uninstall the Version 2.1 or Version 5.0 product later, when you are sure that you no longer need it.

The following table shows which Version 2.1 and Version 5.0 components can exist on the same computer as Version 6.0 components:

Can the components exist together on the same computer?	**Version 6.0 Message Brokers Toolkit**	**Version 6.0 Configuration Manager**	**Version 6.0 broker**	**Version 6.0 User Name Server**
Version 2.1 Control Center	Yes	Yes	Yes	Yes
Version 2.1 Configuration Manager	Yes	Yes	Yes	Yes
Version 2.1 broker	Yes	Yes	Yes	Yes
Version 2.1 User Name Server	Yes	Yes	Yes	Yes[1]
Version 5.0 or Version 5.1 Message Brokers Toolkit	Yes	Yes	Yes	Yes
Version 5.0 Configuration Manager	Yes	Yes	Yes	Yes
Version 5.0 broker	Yes	Yes	Yes	Yes

Can the components exist together on the same computer?	Version 6.0 Message Brokers Toolkit	Version 6.0 Configuration Manager	Version 6.0 broker	Version 6.0 User Name Server
Version 5.0 User Name Server	Yes	Yes	Yes	Yes

Restriction:

1. On distributed systems, only one User Name Server can exist on a single computer. If you need another User Name Server, start one on a remote system.

Coexistence of Version 6.0 components with components from previous versions

With some restrictions, all Version 5.0 components can participate in a Version 6.0 broker domain, and all Version 6.0 components can participate in a Version 5.0 broker domain. A Version 2.1 broker is the only Version 2.1 component that can take part in a Version 6.0 broker domain.

Coexistence of Version 6.0 components with components from previous versions means that you do not have to migrate all of your components at the same time. Instead, migrate those components that can coexist with Version 6.0 components in stages.

The following table shows which Version 2.1 and Version 5.0 components can operate with Version 6.0 components, and shows when restrictions apply:

Can the components operate together in a domain?	Version 6.0 Message Brokers Toolkit	Version 6.0 Configuration Manager	Version 6.0 broker	Version 6.0 User Name Server
Version 2.1 Control Center	No	No	No	No
Version 2.1 Configuration Manager	No	(Not applicable)	No	No
Version 2.1 broker	Yes[1]	Yes[1]	Yes[1]	Yes[1]
Version 2.1 User Name Server	(Not applicable)	No	No	(Not applicable)
Version 5.0 or Version 5.1 Message Brokers Toolkit	Yes[2]	Yes[3]	Yes	Yes
Version 5.0 Configuration Manager	Yes	(Not applicable)	Yes[4]	Yes
Version 5.0 broker	Yes[5]	Yes	Yes	Yes
Version 5.0 User Name Server	(Not applicable)	Yes	Yes	(Not applicable)

Restrictions:

1. There are restrictions on how a Version 2.1 broker can participate in a Version 6.0 domain. For full details, see “Conditions for a Version 2.1 broker participating in a Version 6.0 broker domain” on page 7.
2. There are restrictions on sharing resources between Message Brokers Toolkit Version 6.0 and Message Brokers Toolkit Version 5.0 or Version 5.1. For full details, see “Conditions for using migrated resources with previous versions of the Message Brokers Toolkit”
3. You cannot use a Version 5.0 or Version 5.1 Message Brokers Toolkit to connect to a Version 6.0 Configuration Manager with a domain user ID.
4. If you deploy through a Version 5.0 Configuration Manager, only message flows and dictionaries are extracted from the broker archive file. If you are using a Version 5.0 Configuration Manager to deploy to a Version 6.0 broker, any .XSL, .XML, and .JAR resources that are required by the Java Compute and XML Transformation nodes are not deployed automatically.
5. If you want to use a Version 6.0 Message Brokers Toolkit and a Version 5.0 Configuration Manager to deploy to a Version 5.0 broker that is part of a WebSphere Business Integration Message Broker Version 5.0 installation at service level Fix Pack 3 or earlier, apply Fix Pack 5 to the WebSphere Business Integration Message Broker Version 5.0 installation of which the Version 5.0 Configuration Manager is part.

Coexistence with other products

You can create and operate a heterogeneous network of WebSphere MQ Publish/Subscribe brokers and WebSphere Message Broker brokers. You can also migrate WebSphere MQ Publish/Subscribe brokers to WebSphere Message Broker brokers. Follow the instructions in “Migrating publish/subscribe applications” on page 77.

You cannot install WebSphere Event Broker and WebSphere Message Broker as separate products on the same computer. WebSphere Message Broker, however, can perform all of the functions of WebSphere Event Broker.

Conditions for using migrated resources with previous versions of the Message Brokers Toolkit

WebSphere Message Broker Version 6.0 can coexist with previous versions and other products. This topic explains the conditions that apply when you use migrated resources with previous versions of the Message Brokers Toolkit.

Except for user-defined extensions and mapping files, you do not need to perform any tasks to migrate your Version 5.0 development and deployment resources, such as message flow files, message set definition files, ESQL files, XML Schema files, and broker archive files. You can start using your Version 5.0 resources with WebSphere Message Broker Version 6.0 immediately. After you have migrated Version 5.0 mapping files (.mfmap) to Version 6.0 mapping files (.msgmap) using the mqsimigratemfmaps command, edit them in the Version 6.0 Message Brokers Toolkit. (There is no migration tool to migrate a mapping that is called from ESQL.)

When you migrate from Version 2.1, you must also migrate your message flows, message sets, and user-defined extensions using the **mqsimigratemsgflows** command and **mqsimigratemsgsets** command.

However, after you start using your resources in the Version 6.0 Message Brokers Toolkit, there are restrictions on using these same resources again with the Version 5.0 or Version 5.1 Message Brokers Toolkit. These are described in the following sections.

Development resources

When you use the Version 6.0 Message Brokers Toolkit to save development resources (such as message flow files, message set definition files, ESQL files, XML Schema files) that were originally created in the Version 5.0 or Version 5.1 Message Brokers Toolkit, the resources are saved in Version 6.0 format.

After you have saved resources in the Version 6.0 format, you can no longer use them in the Version 5.0 or Version 5.1 Message Brokers Toolkit. This means that you cannot share development resources between the Version 6.0 Message Brokers Toolkit and previous versions of the Message Brokers Toolkit.

Files in the Version 5.0, Version 5.1, and Version 6.0 formats can coexist in the Version 6.0 workspace. Resources remain in the Version 5.0 or Version 5.1 format until you save them using the Version 6.0 Message Brokers Toolkit.

Deployment resources

You can deploy broker archive (bar) files created in the Version 5.0 or Version 5.1 Message Brokers Toolkit using the Version 6.0 Message Brokers Toolkit.

You can deploy bar files that are created in the Version 6.0 Message Brokers Toolkit using the Version 5.0 or Version 5.1 Message Brokers Toolkit, provided that the development resources referenced in the bar file do not make use of any new features that are available only in WebSphere Message Broker Version 6.0.

You can create a Version 6.0 bar file that contains Version 5.0 or Version 5.1 compiled message flows and message dictionaries.

The Version 6.0 Message Brokers Toolkit can coexist with either the Version 5.0 or Version 5.1 Message Brokers Toolkit on the same computer. However, because you cannot share development resources between the Version 6.0 Message Brokers Toolkit and previous versions of the Message Brokers Toolkit, it is recommended that the Version 6.0 Message Brokers Toolkit uses a different workspace to either the Version 5.0 or Version 5.1 Message Brokers Toolkit.

The Version 5.0 and Version 5.1 Message Brokers Toolkit cannot coexist on the same computer.

Conditions for a Version 2.1 broker participating in a Version 6.0 broker domain

WebSphere Message Broker Version 6.0 can coexist with previous versions and other products. This topic explains the conditions that apply when you use a Version 2.1 broker in a Version 6.0 broker domain.

If you preserve a broker at the Version 2.1 level when migrating to WebSphere Message Broker Version 6.0, the broker can subsequently participate in the Version 6.0 broker domain and can be managed by the Version 6.0 Configuration Manager. The Version 2.1 broker can be from either of the following products:

- WebSphere MQ Integrator Broker Version 2.1 at service level CSD06 or later
- WebSphere MQ Integrator Version 2.1 at service level CSD04 or later

No other components of these products, or any previous releases of these products, can participate in a Version 6.0 broker domain. This means that you must migrate all your Version 2.1 Control Center, Configuration Manager, and User Name Server components at the same time, but you can migrate your Version 2.1 brokers in stages.

It is not possible to preserve a WebSphere MQ Event Broker broker at the Version 2.1 level when you migrate a WebSphere MQ Event Broker Version 2.1 broker domain to a Version 6.0 broker domain.

There are some conditions for a Version 2.1 broker to participate in a WebSphere Message Broker Version 6.0 broker domain:

- A Version 2.1 broker and a Version 6.0 broker must use separate sets of database tables. This means that, if a Version 2.1 broker and a Version 6.0 broker share a database, they must use different database schemas.
- Only certain types of configuration data can be deployed to a Version 2.1 broker, as shown in the table:

Configuration data	**Can it be deployed to a Version 2.1 broker?**	**Conditions**
Message flows	Yes	See “Conditions for deploying a message flow to a Version 2.1 broker.”
Message sets	Yes	See “Conditions for deploying a message set to a Version 2.1 broker” on page 10.
Topology	Yes	There are no conditions.
Topics	Yes	A Version 2.1 broker uses only the access control list for the topic and ignores any multicast or quality of protection (QoP) settings for the topic.
Broker properties	No	Cannot be deployed to a Version 2.1 broker.

After migration, a Version 2.1 broker can continue to use the same WebSphere MQ queue manager, with the same WebSphere MQ configuration, and with the same release and service level of the WebSphere MQ product code, as it did before migration.

Conditions for deploying a message flow to a Version 2.1 broker

WebSphere Message Broker Version 6.0 can coexist with previous versions and other products. This topic explains the conditions that apply when you deploy a message flow to a Version 2.1 broker.

If you want to deploy a message flow to a Version 2.1 broker that is participating in a Version 6.0 broker domain, the message flow must not use any capability that is not present in Version 2.1 brokers; for example:

- The message flow must not contain any of the following built-in nodes:
  - DataDelete
  - DataInsert

- DataUpdate
- Extract
- HTTPInput
- HTTPReply
- HTTPRequest
- JavaCompute
- JMSInput
- JMSOutput
- JMSMQTransform
- MQJMSTransform
- Mapping
- MQGet
- Passthrough
- Real-timeInput
- Real-timeOptimizedFlow
- TimeoutControl
- TimeoutNotification
- Warehouse
- XMLTransformation

Use the following alternative nodes:

Node unavailable in Version 2.1	Alternative node
Extract node	Compute node
Mapping node	Compute node
DataDelete node	Database node
DataInsert node	Database node
DataUpdate node	Database node
Warehouse node	Database node

- The message flow must not contain any of the following elements in its ESQL files:
  - A LOG or RESIGNAL statement
  - An ACOS, ASIN, ATAN, ATAN2, COS, COSH, COT, DEGREES, EXP, FIELDNAMESPACE, LEFT, LN, LOG, LOG10, POWER, RADIANS, RAND, REPLICATE, RIGHT, SIGN, SIN, SINGULAR, SPACE, TAN, TANH, OR TRANSLATE function
  - A PATH clause
  - A BEGIN statement with an ATOMIC keyword
  - A CREATE FUNCTION or CREATE PROCEDURE statement at schema level (i.e. not within a module)
  - A DECLARE statement that uses any of the EXTERNAL, HANDLER, NAMESPACE, NAME, ROW, or SHARED keywords
  - A DELETE statement that uses any of the FIELD, FIRSTCHILD, LASTCHILD, PREVIOUS SIBLING, or NEXTSIBLING keywords
  - A PROPAGATE statement with any clauses of any kind
  - A CAST function with either of the DEFAULT or FORMAT keywords
  - A POSITION function with either of the FROM or REPEAT keywords

  - A SELECT function that is not enclosed in parentheses
  - A dynamically calculated database data source name or schema name
  - Any construct that attempts to change the Root or InputRoot message trees

  In addition, the broker schema must not contain a mapping file.

  These conditions mean that the broker schema can contain only module definitions.
- When you add ESQL to a bar file that is being deployed to Version 2.1, select the **Compile ESQL for broker Version 2.1** check box in the Broker Archive editor.
- There are restrictions on some parameters of existing nodes:
  - For input nodes that have a drop-down list of parsers, such as MQInput and MQeInput, do not select the XMLNSC or MIME parsers.
  - For nodes that have parser options, such as MQInput and Compute, do not specify non-default parser options.
  - For the MQOutput node, do not specify non-default Validate options.

Conditions for deploying a message set to a Version 2.1 broker

WebSphere Message Broker Version 6.0 can coexist with previous versions and other products. This topic explains the conditions that apply when you deploy a message set to a Version 2.1 broker.

If you want to deploy a message set to a Version 2.1 broker that is participating in a Version 6.0 broker domain, the message set must satisfy the following conditions:

- For the logical model:
  - The Use Namespaces property of the message set must not be selected.
  - Integer values used in value constraints must be in the signed 32-bit integer range.
  - Datetime values used in value constraints must not specify a time zone.
- For an XML physical format:
  - The Strict DateTime checking property of the message set must not be selected.
  - The Daylight Savings Time property of the message set must not be selected.
  - The Encoding Numeric Null property of the message set must not have the value NULLXMLSchema.
  - The Encoding Non-Numeric Null property of the message set must not have the value NULLXMLSchema.
- For a Custom Wire Format (CWF) physical format:
  - The Strict DateTime checking property of the message set must not be selected.
  - The Daylight Savings Time property of the message set must not be selected.
  - The Trailing Skip Count property must have the value 0 (zero).
- For a Tagged Delimited String (TDS) physical format:
  - The Strict DateTime checking property of the message set must not be selected.
  - The Daylight Savings Time property of the message set must not be selected.
  - If the Suppress Absent Element Delimiters property of a group or complex type is set, it must have the value End of Type.

Conditions for a Version 5.0 broker participating in a Version 6.0 broker domain

WebSphere Message Broker Version 6.0 can coexist with previous versions and other products. This topic explains the conditions that apply when you use a Version 5.0 broker in a Version 6.0 broker domain.

If you preserve a broker at the Version 5.0 level of code during migration to WebSphere Message Broker Version 6.0, the broker can subsequently participate in the Version 6.0 broker domain and can be managed by the Version 6.0 Configuration Manager. The Version 5.0 broker can be from either of the following products:

- WebSphere Business Integration Message Broker Version 5.0 (Fix Pack 4 or later)
- WebSphere Business Integration Message Broker with Rules and Formatter Extension Version 5.0 (Fix Pack 4 or later)

The conditions under which a Version 5.0 broker can participate in a WebSphere Message Broker Version 6.0 broker domain are as follows:

- A Version 5.0 broker and a Version 6.0 broker must use separate sets of database tables. This means that if a Version 5.0 broker and a Version 6.0 broker share a database, they must use different database schemas.
- You can use a Version 6.0 Message Brokers Toolkit and a Version 5.0 Configuration Manager to deploy to a Version 5.0 broker that is part of a WebSphere Business Integration Message Broker Version 5.0 installation. However, if WebSphere Business Integration Message Broker Version 5.0 is at service level Fix Pack 3 or earlier, apply Fix Pack 5 to the installation of which the Version 5.0 Configuration Manager is part.

After migration, a Version 5.0 broker can continue to use the same WebSphere MQ queue manager, with the same WebSphere MQ configuration, and with the same release and service level of the WebSphere MQ product code, as it did before migration.

Except for user-defined extensions and mapping files, you do not need to perform any tasks to migrate your development and deployment resources, such as message flow files, message set definition files, ESQL files, XML Schema files, and broker archive files. You can just start using these resources with WebSphere Message Broker Version 6.0. When you have migrated Version 5.0 mapping files (.mfmap) to Version 6.0 mapping files (.msgmap) using the **mqsimigratemfmaps** command, edit them in the Version 6.0 Message Brokers Toolkit.

It is not possible to preserve a WebSphere MQ Event Broker broker at the Version 5.0 level of code when you migrate a WebSphere Business Integration Event Broker Version 5.0 broker domain to a Version 6.0 broker domain.

Migrating from Version 2.1 products

This section describes how to migrate to WebSphere Message Broker Version 6.0 or WebSphere Message Broker with Rules and Formatter Extension Version 6.0 from the following products:

- WebSphere MQ Event Broker Version 2.1
- WebSphere MQ Integrator Broker Version 2.1 (CSD02 or later)[1]
- WebSphere MQ Integrator Version 2.1 (CSD02 or later)[1]

Note:

1. If you have applied CSD02, CSD03, CSD04, or CSD05 to Version 2.1, you must apply an additional APAR (IY45459) in order for migration to be successful. APAR IY45459 is included in CSD06.

If you are already using WebSphere MQ Publish/Subscribe, migrate your applications to use the publish/subscribe functions provided by WebSphere Message Broker. Follow the instructions in "Migrating publish/subscribe applications" on page 77.

The instructions apply to all platforms supported by the Version 2.1 products. This section contains the following topics:

- "Planning and pre-migration tasks"
- "Migrating from WebSphere MQ Integrator Broker Version 2.1 to WebSphere Message Broker Version 6.0" on page 16
- "Migrating from WebSphere MQ Event Broker Version 2.1 to WebSphere Message Broker Version 6.0" on page 39
- "Migrating from WebSphere MQ Integrator Version 2.1 to WebSphere Message Broker Version 6.0" on page 48
- "Post-migration tasks" on page 48

Planning and pre-migration tasks

The following topics describe what to do before migrating from Version 2.1 to Version 6.0:

- "Differences between Version 2.1 and Version 6.0 products"
- "Planning for migration from Version 2.1" on page 15
- "Backing up Version 2.1 resources" on page 16

Differences between Version 2.1 and Version 6.0 products

The following table outlines the main differences between the Version 2.1 and Version 6.0 products:

Version 2.1	Version 6.0	Why has this changed?
Installing the product		
In Version 2.1, there can be only one copy of the product installed on the computer.	You can install the Version 6.0 product alongside the Version 2.1 product and the two products can run side-by-side. The new **mqsimigratecomponents** command allows you to migrate components from Version 2.1 to Version 6.0.	Permitting Version 6.0 to coexist with Version 2.1 allows for easier migration.
Entering commands on Windows		
You enter Version 2.1 commands in a command window.	You enter Version 6.0 commands in the Command Console, which is in the **Start** menu: **Start** → **Programs** → **IBM WebSphere Message Brokers 6.0** → **Command Console**.	Multiple versions of the product can now coexist. The Command Console ensures that the commands that you enter are directed to the correct version.
Configuration Manager Database Requirements		

Version 2.1	Version 6.0	Why has this changed?
You must specify two data sources when creating a Version 2.1 Configuration Manager: a configuration database for message flow source and domain information, and an MRM database for message dictionary source. When deleting a Configuration Manager, there are flags to remove the contents of these repositories (**-n** and **-m** for the configuration and MRM sources respectively).	There are no database requirements on the Configuration Manager. Source data for both message flows and dictionaries is now stored inside the Eclipse workspace. Domain information is stored in an internal configuration repository in the file system on the Configuration Manager computer, and there are commands to back up and restore this repository (**mqsibackupconfigmgr** and **mqsirestoreconfigmgr**). When you delete a Configuration Manager, the **-n** parameter now removes the contents of the internal repository and the **-m** parameter is ignored.	Removing the requirements on an external database simplifies the setup and administration of a Configuration Manager.
Starting the Control Center		
When you start the Control Center, a prompt opens for the connection details of the Configuration Manager.	The Control Center is now called the Message Brokers Toolkit and is based on the Eclipse framework. To connect the Configuration Manager, you must create a connection file by clicking **File** → **New** → **Other** → **Broker Administration** → **Domain**.	Each set of Configuration Manager connection parameters is a file stored within the Eclipse workspace. This allows you to share connection details more easily among multiple users, and allows you to manage multiple domains from a single Message Brokers Toolkit.
Displaying information in the Control Center		

Version 2.1	Version 6.0	Why has this changed?
The Version 2.1 Control Center contains a set of tabs that determine the type of information that is shown (for example, assignments, topology, topics, subscriptions).	In Version 6.0, there are two main screens of information (called perspectives): • **Broker Application Development perspective**, which allows you to create and edit message flows and message sets. • **Broker Administration perspective**, which shows you the state of the domain, as reported by the Configuration Manager, and allows you to manipulate it. It is possible to switch perspectives using the **Window** menu.	The concept of perspectives is inherited from the Eclipse framework and allows you to switch easily to the information set that is most relevant to what you are doing.
Modifying message flows		
In Version 2.1, message flows under development are stored in the Configuration Manager. In order to modify them, you must check them out, make your changes and then check them back in.	In Version 6.0, message flows are created and modified in an Eclipse workspace; by default, this is on the file system on which the Message Brokers Toolkit is installed.	Storing development objects in an Eclipse workspace allows you to manage them more effectively. Plug in third-party change-management tools into the Eclipse framework to provide better change control.
Assigning and deploying message flows and message sets		
In Version 2.1, message sets were assigned to brokers and were available to all execution groups in that broker, and message flows were assigned to execution groups. When this is done, the deploy action causes the Configuration Manager to send the assigned message flows to the correct execution group.	In Version 6.0, message flows and message sets are packaged in broker archives (bar files). These archives use a zipped file format to envelop a set of deployable objects. To deploy a bar file to an execution group, use the Broker Administration perspective to drag and drop the bar file to the required execution group.	Bar files provide a transferable package of deployment, so that the same message flow logic can be applied to multiple brokers and domains. Similar deployment concepts are used in other products.
Monitoring message flows		
In Version 2.1, traffic lights show the run state of message flows.	In Version 6.0, the Broker Administration perspective has an Alerts view that identifies any broker, execution group or message flow that is not running.	The use of the Alerts view is consistent with other tools in the Eclipse framework and provides an accessible way of notifying you if anything is not processing messages as expected.
User roles		

Version 2.1	Version 6.0	Why has this changed?
In Version 2.1, several Windows user groups in the Configuration Manager are used to describe the typical role, and domain access, that each user has (for example, **mqbrkrs**, **mqbrops**, **mqbrtpic**, **mqbrasgn**).	Membership of the **mqbrkrs** group is still required for administration of the broker component, but the **mqsicreateaclentry** command is now used to define specific user or group access control lists (ACLs) for each component (such as brokers and execution groups).	Access control lists (ACLs) in the Configuration Manager allow administrators to control access to the domain at a much finer level of granularity.

Planning for migration from Version 2.1

Before you start:

Before you carry out any migration tasks, back up Version 2.1 resources and read “Coexistence with previous versions and other products” on page 4.

This section explains the different ways in which to migrate from Version 2.1 to Version 6.0. Before carrying out any migration tasks, you should make the following decisions:

1. Decide how you want to migrate the product components:
   a. Find out what is new in Version 6.0. These new and changed functions might affect how you want to use your migrated components in the future.
   b. Decide where you want to migrate the product components. Migrate your components to the same computer or to a second computer. For example, you might want to migrate components to another location to maintain availability during the migration.
   c. Decide when you want to migrate the product components. You might want to preserve some components at the Version 2.1 level of code temporarily, and migrate them later.
   d. Decide the order in which you want to migrate your components. You can migrate components in any order, but your specific circumstances might mean that you need to migrate components in a particular order.

   See “Coexistence with previous versions and other products” on page 4 for information about how Version 6.0 can coexist on the same computer with Version 2.1, and how Version 6.0 components can operate with Version 2.1 components.
2. Decide how you want to use your existing resources with WebSphere Message Broker Version 6.0.

   You migrate your Version 2.1 message flows, message sets, and user-defined extensions using the **mqsimigratemsgflows** command and **mqsimigratemsgsets** command. The tooling part of any Version 2.1 user-defined extensions that you have migrated needs to be rewritten in Version 6.0. Decide which user-defined extensions you want to use in Version 6.0 and re-create them in the Version 6.0 tooling.

   When you start using your resources in the Version 6.0 Message Brokers Toolkit, there are restrictions to using these same resources again with the Version 5.0 or Version 5.1 Message Brokers Toolkit. For more information, see “Conditions for using migrated resources with previous versions of the Message Brokers Toolkit” on page 6

3. Decide whether you need to carry out any testing to ensure a successful migration.

   Migrating your development and test domains before migrating your production domain allows you to identify problems and develop a strategy for dealing with further problems.
4. When you are ready to migrate, run the **mqsimigratecomponents** command with the **-c** parameter. This performs a pre-migration check against the Version 2.1 components to ensure that they can be migrated. The pre-migration check identifies potential problems and allows you to correct them before proceeding with migration.

You do not need to change the configuration of a queue manager that is preserved during migration.

The other topics in this section provide instructions for different migration scenarios:
- “Migrating from Version 2.1 to Version 6.0: migrating applications and runtime components” on page 19
- “Migrating from Version 2.1 to Version 6.0: Version 6.0 coexisting temporarily with Version 2.1” on page 33

Backing up Version 2.1 resources

Before you carry out any migration tasks, back up your Version 2.1 resources:
1. Back up the configuration repository.
2. Back up your broker database tables.
3. Stop any debug sessions and back up your broker application logic (message flow files, message set definition files, and user-defined node definition files).

For detailed instructions on how to back up the configuration repository and broker database tables, see your Version 2.1 documentation. For information on backing up message flow files, message set definition files, and user-defined node definition files, see “Preparing to migrate from WebSphere MQ Integrator Broker Version 2.1” on page 17.

Now that you have backed up your resources, you are ready to perform migration. Go to the appropriate topic for your product:
- Migrating from WebSphere MQ Integrator Broker
- Migrating from WebSphere MQ Event Broker
- Migrating from WebSphere MQ Integrator

Migrating from WebSphere MQ Integrator Broker Version 2.1 to WebSphere Message Broker Version 6.0

Before you start

Read the following topics:
- “Planning for migration from Version 2.1” on page 15
- “Backing up Version 2.1 resources”
- “Preparing to migrate from WebSphere MQ Integrator Broker Version 2.1” on page 17
- “Coexistence with previous versions and other products” on page 4

The topics in this section describe two different migration scenarios that are available to you when migrating from WebSphere MQ Integrator Broker Version 2.1 to WebSphere Message Broker Version 6.0. The following table lists each scenario, summarizes what is involved in that scenario, and tells you where to find the detailed, step-by-step instructions:

Scenario	Summary	Detailed instructions
Scenario 1		
You have a domain of Version 2.1 components that you want to migrate to Version 6.0, while maintaining the state of each component.	Application and server migration: migrate your applications and runtime components. This scenario allows you to maintain the state of the broker, for example, publish/subscribe retained publications and subscriptions.	Follow the instructions in "Migrating from Version 2.1 to Version 6.0: migrating applications and runtime components" on page 19
Scenario 2		
You have a Version 2.1 domain and you want to create a Version 6.0 domain that is running the same applications, and is running in parallel with the Version 2.1 domain.	Application-only migration: migrate your applications and create new runtime components, leaving the Version 2.1 components as they are. In this scenario, you must re-create the state of your broker, such as publish/subscribe retained publications and subscriptions, after you have redeployed the message flow applications.	Follow the instructions in "Migrating from Version 2.1 to Version 6.0: Version 6.0 coexisting temporarily with Version 2.1" on page 33

Preparing to migrate from WebSphere MQ Integrator Broker Version 2.1

This topic describes what to do before you start to migrate a WebSphere MQ Integrator Broker Version 2.1 broker domain in any of the following tasks:

- "Migrating from WebSphere MQ Integrator Broker Version 2.1 to WebSphere Message Broker Version 6.0" on page 16
- "Migrating from WebSphere MQ Event Broker Version 2.1 and WebSphere MQ Integrator Broker Version 2.1 to WebSphere Message Broker Version 6.0" on page 42

1. Make sure that Control Center users have checked in all WebSphere MQ Integrator Broker resources.
2. Back up all configuration repository, message repository, and broker database tables.
3. Decide which message flows and message sets you want to migrate and use in the WebSphere Message Broker broker domain and export them:
   a. Stop any debugging sessions in the Control Center. It is not possible to migrate message flows that are being debugged.
   b. Using a Control Center session in which all the required message flows are visible in the workspace, export the message flows. Save the export files in a directory other than the one in which WebSphere MQ Integrator Broker is

installed. The export files also contain information about the user-defined nodes that are used by the message flows.

Alternatively, on the system where the Configuration Manager is running, export all the required Version 2.1 message sets using the **mqsiimpexpmsgset** command with the **-e** parameter. Save the export files in a directory other than the one in which WebSphere MQ Integrator Broker is installed.

4. Decide how you are going to migrate the brokers:
   - Decide which brokers you no longer require after migration.
   - Decide which brokers you want to migrate from Version 2.1 to Version 6.0.
   - Decide which brokers you want to preserve at Version 2.1.

   If two or more of the brokers share the same set of database tables, and you still require these brokers after migration, either migrate all of the brokers at the same time or preserve them all at Version 2.1.
5. For each broker that you want to migrate from Version 2.1 to Version 6.0, and for the associated assignments configuration data you want to preserve, record the information in the following list. Record this information manually from a Control Center session or export everything in the Control Center workspace by clicking **File → Export All in Workspace**, save the export file in a directory other than the one in which WebSphere MQ Integrator Broker is installed and extract the required information from the export file. (To find out how to do this, see "Assignments configuration data in an export file" on page 124.) Record:
   - The name of the broker
   - The name of each message set that is assigned to the broker
   - The name of each execution group within the broker
   - For each execution group within the broker, the name of each message flow that is assigned to the execution group
   - For each message flow assigned to an execution group, the following properties:
     – Additional instances
     – Commit count
     – Commit interval
     – Coordinated transaction
6. Decide whether you want to preserve the following configuration data, which is stored in the configuration repository:
   - Assignments data
   - Topology data
   - Topics data

   Unless you decide that you no longer require any of your existing brokers after migration, you must preserve this configuration data.
7. Decide which components of WebSphere Message Broker you want to run on each of your systems after migration.

   Consider the following criteria when making your decision:
   - A broker must run on the same system and use the same queue manager as it did before migration, unless you decide that you no longer need the broker after migration.

- The Configuration Manager must run on the same system and use the same queue manager as it did before migration, unless you decide not to preserve the assignments, topology, and topics data in the configuration repository.
- A workbench can run on any system after migration. It does not have to run on a system where Control Center sessions used to run.
- To avoid adding unnecessary complexity to the migration process itself, run a User Name Server on the same system, and configure it to use the same queue manager as it did before migration. If required, change the location of a User Name Server and the queue manager it uses after successful migration.

You do not need to change the configuration of a queue manager that is preserved during migration. You might, however, need to ensure that the WebSphere MQ product code is at the required release and service level to support WebSphere Message Broker Version 6.0. At the same time, you might want to ensure that you have installed the other software prerequisites.

If you are not able to stop a broker during migration because it is doing critical work, but you want to migrate the broker to Version 6.0:

a. Create a new Version 2.1 broker on a system on which you are not going to install WebSphere Message Broker.
b. Using a Control Center session, deploy to the new broker all the configuration data that was deployed to the original broker.

The new broker can then take over the work of the original broker during the migration.

If you intend to preserve any brokers at Version 2.1, see "Conditions for a Version 2.1 broker participating in a Version 6.0 broker domain" on page 7 for what you need to consider in this case.

8. Decide where you are going to store the development data that is created and maintained in the workbench. Store the data in the local file system, on a shared drive, or in a shared repository that is supported by Eclipse. The instructions for the individual migration tasks assume that you are using the local file system or a shared drive.
9. When you are ready to migrate, run the **mqsimigratecomponents** command with the **-c** parameter. This performs a pre-migration check against the Version 2.1 components to ensure that they can be migrated. The pre-migration check identifies any potential problems and allows you to correct them before proceeding with migration.

When you have run the pre-migration check successfully, you are ready to perform migration. There are three scenarios for migration to Version 6.0:

- "Migrating from Version 2.1 to Version 6.0: migrating applications and runtime components"
- "Migrating from Version 2.1 to Version 6.0: Version 6.0 coexisting temporarily with Version 2.1" on page 33

Migrating from Version 2.1 to Version 6.0: migrating applications and runtime components

This topic describes one of two migration scenarios that are available to you when migrating from WebSphere MQ Integrator Broker Version 2.1 to WebSphere Message Broker Version 6.0.

Before you start:

- Back up your Version 2.1 resources

- Read the following topics:
  - “Planning for migration from Version 2.1” on page 15
  - “Coexistence with previous versions and other products” on page 4

In this migration scenario, your Version 2.1 domain components might be on several computers, but the steps are the same for each. After migration to Version 6.0, you no longer need to keep Version 2.1. The following topics tell you how to migrate individual components. The Version 2.1 tooling and Configuration Manager do not work with Version 6.0 components, so you must migrate components in the order shown:
1. Migrate the tooling
2. Migrate the Configuration Manager
3. Migrate the User Name Server
4. Migrate the broker

Migrating the tooling:

Before you start

Record the following information for the broker and for the associated assignments configuration data that you want to preserve:
- The name of the broker
- The name of each message set that is assigned to the broker
- The name of each execution group within the broker
- For each execution group within the broker, the name of each message flow that is assigned to the execution group
- For each message flow assigned to an execution group, the following properties:
  - Additional instances
  - Commit count
  - Commit interval
  - Coordinated transaction.

View this information in the workbench and record it manually.

This topic describes the first step in migrating individual components from WebSphere MQ Integrator Broker Version 2.1 to WebSphere Message Broker Version 6.0.
1. Use the Control Center to export all Version 2.1 resources, such as message flows, message sets, and user-defined nodes.
2. Stop WebSphere MQ Integrator Broker Version 2.1.
3. Uninstall the Version 2.1 Control Center.
4. Install the Version 6.0 Message Brokers Toolkit in a different location to where the Version 2.1 Message Brokers Toolkit is installed.
5. Import all Version 2.1 message flows, message sets, and user-defined extensions into Eclipse projects using the **mqsimigratemsgflows** and **mqsimigratemsgsets** commands.

Assignments data is not migrated automatically. After migration, if you want to deploy message flows with the same pre-migration parameters, compile the migrated message flows and message sets using the Version 6.0 Message Brokers Toolkit to create broker archive (bar) files, then update any configurable parameters to match the Version 2.1 settings that you recorded before migration. See the Deployment overview topic for more information.

After you have migrated the tooling, migrate your message flows and message sets and the Configuration Manager.

Migrating a Configuration Manager from WebSphere MQ Integrator Broker Version 2.1 to WebSphere Message Broker Version 6.0:

This topic describes the second step in migrating individual components from WebSphere MQ Integrator Broker Version 2.1 to WebSphere Message Broker Version 6.0.

Before you start

Before you migrate the Configuration Manager, migrate the tooling.

In WebSphere Message Broker Version 6.0, the Configuration Manager no longer uses an external database to store the domain configuration but uses an internal repository instead. The Configuration Manager is also now available on more operating systems than in WebSphere MQ Integrator Broker Version 2.1.

From the following list of topics in this section, select the topic that is appropriate to your environment.
- "Migrating a Version 2.1 Configuration Manager to Version 6.0: to a different computer that has DB2 installed" on page 22
- "Migrating a Version 2.1 Configuration Manager to Version 6.0: to a different computer that does not have DB2 installed" on page 24
- "Migrating a Version 2.1 Configuration Manager to Version 6.0 on the same computer"

Migrating a Version 2.1 Configuration Manager to Version 6.0 on the same computer:

This topic describes one of three ways in which to migrate a Version 2.1 Configuration Manager to Version 6.0.

Before you start

Read about "Coexistence with previous versions and other products" on page 4.

The following steps describe how to replace the Version 2.1 Configuration Manager with a Version 6.0 Configuration Manager:
1. Install WebSphere Message Broker Version 6.0 in a location other than where Version 2.1 is installed.
2. Stop the Version 2.1 Configuration Manager.
3. Use the **mqsimigratecomponents** command to migrate your Version 2.1 Configuration Manager.
4. Start the Version 6.0 Configuration Manager.

If you do not have additional hardware available but want to confirm the successful migration of the Configuration Manager before replacing the Version 2.1 Configuration Manager, migrate the Configuration Manager to a different location on the same computer. If you use this method, you can also change the name of the Configuration Manager. Complete the following steps:

1. Install WebSphere Message Broker Version 6.0 in a location other than where Version 2.1 is installed.
2. Stop the Version 2.1 Configuration Manager.
3. Create a Version 6.0 Configuration Manager using the **mqsicreateconfigmgr** command. Make sure that the Version 6.0 Configuration Manager has a different name and a different queue manager name to the Version 2.1 Configuration Manager. Use the **-n** parameter of the **mqsicreateconfigmgr** command to specify the Version 2.1 Configuration Manager's database.
4. Configure WebSphere MQ to allow the Version 6.0 Configuration Manager to communicate with the broker network. For example, you might need to configure channels and transmission queue definitions.
5. Start the Version 6.0 Configuration Manager.
6. In order to associate all brokers in the domain with the new Configuration Manager, so that the brokers publish their status messages to the correct queue manager, a complete topology deploy of the new Configuration Manager is required. Deploy the topology using either the Message Brokers Toolkit or the command-line interface. When the topology deploy has completed re-subscribing to the brokers in the domain, the brokers are managed by the new Configuration Manager; do not use the previous Configuration Manager. You can now delete the previous Configuration Manager.

If you have configured MCA users on the WebSphere MQ channels to the Configuration Manager, you might encounter problems when you try to connect to the migrated Configuration Manager from the Version 6.0 Message Brokers Toolkit. To resolve this, configure access control lists (ACLs) for users who are running the Version 6.0 Message Brokers Toolkit. Follow the instructions in Considering security for the workbench.

Before you migrate your brokers, or make any configuration changes, ensure that your Configuration Manager has been migrated correctly by performing the following checks:
1. Start the migrated Configuration Manager.
2. Start the Message Brokers Toolkit and connect it to the Configuration Manager's domain.
3. Check the list of domain components and see if any errors are produced.
4. Check the list of alerts for the domain and see if anything has stopped.
5. Check that you can open the Event Log.
6. Optional: If you have a broker running, stop and start a message flow, and check for a successful response; check that the Alerts view is updated when the message flow is stopped, and clear when the message flow is started.

If these steps are successful, your Configuration Manager has been migrated successfully. If they are unsuccessful, resolve any problems before you migrate further components or make configuration changes.

After you have migrated the Configuration Manager, migrate the User Name Server.

Migrating a Version 2.1 Configuration Manager to Version 6.0: to a different computer that has DB2 installed:

This topic describes one of three ways to migrate a Version 2.1 Configuration Manager to Version 6.0.

Before you start

Read about "Coexistence with previous versions and other products" on page 4.

An existing Windows Configuration Manager can be migrated to a Version 6.0 Configuration Manager on any of the supported operating systems via a JDBC Type 4 Universal DB2 connection.

To migrate a Configuration Manager to a different computer that has a JDBC Type 4 Universal DB2 connection installed, complete the following steps:

1. Install WebSphere Message Broker Version 6.0 on the computer to which you want to migrate your Version 2.1 Configuration Manager.
2. Add the following files to the environment in which you are going to create the new Configuration Manager. On z/OS, update the BIPCPROF file and submit the BIPGEN job. On all other operating systems, update the local environment.
   a. Add the following classes to the CLASSPATH:

      ```
      <db2 install>/jcc/classes/sqlj.zip
      <db2 install>/jcc/classes/db2jcc.jar
      <db2 install>/jcc/classes/db2jcc_javax.jar
      <db2 install>/jcc/classes/db2jcc_license_cisuz.jar
      ```
   b. Add to the computer-specific environment variable for libraries (for example LIBPATH on z/OS):

      <db2 install>/jcc/lib
   c. Add to the PATH:

      <db2 install>/jcc/bin

   where <db2 install> is where DB2 is installed at your location (for example /usr/lpp/db2710/db2710).
3. Stop the Version 2.1 Configuration Manager.
4. Create a Version 6.0 Configuration Manager on the second computer by running the **mqsicreateconfigmgr** command (or the BIPCRCM job on z/OS), specifying the data source, user name, and password required to access the Version 2.1 Configuration Manager database. For example:

   -u (userid)
   : The database userid on the Windows computer for the Configuration Manager

   -p (password)
   : The database password on the Windows computer for the Configuration Manager

   -n (database name)
   : *//server:port/database name*

   where:
   - *//server* is the IP address of the Windows computer (for example `//9.20.235.197`)
   - *:port* is the port number of DB2 on Windows
   - */database name* is the name of the Windows Configuration Manager database (for example `/MQSICMDB`)

   For example: `//9.20.235.197:50000/MQSICMDB`

   Use different queue manager names for the two Configuration Managers to maintain uniqueness in the WebSphere MQ network.

When you create the Version 6.0 Configuration Manager, domain information from the Version 2.1 Configuration Manager database is copied automatically to the Version 6.0 Configuration Manager internal repository. This might take a few minutes to migrate the database.

5. On the second computer, configure WebSphere MQ to allow the Version 6.0 Configuration Manager to communicate with the broker network. For example, you might need to configure channels, transmission queues, and remote queue manager definitions.
6. Start the Version 6.0 Configuration Manager.
7. In order to associate all brokers in the domain with the new Configuration Manager, so that the brokers publish their status messages to the correct queue manager, a complete topology deploy is required on the second computer. Deploy the topology using either the Message Brokers Toolkit or the command-line interface.

   When the topology deploy has completed re-subscribing to the brokers in the domain, the brokers are managed by the new Configuration Manager on the new system; do not use the previous Configuration Manager.
8. If necessary, uninstall DB2 and the Version 2.1 broker.

Before you migrate your brokers, or make any configuration changes, ensure that your Configuration Manager has been migrated correctly by performing the following checks:

1. Start the migrated Configuration Manager.
2. Start the Message Brokers Toolkit and connect it to the Configuration Manager's domain.
3. Check the list of domain components and see if any errors are produced.
4. Check the list of alerts for the domain and see if anything has stopped.
5. Check that you can open the Event Log.
6. Optional: If you have a broker running, stop and start a message flow, and check for a successful response; check that the Alerts view is updated when the message flow is stopped, and clear when the message flow is started.

If these steps are successful, your Configuration Manager has been migrated successfully. If they are unsuccessful, resolve any problems before you migrate further components or make configuration changes.

After you have migrated the Configuration Manager, migrate the User Name Server.

Migrating a Version 2.1 Configuration Manager to Version 6.0: to a different computer that does not have DB2 installed:

This topic describes one of three ways in which to migrate a Version 2.1 Configuration Manager to Version 6.0.

Before you start

Read about "Coexistence with previous versions and other products" on page 4.

To migrate a Configuration Manager to a different computer that does not have DB2 or a JDBC client for DB2 installed, complete the following steps:

1. Migrate the Version 2.1 Configuration Manager to Version 6.0 on the same computer. Complete the following tasks:

   a. Install WebSphere Message Broker Version 6.0 in a location other than where WebSphere MQ Integrator Broker Version 2.1 is installed.
   b. Stop the Version 2.1 Configuration Manager.
   c. Launch a Version 6.0 Command Console and issue the **mqsimigratecomponents** command to migrate your Version 2.1 Configuration Manager.
   d. Start the Version 6.0 Configuration Manager.

      When you start the Version 6.0 Configuration Manager for the first time, it automatically detects domain information in the DB2 database of the Version 2.1 Configuration Manager, and migrates it into the internal repository of the Version 6.0 Configuration Manager. No user intervention is required, and the DB2 database is not modified.
2. Stop the Version 6.0 Configuration Manager, and make a copy of its internal repository using the **mqsibackupconfigmgr** command.
3. Install WebSphere Message Broker Version 6.0 on the second computer.
4. On the second computer, create a Version 6.0 Configuration Manager by issuing the **mqsicreateconfigmgr** command (or the BIPCRCM job on z/OS).

   Use different queue manager names for the Version 2.1 Configuration Manager and Version 6.0 Configuration Managers to maintain uniqueness in the WebSphere MQ network.
5. On the second computer, configure WebSphere MQ to allow the Version 6.0 Configuration Manager to communicate with the broker network. For example, you might need to configure channels, transmission queues, and remote queue manager definitions.
6. On the second computer, issue the **mqsirestoreconfigmgr** command (or the BIPRSCM job on z/OS) to overwrite the contents of the empty Version 6.0 Configuration Manager repository with the repository that you backed up from the original computer.
7. On the second computer, start the Version 6.0 Configuration Manager.
8. In order to associate all brokers in the domain with the new Configuration Manager, so that the brokers publish their status messages to the correct queue manager, a complete topology deploy is required on the second computer. Deploy the topology using either the Message Brokers Toolkit or the command-line interface.

   When the topology deploy has completed re-subscribing to the brokers in the domain, the brokers are managed by the new Configuration Manager on the new system; do not use the previous Configuration Manager.

Before you migrate your brokers, or make any configuration changes, ensure that your Configuration Manager has been migrated correctly by performing the following checks:
1. Start the migrated Configuration Manager.
2. Start the Message Brokers Toolkit and connect it to the Configuration Manager's domain.
3. Check the list of domain components and see if any errors are produced.
4. Check the list of alerts for the domain and see if anything has stopped.
5. Check that you can open the Event Log.
6. Optional: If you have a broker running, stop and start a message flow, and check for a successful response; check that the Alerts view is updated when the message flow is stopped, and clear when the message flow is started.

If these steps are successful, your Configuration Manager has been migrated successfully. If they are unsuccessful, resolve any problems before you migrate further components or make configuration changes.

After you have migrated the Configuration Manager, migrate the User Name Server.

Migrating the User Name Server:

This topic describes the third step in migrating individual components from WebSphere MQ Integrator Broker Version 2.1 to WebSphere Message Broker Version 6.0.

To migrate the User Name Server from WebSphere MQ Integrator Broker Version 2.1 to WebSphere Message Broker Version 6.0, see the appropriate topic for your operating system:

- "Migrating a Version 2.1 User Name Server to Version 6.0 on distributed systems"
- "Migrating a Version 2.1 User Name Server to Version 6.0 on z/OS"

Migrating a Version 2.1 User Name Server to Version 6.0 on distributed systems:

Before you start:

Before you migrate the User Name Server, migrate the tooling and the Configuration Manager.

If you have a User Name Server in your Version 2.1 domain, you must migrate it to Version 6.0 when you migrate the Configuration Manager for that domain. This applies if the Configuration Manager and User Name Server are on the same computer, or different computers. This does not affect the migration of brokers, which can be migrated after the User Name Server and Configuration Manager have been migrated.

1. Stop the Version 2.1 User Name Server.
2. Install WebSphere Message Broker Version 6.0.
3. Issue the **mqsimigratecomponents** command for the User Name Server.
4. Start the Version 6.0 User Name Server.

When you have migrated all of your components, migrate the broker.

Migrating a Version 2.1 User Name Server to Version 6.0 on z/OS:

Before you start

- Ensure that you are familiar with the steps involved in creating a User Name Server on z/OS.
- The JCL uses the **mqsimigratecomponents** command to migrate a User Name Server on z/OS. This command takes many parameters, which you must understand fully before attempting to migrate the User Name Server.

If you have a User Name Server in your Version 2.1 domain, you must migrate it to Version 6.0 when you migrate the Configuration Manager for that domain. This applies if the Configuration Manager and User Name Server are on the same computer, or different computers. This does not affect the migration of brokers, which can be migrated after the User Name Server and Configuration Manager have been migrated.

1. Stop the Version 2.1 User Name Server.
2. Create a new User Name Server PDSE.
3. Copy all User Name Server JCL from the Version 6.0 installed SBIPPROC/SBIPSAMP PDSEs to the new User Name Server PDSE and customize them all. See Customizing the User Name Server JCL for more information.

   When you customize the new Version 6.0 JCL for this migration, do not make the ++*HOME*++ JCL variable the same value as the existing Version 2.1 component directory, otherwise the previous ENVFILE will be overwritten when you submit the BIPGEN job. Set the ++*HOME*++ variable to the Started Task user ID home directory (for example, /u/mq01uns).

   a. Customize the BIPEDIT file using values that are defined in the User Name Server's Version 2.1 mqsicompcif file. "Equivalent customization variables in mqsicompcif and BIPEDIT" on page 123 lists the customization variables that are used in the Version 2.1 mqsicompcif file, and the equivalent variables in the Version 6.0 BIPEDIT file.
   b. Copy any additional changes that you have made to the environment file, ENVFILE, to BIPUPROF in the component data set. Submit the BIPGEN job to create the environment file ENVFILE.
   c. Customize and submit the BIPMGCMP job.
4. Copy the started task JCL (BIPUNSP) to the procedures library. When you copy the started task, keep a second copy of the original in a safe place for backup purposes.
5. The verification program runs when you start the Version 6.0 User Name Server.

When you have migrated your User Name Server, migrate the broker.

Migrating the broker:

This topic describes the third step in migrating individual components from WebSphere MQ Integrator Broker Version 2.1 to WebSphere Message Broker Version 6.0.

Before you start

- Before you migrate the broker, migrate the tooling and migrate the Configuration Manager.
- On Linux and UNIX operating systems, ensure that you are using the updated database drivers for Oracle and Sybase. "Changing the 32-bit ODBC connection and XA resource manager definitions for a migrated broker" on page 29 tells you how to make the necessary changes.
- Back up the database tables of the Version 2.1 broker that you want to migrate.
- The broker must run on the same system and use the same queue manager as it did before migration. You do not need to change the configuration of the queue manager, but you might need to ensure that the WebSphere MQ product code is at the required release and service level to support WebSphere Message Broker Version 6.0. At the same time, you might want to ensure that you have installed the other software prerequisites.

When you migrate a broker domain from WebSphere MQ Integrator Broker Version 2.1 to WebSphere Message Broker Version 6.0, you can preserve some or all of your brokers at the Version 2.1 level of code. These brokers can participate in the Version 6.0 broker domain but, at any time subsequently, you can migrate the brokers to the Version 6.0 level of code, and you can do this in stages if you want. The following topics describe how to migrate a Version 2.1 broker to Version 6.0 when the Version 2.1 broker is already participating in a Version 6.0 broker domain.

When you migrate a broker from Version 2.1 to Version 6.0, retained publications are lost on z/OS. This is also the case when you migrate a Version 6.0 z/OS broker back to Version 2.1.

If two or more brokers share the same set of database tables, you must migrate all of these brokers at the same time. If you do need to migrate multiple brokers, on the same system or on different systems, adapt the instructions that follow.

The information in this topic assumes that you are migrating a WebSphere MQ Integrator Broker Version 2.1 broker. The same information applies, however, if you are migrating a WebSphere MQ Integrator Version 2.1 broker. Simply replace all references to "WebSphere MQ Integrator Broker" with "WebSphere MQ Integrator" in the instructions that follow.

If you are migrating a WebSphere MQ Integrator Version 2.1 broker that is participating in a WebSphere Message Broker with Rules and Formatter Extension Version 6.0 broker domain, replace all references to "WebSphere Message Broker" with "WebSphere Message Broker with Rules and Formatter Extension" in the instructions that follow. For instructions on how to migrate the New Era of Networks Rules and Formatter support for WebSphere MQ Integrator, see the New Era of Networks documentation.

The following topics tell you how to migrate the broker on distributed systems or on z/OS:
- "Migrating a Version 2.1 broker to Version 6.0 on distributed systems"
- "Migrating a Version 2.1 broker to Version 6.0 on z/OS" on page 32

Migrating a Version 2.1 broker to Version 6.0 on distributed systems:

Before you start

Before migrating a broker, ensure that you do not have any aggregations in progress. When you migrate a broker to Version 6.0 any live data being stored for aggregations in progress will be lost.

If the broker runs in a locale that is not listed in the Locales topic, check that the code page is one of the supported code pages and that the locale is set up correctly.

To migrate a Version 2.1 broker to Version 6.0 on distributed operating systems, complete the following steps:
1. Install WebSphere Message Broker Version 6.0 in a different location from Version 2.1.
2. Stop Version 2.1.
3. Update the .odbc.ini file to Version 6.0 format by following the instructions in "Updating your ODBC definitions when migrating" on page 29.

4. Issue the **mqsimigratecomponents** command for the broker from the Version 6.0 Command Console.
5. Start the Version 6.0 broker.
6. Optional: Uninstall the Version 2.1 product.
7. Repeat these steps for all remaining brokers.

Updating your ODBC definitions when migrating:

As part of migrating a broker, you must create ODBC definitions for broker and user databases, with database drivers that are appropriate for the new product version. You must do this before you run the **mqsimigratecomponents** command. The way that you do this depends on your operating system:

Windows

ODBC definitions are stored by the operating system. There is only one repository of ODBC data source names. You do not need to change DB2 and SQL Server data source definitions. Change Oracle and Sybase data source definitions according to the instructions in "Changing the 32-bit ODBC connection and XA resource manager definitions for a migrated broker."

Linux and UNIX

ODBC definitions are stored in a text file. For previous versions of the product, this file was located in /var/mqsi/odbc/.odbc.ini. When working with Version 6.0, create a new text file under this directory, but do not replace the existing file. This means that you do not affect existing brokers. Set the environment variable ODBCINI to point to the new file; perform this step as part of the setup for a Version 6.0 Command Console.

In your new Version 6.0 ODBC configuration file, ensure that you list all of the data sources that will be used by brokers that you want to migrate. For information on how to create a new ODBC.INI file, see Connecting to a database from Linux and UNIX systems. If you are using XA to enable coordinated transactions, you must also update your qm.ini file for WebSphere MQ, as described in "Changing the 32-bit ODBC connection and XA resource manager definitions for a migrated broker."

If you revert to a previous version of WebSphere Message Broker, you must reverse the changes that you make to the ODBC definitions.

- On UNIX, reset the ODBCINI environment variable to point to the previous version of the odbc.ini file.
- On Windows, use the Control Panel to adjust the ODBC settings.

Update the odbc.ini after you have run the **mqsimigratecomponents** command, but before you restart the broker.

Changing the 32-bit ODBC connection and XA resource manager definitions for a migrated broker:

WebSphere Message Broker includes new versions of the ODBC drivers that are supplied by DataDirect Technology (formerly Merant). These ODBC drivers are for Oracle and Sybase databases. A new version of the DataDirect Driver Manager for Linux and UNIX systems is also included. There are a number of changes that you need to make as a result of these new versions of the ODBC drivers for Oracle and Sybase databases.

- If you are migrating a broker from Version 2.1 or Version 5.0 to Version 6.0 on a Windows system, update the ODBC definition for each broker and the user data source name as described below. This applies to Oracle and Sybase databases only.
- If you are migrating a broker from Version 2.1 or Version 5.0 to Version 6.0 on UNIX, create a new ODBC configuration file that contains definitions for each of your existing data sources. Use the Version 6.0 template to do this, as described in Connecting to a database from Linux and UNIX systems. Do not change the existing definition file.
- You must also change the XA resource manager definition for the database if you have a message flow that is running in the broker and updates an Oracle or Sybase database within a global unit of work that is coordinated by the broker queue manager.
- If you are reverting a broker from Version 6.0 back to an earlier version, you must reverse these changes.

Make the required changes before you run the Version 6.0 **mqsimigratecomponents** command. If you are reverting to an earlier version, keep the new definitions until after you have finished running the **mqsimigratecomponents** command. The following sections describe the changes in detail.

Changing the 32-bit ODBC connection definitions:

Linux and UNIX systems:

Create a new ODBC definitions file by following the instructions in Connecting to a database from Linux and UNIX systems. Ensure that your ODBCINI environment variable points to the new file and not to the existing file before you run the Version 6.0 commands.

Windows systems:

To change the ODBC connection definitions on a Windows system:
1. Open the ODBC Data Source Administrator window.
2. Open the System DSN page.
3. For each Oracle and Sybase database that is accessed by the broker, associate the data source name with the new ODBC driver:
   a. Delete the data source by clicking **Remove**.
   b. Re-create the data source with the new ODBC driver by clicking **Add**.

The following table displays the name of the new ODBC driver for each database management system (DBMS):

DBMS	New ODBC driver
Oracle	MQSeries DataDirect Technologies 5.0 32-BIT Oracle
Sybase	MQSeries DataDirect Technologies 5.0 32-BIT Sybase Wire Protocol

Changing the XA resource manager definitions:

Linux and UNIX systems using 32-bit ODBC drivers and WebSphere MQ Version 5.3:

To change the XA resource manager definitions on a Linux or UNIX system, edit the queue manager configuration file (qm.ini) of the broker queue manager. This file is in the queue manager's directory.

In the XAResourceManager stanza for each Oracle and Sybase database that participates in a global unit of work, coordinated by the broker queue manager, change the entry for the switch file. The following tables specify what you need to change for each broker operating system and database management system (DBMS).

On AIX:

DBMS	Change ...	To ...
Oracle	`SwitchFile= /usr/opt/mqsi/merant/lib/UKor8dtc16.so` or `SwitchFile= /usr/opt/mqsi/merant/lib/UKor8dtc18.so`	`SwitchFile= <install_dir>/merant/lib/UKor8dtc20.so`
Sybase	`SwitchFile= /usr/opt/mqsi/merant/lib/UKase16.so` or `SwitchFile= /usr/opt/mqsi/merant/lib/UKase18.so`	`SwitchFile= <install_dir>/merant/lib/UKase20.so`

On Solaris:

DBMS	Change ...	To ...
Oracle	`SwitchFile= /opt/mqsi/merant/lib/UKor8dtc16.so` or `SwitchFile= /opt/mqsi/merant/lib/UKor8dtc18.so`	`SwitchFile= <install_dir>/merant/lib/UKor8dtc20.so`
Sybase	`SwitchFile= /opt/mqsi/merant/lib/UKase16.so` or `SwitchFile= /opt/mqsi/merant/lib/UKase18.so`	`SwitchFile= <install_dir>/merant/lib/UKase20.so`

On HP-UX:

DBMS	Change ...	To ...
Oracle	`SwitchFile= /opt/mqsi/merant/lib/libSwitchOracle.sl`	`SwitchFile= <install_dir>/merant/lib/UKor8dtc20.sl`
Sybase	`SwitchFile= /opt/mqsi/merant/lib/libSwitchSybase.sl`	`SwitchFile= <install_dir>/merant/lib/UKasedtc20.sl`

Windows systems:

To change the XA resource manager definitions on a Windows system:

1. Open the **Properties** window of the broker queue manager using the WebSphere MQ Services snap-in.
2. Open the **Resources** page.
3. For each Oracle and Sybase database that participates in a global unit of work, coordinated by the broker queue manager, change the contents of the **SwitchFile** field. The following table specifies what you need to change for each database management system (DBMS):

DBMS	Change ...	To ...
Oracle	*WMQIB*`\bin\UKor8dtc16.dll` [1] or *WMQIB*`\bin\UKor8dtc18.dll` [1]	*WBIMB*`\bin\UKor8dtc20.dll` [2]
Sybase	*WMQIB*`\bin\UKase16.dll`[1] or *WMQIB*`\bin\UKase18.dll`[1]	*WBIMB*`\bin\UKase20.dll`[2]

Notes:

1. *WMQIB* is the fully qualified path name of the directory where you originally installed WebSphere MQ Integrator Broker or WebSphere MQ Integrator.
2. *WBIMB* is the fully qualified path name of the directory where you have installed WebSphere Message Broker.

Migrating a Version 2.1 broker to Version 6.0 on z/OS:

Before you start

- Ensure that you are familiar with the steps that are involved in creating a broker on z/OS.
- The JCL uses the **mqsimigratecomponents** command to migrate a broker on z/OS. This command takes many parameters, which you must understand fully before attempting to migrate the broker.
- Before migrating a broker, ensure that you do not have any aggregations in progress. When you migrate a broker to Version 6.0, any live data being stored for aggregations in progress will be lost.
- If the broker runs in a locale that is not listed in the Locales topic, check that the code page is one of the supported code pages and that the locale is set up correctly.

When you migrate a broker from Version 2.1 to Version 6.0, retained publications are lost on z/OS. This is also the case when you migrate a Version 6.0 z/OS broker back to Version 2.1.

1. Stop the Version 2.1 broker.
2. Back up the broker database tables.
3. Create a new broker PDSE.
4. Copy all broker JCL from the Version 6.0 installed SBIPPROC and SBIPSAMP PDSEs to the new broker PDSE and customize them all.

   When you customize the new Version 6.0 JCL for this migration, do not make the ++*HOME*++ JCL variable the same value as the existing Version 2.1 component directory, otherwise the previous ENVFILE will be overwritten when you submit the BIPGEN job. Set the ++*HOME*++ variable to the Started Task user ID home directory (for example, /u/mq01brk).

   a. Customize the BIPEDIT file using values that are defined in the broker's Version 2.1 mqsicompcif file. "Equivalent customization variables in mqsicompcif and BIPEDIT" on page 123 lists the customization variables

   that are used in the Version 2.1 mqsicompcif file, and the equivalent variables in the Version 6.0 BIPEDIT file.
   b. Copy any additional changes that you have made to the environment file, ENVFILE, and the ODBC initialization file, `dsnaoini`, to BIPBPROF and BIPDSNAO in the component data set. Submit the BIPGEN job to create the environment file ENVFILE.
   c. Customize and submit the JCL BIPMGTB job. This creates LOB tablespaces. The broker database on z/OS introduced the use of LOB tablespaces in Version 5.0. They were not used in Version 2.1, so they need to be created at this point.
   d. Customize and submit the BIPMGCMP job. This migrates the registry, queues and broker database. As part of the database migration, database tables are created or deleted, and dropped, so you must you have the correct DB2 privileges. These privileges would be the same as those required for running the **mqsicreatebroker** command.
5. Copy the started task JCL (BIPBRKP) to the procedures library. When you copy the started task, keep a second copy of the original in a safe place for backup purposes.
6. The verification program runs when you start the Version 6.0 broker.

Migrating from Version 2.1 to Version 6.0: Version 6.0 coexisting temporarily with Version 2.1

This topic describes one of two migration scenarios that are available to you when migrating from WebSphere MQ Integrator Broker Version 2.1 to WebSphere Message Broker Version 6.0.

Before you start:
- Back up your Version 2.1 resources
- Read the following topics:
  – "Planning for migration from Version 2.1" on page 15
  – "Coexistence with previous versions and other products" on page 4

In this migration scenario, all of your Version 2.1 components are on the same computer. You want to keep Version 2.1 installed temporarily during and after migration to Version 6.0. You can then stop and uninstall Version 2.1 at any time after migration has been completed.

1. Use the Control Center to export all Version 2.1 message flows, message sets, and user-defined nodes. See the *WebSphere MQ Integrator Broker Version 2.1 Administration Guide* for more information.
2. Install WebSphere Message Broker Version 6.0 in a different location from where WebSphere MQ Integrator Broker Version 2.1 is installed.
3. Create a Version 6.0 User Name Server using the **mqsicreateusernameserver** command.
4. Start the Version 6.0 User Name Server.
5. Create a Version 6.0 Configuration Manager, using the **mqsicreateconfigmgr** command, specifying the Version 2.1 database but the Version 6.0 User Name Server.
6. Start the Version 6.0 Configuration Manager.
7. Create a Version 6.0 broker using the **mqsicreatebroker** command.
8. Start the Version 6.0 broker.

9. Import all Version 2.1 message flows, message sets, and user-defined extensions using the **mqsimigratemsgflows** and **mqsimigratemsgsets** commands.
10. Stop the Version 2.1 broker immediately after deployment.

When you no longer need to run Version 2.1, stop and uninstall any Version 2.1 components.

After you have completed the migration, see the post-migration tasks for information about tasks that you might want to perform after migration.

Migrating a message flow

Before you start:

Read the concept topic about conditions for a Version 5.0 broker participating in a Version 6.0 broker domain.

You can migrate the message flows that you have created in your Version 2.1 product (WebSphere MQ Event Broker, WebSphere MQ Integrator Broker, or WebSphere MQ Integrator) and use them in WebSphere Message Broker Version 6.0.

If you are migrating from WebSphere MQ Event Broker Version 2.1, all information in this topic that refers to user-defined plug-ins and ESQL does not apply: these facilities are not available on WebSphere MQ Event Broker Version 2.1.

If you have migrated from WebSphere MQ Integrator Broker Version 2.1, you might have written message flows that handle XML messages that use XML namespaces. In Version 2.1, such XML messages are parsed in a different way to that used by WebSphere Message Broker Version 6.0. While such message flows will continue to work correctly when hosted by Version 6.0, it is better to upgrade them to become namespace aware by following the steps in “Making a message flow namespace aware” on page 37.

You might want to change the message flows that you migrate to take advantage of the new nodes and features that are available in Version 6.0 . For example, you might want to replace a user-defined node that receives web services requests with the built-in HTTPInput node.

For more information about changes in this release, see What’s new in Version 6.0?.

You can migrate more than one message flow at one time if you want them to be defined in the same message flow project. You must migrate subflows and user-defined nodes with the message flows in which they are included to ensure consistent references.

If you have defined more than one message flow with the same name, or the message flow has been exported into more than one export file, the migration task overwrites any existing message flow with the next flow that it finds of the same name without warning. Therefore be careful to avoid conflicts, and to ensure that the most recent version of a multiply-defined message flow is the last one to be migrated.

If you have multiple versions of the same message flow, which you use as a subflow in other flows in the same migration directory, the results of the import are unpredictable.

To migrate a message flow:

1. **Before you uninstall Version 2.1**, export the message flow or flows from the Control Center using the Version 2.1 tools (see the Version 2.1 documentation for detailed information).

   The migration process is most efficient when all referenced subflows are included in the same export file, therefore export all message flows that you want to migrate to a single message flow project into a single export file.
2. Transfer the export file or files to the new system on which you are running the workbench. Check that the directory in which you store these files does not contain any other files. Store the files that you intend to import into a single message flow project in a separate directory and migrate each directory separately. Make sure that you do not store any files in subdirectories of the project directory, because these files are ignored by the migrate command.
3. If you have a workbench session active, close it. You cannot run the migrate command if the workbench is running.
4. At a command prompt, invoke the **mqsimigratemsgflows** command, specifying the new project name and the directory in which you stored the export files. When the command has completed:
   - The message flows that are contained in the export files in the specified directory are imported into the specified message flow project. If the project already exists, the additional message flows are included with current content, if any. If the project does not exist before you invoke the command, it is created for you. It is better to allow the command to create the message flow project for you.
   - Message flows and subflows are created and their definitions are stored in files named *flow_name*.msgflow. User-defined nodes are created and their definitions are stored in files named *node_name*.msgnode.

     If you want to rename any of these message flows or nodes after import to conform to your local naming conventions, use the facilities provided by the workbench to preserve consistency and integrity of all references. Do not rename any files within the file system.
   - If any of the nodes in the message flows contain ESQL, this is extracted from the node itself and stored in the ESQL file *message_flow_name*.esql. The ESQL for each node is wrapped between the appropriate CREATE and END MODULE statements (for Compute, Database, or Filter). The command creates the ESQL file if it does not already exist.

     When you add a message flow to a bar file, runtime ESQL code is generated at Version 6.0 level. This code is incompatible with Version 2.1 brokers. If you want the bar file to include version 2.0 runtime ESQL, check the **Compile ESQL for brokers version 2.1** box. If you do so, you cannot include Version 6.0 enhancements in the ESQL code, but you can deploy the flows to both Version 2.1 and Version 6.0 brokers.

     For more information, see the Adding files to a broker archive topic.
5. Check the report file mqsimigratemsgflows.report.txt that is written to the directory from which you invoked the command. The command provides the following information:
   - The name of each message flow, subflow, and user-defined node that has been migrated. If any of these resources had a name that is incompatible with Version 6.0, the command updates the name and all references to that

name to ensure consistency. (If you migrate a resource with an invalid name more than once, the correction that is made to the name is always the same.)

- The success or failure of each resource that is migrated
- An indication of a subflow that cannot be located (its definition is not contained in any of the export files, but it is included in one or more of the migrated message flows). If this occurs, find the missing subflow and import it into the appropriate project. If you cannot retrieve the missing subflow for any reason, re-create it with the original name. All affected flows can then link correctly to the new subflow.

  You do not need to repeat the whole export and import process.
- An indication that a resource that was migrated as a message flow and stored in a .msgflow file might be a user-defined node. If this warning occurs, check if the specified resource is a user-defined node or a message flow. If it is a message flow, it has been migrated correctly. If it is a user-defined node, complete the actions outlined in step 11 on page 37.

6. Start the workbench and switch to the Broker Application Development perspective.
7. Open the message flow project that was created or updated by the migrate command by right-clicking the project and clicking **Open Project**.

   If the project is already open, right-click and click **Refresh**, then **Rebuild Project** to ensure that the Navigator view reflects the new content. Rebuild also performs a validation of the message flow project contents.

   Because ESQL and mappings are handled in a different way in Version 6.0, the migration process replaces some of the Version 2.1 nodes with different Version 6.0 nodes. The following table lists the replaced nodes. The ESQL that is associated with each node is created as a module with a default name, and the node property is set to the name of that module.

Version 2.1 node	Version 6.0 node
Compute	Compute
Filter	Filter
Database	Database
DataDelete	Database
DataInsert	Database
DataUpdate	Database
Extract	Compute
Warehouse	Database

8. If a message flow includes one or more Filter nodes, check the ESQL module for each node in the ESQL file to ensure that the RETURN statement correctly returns a valid expression that resolves to a Boolean value.
9. If a message flow includes a node with ESQL, the ESQL references fields in a message derived from an imported C header, and you have re-created the message model by importing the C header into the workbench, check the ESQL statements that refer to this message. The import into the Version 6.0 workbench might create a model with different naming conventions than that created by the Version 2.1 importer, and you might need to update one or more field references.
10. If you had promoted the ESQL property of any of the Version 2.1 nodes that included ESQL customization to reuse ESQL in multiple nodes, this is not

maintained in the migrated Version 6.0 message flows because promotion of ESQL related properties is no longer supported. The Tasks view shows an error for each ESQL promoted property. To achieve the same effect, create an ESQL function and call that function from each node's ESQL module.

11. If you have migrated a user-defined node, only the XML interface definition file is migrated into a node .msgnode file (this defines only the terminals and properties of the node). Complete its migration and its definition manually in Version 6.0 of the product. The following steps provide an outline of the required process: for full details, see Creating the user interface representation of a user-defined node in the workbench.
    a. Create a user-defined node project and move the .msgnode file from the message flow project into the new user-defined node project. When you do this, the associated properties files are created.
    b. Optional: Complete development of the user-defined node in the Eclipse environment to create the user-defined node Eclipse plug-in (the directory structure that contains the files that make up this node). This task includes creating node resources for help, icons, and property editors and compilers, if required.
    c. Check for errors in the task list. These might be generated, for example, if the node or its terminal names include the space character (not supported in Version 6.0), or if a flow embeds another migrated flow and the reference is not correct. Resolve these errors by correcting names, or by using the **Locate subflow** menu option.
    d. Install the runtime code for the node (the .lil file) on the appropriate broker systems. You do not need to recompile the code for your user-defined node when you migrate it.
    e. Stop and restart the broker to recognize the new or changed files.

When you have migrated your resources, see the Version 2.1 post-migration tasks for information about tasks that you might want to perform after migration.

Making a message flow namespace aware:

This topic describes how to make a message flow namespace aware.

If you have migrated from WebSphere MQ Integrator Broker Version 2.1, you might have written message flows that handle XML messages that use XML namespaces. In Version 2.1, such XML messages are parsed in a different way to that used by WebSphere Message Broker. While such message flows will continue to work correctly when hosted by WebSphere Message Broker, it is better to upgrade them to become namespace aware by performing the following steps.

1. Correct the message model.

   If you are using the MRM domain, you will have modeled the XML message in question. Create a new message set, ensuring that the *Use namespaces* property is set and that you create an XML physical format with the same name as the original. There are two cases to consider:
   - The XML message is described by an XML Schema. Import the XML Schema into the new message set. This creates a new namespace-aware message definition file automatically.
   - The XML message is modeled by hand. Create a new message definition file in the desired target namespace (this is specified on the last page of the wizard), then re-create your message model using the editor.

   - Do not set the *XML Name* property of an element, as you did in Version 2.1. It should be left to take the default value.
   - Do not re-create any elements or attributes with names commencing "xsi_" or "xmlns_"; these are all handled automatically by the parser.
   - Re-create XML attributes as attributes in the model (instead of elements with an *XML Render* property set to `XMLAttribute`).
2. Correct the message flow.
   - If you were using the MRM domain, change any references to the message set to input nodes or compute nodes.
   - If you were using the XML domain, change this to either the XMLNS or XMLNSC domain in input nodes and ESQL statements. These domains are namespace aware; the original XML domain is not and is effectively deprecated.
   - Change ESQL paths that refer to elements in the message to use the correct namespace-aware syntax. The ESQL editor content assist feature can help with the automatic creation of namespace constants if a message model exists.
3. Deploy the corrected message flow and the new message set.

   Add the corrected resources to a new broker archive file and deploy it to the target broker execution group. If the XML messages that are received by the message flow contain an MQRFH2 header that specifies the message set, the sending application must be changed in step.

You have upgraded your message flows to become namespace aware.

Migrating a message set

This task topic describes how to migrate a message set into WebSphere Message Broker Version 6.0.

Before you start:

Before you attempt this task, read the following information:
- mqsimigratemsgsets command

Note: This task topic only applies to migrating a message set from WebSphere MQ Integrator Version 2.1 or WebSphere MQ Integrator Broker Version 2.1 to WebSphere Message Broker Version 6.0.

It is possible to use message sets in WebSphere Message Broker Version 6.0 that were created using WebSphere MQ Integrator Version 2.1 or WebSphere MQ Integrator Broker Version 2.1. If the `*.mrp` file is modified by hand between the export from WebSphere MQ Integrator Version 2.1 or WebSphere MQ Integrator Broker Version 2.1, and then imported into WebSphere Message Broker Version 6.0, the results are not guaranteed and errors could occur.

To migrate a message set:
1. Export the message set or sets from WebSphere MQ Integrator Broker Version 2.1. For more detailed information on this task refer to the WebSphere MQ Integrator Broker Version 2.1 documentation library.
2. Transfer the exported `*.mrp` file or files to the system on which you are running the WebSphere Message Broker Version 6.0 workbench. Check that the directory in which you store these files contains only the files that you want to import.

3. Close the workbench. This must not be running when you invoke the **mqsimigratemsgsets** command.
4. From a command line, invoke the **mqsimigratemsgsets** command, specifying the parameters that you require.
5. Check the report file `mqsimigratemsgsets.report.txt` that is written to the directory from which you invoked the command. The command provides the following information:
   - The parameters specified when the mqsimigratemsgsets command was invoked
   - The name of each message set file (`*.mrp` file) that has been migrated
   - Any warnings that have been generated
   - Details of the objects created
   - The time taken to process the message set
   - The number of warnings produced for the message set
   - The number of objects created for the message set
   - The number of exported files that were processed
6. Start the workbench and switch to the Broker Application Development perspective.
7. You can now examine and manipulate the newly created message set project or projects. The created projects are closed in the Broker Application Development perspective and must be opened (right-click the project then click **Open Project**).

When you have migrated your resources, see the Version 2.1 post-migration tasks or Version 5.0 post-migration tasks for information about tasks that you might want to perform after migration.

Migrating from WebSphere MQ Event Broker Version 2.1 to WebSphere Message Broker Version 6.0

Before you start

Complete the following task:
- "Preparing to migrate from WebSphere MQ Event Broker Version 2.1" on page 41

and find out more about "Coexistence with previous versions and other products" on page 4.

To migrate a broker domain from WebSphere MQ Event Broker Version 2.1 to WebSphere Message Broker Version 6.0:
1. Close all Control Center sessions.
2. On each system where one or more brokers are running, perform the following steps for each broker:
   a. Stop the broker by issuing the **wmqpsstop** command with the name of the broker.
   b. Delete the broker by issuing the **wmqpsdeletebroker** command with the name of the broker.
3. Drop the database tables that were used by the brokers that you have just deleted. If a set of broker database tables are in a database by themselves and you do not intend to use the database after migration, drop the whole database.

The broker database tables are listed in the *WebSphere MQ Event Broker Version 2.1 Administration Guide*.

4. On the system where the Configuration Manager is running:
   a. Stop the Configuration Manager by issuing the **wmqpsstop** command for the component ConfigMgr.
   b. Delete the Configuration Manager by issuing the **wmqpsdeleteconfigmgr** command with the **-n** parameter. Using this parameter causes the configuration repository to be deleted.
5. On a system where a User Name Server is running:
   a. Stop the User Name Server by issuing the **wmqpsstop** command for the component UserNameServer.
   b. Delete the User Name Server by issuing the **wmqpsdeleteusernameserver** command.
6. On each system, uninstall WebSphere MQ Event Broker. Follow the instructions in the appropriate book for your operating system:
   - *WebSphere MQ Event Broker for AIX Version 2.1 Installation Guide*
   - *WebSphere MQ Event Broker for HP-UX Version 2.1 Installation Guide*
   - *WebSphere MQ Event Broker for Solaris Version 2.1 Installation Guide*
   - *WebSphere MQ Event Broker for Windows NT and Windows 2000 Version 2.1 Installation Guide*

   Windows On a Windows system, select the option to uninstall including data.
7. On each system, install the components of WebSphere Message Broker that you require. For detailed instructions on how to perform this task, see the Installation Guide.
8. Re-create your broker domain. As part of this task, perform the following steps, using the information that you recorded previously, where appropriate:
   a. Import the message flows that you want to migrate by issuing the **mqsimigratemsgflows** command.
   b. Using WebSphere Message Broker commands, re-create and start the Configuration Manager, a User Name Server, and the brokers that you want to migrate.
   c. Using the workbench, add each broker to the broker domain and re-create the execution groups that were originally within the broker. Re-create only the execution groups that you want to preserve.
   d. Using the workbench, enter and deploy the assignments, topology, and topics configuration data that you want to preserve.

      Deploy migrated message flows to a test environment first. When you are sure that they are working correctly, deploy them to a production environment.

The migration is now complete and the broker domain is ready for use. Delete any queue managers that you no longer need.

Note that the migration does not preserve the subscriptions and retained publications at the brokers. To re-create this information after migration, subscribers must renew their subscriptions and publishers must republish.

Preparing to migrate from WebSphere MQ Event Broker Version 2.1

This topic describes what to do before you start to migrate a WebSphere MQ Event Broker Version 2.1 broker domain in any of the following tasks:

- "Migrating from WebSphere MQ Event Broker Version 2.1 to WebSphere Message Broker Version 6.0" on page 39
- "Migrating from WebSphere MQ Event Broker Version 2.1 and WebSphere MQ Integrator Broker Version 2.1 to WebSphere Message Broker Version 6.0" on page 42

1. Make sure that Control Center users have checked in all WebSphere MQ Event Broker resources.
2. Back up all configuration repository and broker database tables.
3. Decide which message flows you want to migrate and use in the WebSphere Message Broker Version 6.0 broker domain. Using a Control Center session, in which all the required message flows are visible in the workspace, export the message flows. Save the export files in a directory other than the one in which WebSphere MQ Event Broker is installed.
4. Decide which brokers you want to migrate, and which assignments, topology, and topics configuration data you want to preserve.

   For each broker that you want to migrate, and for the associated assignments configuration data that you want to preserve, record the following information (view this information in a Control Center session and record it manually):

   - The name of the broker
   - The name of each execution group within the broker
   - For each execution group within the broker, the name of each message flow that is assigned to the execution group
   - For each message flow assigned to an execution group, the following properties:
     - Additional instances
     - Commit count
     - Commit interval
     - Coordinated transaction

   For the topology configuration data that you want to preserve, record the following information:

   - The name of each collective and the names of the brokers that are in the collective
   - All direct connections between brokers

   For the topics configuration data that you want to preserve, record the following information:

   - The hierarchy of topics, including the name of each topic
   - The access control list for each topic
5. Decide which databases and queue managers you want to preserve.

   When you migrate, you delete all the components of WebSphere MQ Event Broker and drop all the database tables used by the Configuration Manager and the brokers. However, to simplify the task of configuring the new broker domain, preserve the databases and queue managers used by the brokers that you want to migrate. If you are migrating a single broker domain from WebSphere MQ Event Broker to WebSphere Message Broker, also preserve the database and queue manager used by the Configuration Manager, and the queue manager used by a User Name Server.

You do not need to change the configuration of a queue manager that is preserved during migration. You might, however, need to ensure that the WebSphere MQ product code is at the required release and service level to support WebSphere Message Broker. At the same time, ensure that you have installed the other software prerequisites.

6. Decide where you are going to store the development data that is created and maintained in the workbench. Store the data in the local file system, on a shared drive, or in a shared repository that is supported by Eclipse. The instructions for the individual migration tasks assume that you are using the local file system or a shared drive.
7. When you are ready to migrate, run the **mqsimigratecomponents** command with the **-c** parameter. This performs a pre-migration check against the Version 2.1 components to ensure that they can be migrated. The pre-migration check identifies potential problems and allows you to correct them before proceeding with migration.

When you have run the pre-migration check successfully, you are ready to perform migration. Go to the appropriate topic for your product:

- Migrating from WebSphere MQ Event Broker Version 2.1 and WebSphere MQ Integrator Broker Version 2.1
- Migrating from WebSphere MQ Event Broker Version 2.1 and WebSphere MQ Integrator Version 2.1
- Migrating from WebSphere MQ Event Broker Version 2.1 and WebSphere MQ Integrator Version 2.1 to WebSphere Message Broker with Rules and Formatter Extension Version 6.0

Migrating from WebSphere MQ Event Broker Version 2.1 and WebSphere MQ Integrator Broker Version 2.1 to WebSphere Message Broker Version 6.0

Before you start

Complete the task:

- "Preparing to migrate from WebSphere MQ Event Broker Version 2.1" on page 41

and find out more about "Coexistence with previous versions and other products" on page 4.

WebSphere MQ Event Broker and WebSphere MQ Integrator Broker might be installed on the same system, or there might be no system in the two domains on which both products are installed. Use the procedure that follows in either case.

To migrate a WebSphere MQ Event Broker Version 2.1 broker domain and a WebSphere MQ Integrator Broker Version 2.1 broker domain into a single WebSphere Message Broker Version 6.0 broker domain, perform the following steps:

1. Close all Control Center sessions for the WebSphere MQ Event Broker broker domain.
2. On each system where one or more WebSphere MQ Event Broker brokers are running, perform the following steps for each broker:

    a. Stop the broker by issuing the **wmqpsstop** command with the name of the broker.
    b. Delete the broker by issuing the **wmqpsdeletebroker** command with the name of the broker.
3. Drop the database tables that were used by the brokers that you have just deleted. If a set of broker database tables are in a database by themselves and you do not intend to use the database after migration, drop the whole database.

   The broker database tables are listed in the *WebSphere MQ Event Broker Version 2.1 Administration Guide*.
4. On the system where the WebSphere MQ Event Broker Configuration Manager is running:
    a. Stop the Configuration Manager by issuing the **wmqpsstop** command for the component ConfigMgr.
    b. Delete the Configuration Manager by issuing the **wmqpsdeleteconfigmgr** command with the **-n** parameter. Using this parameter causes the configuration repository to be deleted.
5. On a system where a WebSphere MQ Event Broker User Name Server is running:
    a. Stop the User Name Server by issuing the **wmqpsstop** command for the component UserNameServer.
    b. Delete the User Name Server by issuing the **wmqpsdeleteusernameserver** command.
6. On each system where WebSphere MQ Event Broker is installed, uninstall WebSphere MQ Event Broker by following the instructions in the appropriate book for your operating system:
    - *WebSphere MQ Event Broker for AIX Version 2.1 Installation Guide*
    - *WebSphere MQ Event Broker for HP-UX Version 2.1 Installation Guide*
    - *WebSphere MQ Event Broker for Solaris Version 2.1 Installation Guide*
    - *WebSphere MQ Event Broker for Windows NT and Windows 2000 Version 2.1 Installation Guide*

   Windows On a Windows system, select the option to uninstall including data.
7. Migrate the WebSphere MQ Integrator Broker broker domain to a WebSphere Message Broker broker domain by following the instructions in “Migrating from WebSphere MQ Integrator Broker Version 2.1 to WebSphere Message Broker Version 6.0” on page 16.

   At the point where you import the message flows that you exported from the WebSphere MQ Integrator Broker broker domain, import the message flows that you exported from the WebSphere MQ Event Broker broker domain.
8. Within the WebSphere Message Broker broker domain, re-create those aspects of the WebSphere MQ Event Broker broker domain that you want to migrate. Use the information you recorded previously, where appropriate.
    a. Using WebSphere Message Broker commands, re-create and start the brokers that you want to migrate.
    b. Using the workbench, add each broker to the broker domain and re-create the execution groups that were originally within the broker. Re-create only the execution groups that you want to preserve.
    c. Using the workbench, enter and deploy the assignments, topology, and topics configuration data that you want to preserve.

Deploy migrated message flows to a test environment first. When you are sure that they are working correctly, deploy them to a production environment.

The migration is now complete and the broker domain is ready for use. Drop any databases and delete any queue managers that you no longer need.

Note that the migration does not preserve the subscriptions and retained publications at the WebSphere MQ Event Broker brokers. To re-create this information after migration, subscribers must renew their subscriptions and publishers must republish.

Migrating from WebSphere MQ Event Broker Version 2.1 and WebSphere MQ Integrator Version 2.1 to WebSphere Message Broker Version 6.0

If you have used the New Era of Networks Rules and Formatter support in WebSphere MQ Integrator Version 2.1 and want to continue using it after migration, migrate to WebSphere Message Broker with Rules and Formatter Extension Version 6.0 by following the instructions in "Migrating from WebSphere MQ Event Broker Version 2.1 and WebSphere MQ Integrator Version 2.1 to WebSphere Message Broker with Rules and Formatter Extension Version 6.0."

To migrate a WebSphere MQ Event Broker Version 2.1 broker domain and a WebSphere MQ Integrator Version 2.1 broker domain into a single WebSphere Message Broker Version 6.0 broker domain, follow the instructions in "Migrating from WebSphere MQ Event Broker Version 2.1 and WebSphere MQ Integrator Broker Version 2.1 to WebSphere Message Broker Version 6.0" on page 42, replacing all references to "WebSphere MQ Integrator Broker" with "WebSphere MQ Integrator".

Migrating from WebSphere MQ Event Broker Version 2.1 and WebSphere MQ Integrator Version 2.1 to WebSphere Message Broker with Rules and Formatter Extension Version 6.0

To migrate a WebSphere MQ Event Broker Version 2.1 broker domain and a WebSphere MQ Integrator Version 2.1 broker domain into a single WebSphere Message Broker with Rules and Formatter Extension Version 6.0 broker domain, follow the instructions in "Migrating from WebSphere MQ Event Broker Version 2.1 and WebSphere MQ Integrator Broker Version 2.1 to WebSphere Message Broker Version 6.0" on page 42, replacing all references to "WebSphere MQ Integrator Broker" with "WebSphere MQ Integrator" and all references to "WebSphere Message Broker" with "WebSphere Message Broker with Rules and Formatter Extension".

For instructions on how to migrate the New Era of Networks Rules and Formatter support for WebSphere MQ Integrator, see the New Era of Networks documentation.

Migrating a message flow

Before you start:

Read the concept topic about conditions for a Version 5.0 broker participating in a Version 6.0 broker domain.

You can migrate the message flows that you have created in your Version 2.1 product (WebSphere MQ Event Broker, WebSphere MQ Integrator Broker, or WebSphere MQ Integrator) and use them in WebSphere Message Broker Version 6.0.

If you are migrating from WebSphere MQ Event Broker Version 2.1, all information in this topic that refers to user-defined plug-ins and ESQL does not apply: these facilities are not available on WebSphere MQ Event Broker Version 2.1.

If you have migrated from WebSphere MQ Integrator Broker Version 2.1, you might have written message flows that handle XML messages that use XML namespaces. In Version 2.1, such XML messages are parsed in a different way to that used by WebSphere Message Broker Version 6.0. While such message flows will continue to work correctly when hosted by Version 6.0, it is better to upgrade them to become namespace aware by following the steps in "Making a message flow namespace aware" on page 37.

You might want to change the message flows that you migrate to take advantage of the new nodes and features that are available in Version 6.0 . For example, you might want to replace a user-defined node that receives web services requests with the built-in HTTPInput node.

For more information about changes in this release, see What's new in Version 6.0?.

You can migrate more than one message flow at one time if you want them to be defined in the same message flow project. You must migrate subflows and user-defined nodes with the message flows in which they are included to ensure consistent references.

If you have defined more than one message flow with the same name, or the message flow has been exported into more than one export file, the migration task overwrites any existing message flow with the next flow that it finds of the same name without warning. Therefore be careful to avoid conflicts, and to ensure that the most recent version of a multiply-defined message flow is the last one to be migrated.

If you have multiple versions of the same message flow, which you use as a subflow in other flows in the same migration directory, the results of the import are unpredictable.

To migrate a message flow:

1. **Before you uninstall Version 2.1**, export the message flow or flows from the Control Center using the Version 2.1 tools (see the Version 2.1 documentation for detailed information).

   The migration process is most efficient when all referenced subflows are included in the same export file, therefore export all message flows that you want to migrate to a single message flow project into a single export file.
2. Transfer the export file or files to the new system on which you are running the workbench. Check that the directory in which you store these files does not contain any other files. Store the files that you intend to import into a single message flow project in a separate directory and migrate each directory separately. Make sure that you do not store any files in subdirectories of the project directory, because these files are ignored by the migrate command.
3. If you have a workbench session active, close it. You cannot run the migrate command if the workbench is running.

4. At a command prompt, invoke the **mqsimigratemsgflows** command, specifying the new project name and the directory in which you stored the export files. When the command has completed:
   - The message flows that are contained in the export files in the specified directory are imported into the specified message flow project. If the project already exists, the additional message flows are included with current content, if any. If the project does not exist before you invoke the command, it is created for you. It is better to allow the command to create the message flow project for you.
   - Message flows and subflows are created and their definitions are stored in files named *flow_name*.msgflow. User-defined nodes are created and their definitions are stored in files named *node_name*.msgnode.

     If you want to rename any of these message flows or nodes after import to conform to your local naming conventions, use the facilities provided by the workbench to preserve consistency and integrity of all references. Do not rename any files within the file system.
   - If any of the nodes in the message flows contain ESQL, this is extracted from the node itself and stored in the ESQL file *message_flow_name*.esql. The ESQL for each node is wrapped between the appropriate CREATE and END MODULE statements (for Compute, Database, or Filter). The command creates the ESQL file if it does not already exist.

     When you add a message flow to a bar file, runtime ESQL code is generated at Version 6.0 level. This code is incompatible with Version 2.1 brokers. If you want the bar file to include version 2.0 runtime ESQL, check the **Compile ESQL for brokers version 2.1** box. If you do so, you cannot include Version 6.0 enhancements in the ESQL code, but you can deploy the flows to both Version 2.1 and Version 6.0 brokers.

     For more information, see the Adding files to a broker archive topic.
5. Check the report file mqsimigratemsgflows.report.txt that is written to the directory from which you invoked the command. The command provides the following information:
   - The name of each message flow, subflow, and user-defined node that has been migrated. If any of these resources had a name that is incompatible with Version 6.0, the command updates the name and all references to that name to ensure consistency. (If you migrate a resource with an invalid name more than once, the correction that is made to the name is always the same.)
   - The success or failure of each resource that is migrated
   - An indication of a subflow that cannot be located (its definition is not contained in any of the export files, but it is included in one or more of the migrated message flows). If this occurs, find the missing subflow and import it into the appropriate project. If you cannot retrieve the missing subflow for any reason, re-create it with the original name. All affected flows can then link correctly to the new subflow.

     You do not need to repeat the whole export and import process.
   - An indication that a resource that was migrated as a message flow and stored in a .msgflow file might be a user-defined node. If this warning occurs, check if the specified resource is a user-defined node or a message flow. If it is a message flow, it has been migrated correctly. If it is a user-defined node, complete the actions outlined in step 11 on page 37.
6. Start the workbench and switch to the Broker Application Development perspective.

7. Open the message flow project that was created or updated by the migrate command by right-clicking the project and clicking **Open Project**.

   If the project is already open, right-click and click **Refresh**, then **Rebuild Project** to ensure that the Navigator view reflects the new content. Rebuild also performs a validation of the message flow project contents.

   Because ESQL and mappings are handled in a different way in Version 6.0, the migration process replaces some of the Version 2.1 nodes with different Version 6.0 nodes. The following table lists the replaced nodes. The ESQL that is associated with each node is created as a module with a default name, and the node property is set to the name of that module.

Version 2.1 node	Version 6.0 node
Compute	Compute
Filter	Filter
Database	Database
DataDelete	Database
DataInsert	Database
DataUpdate	Database
Extract	Compute
Warehouse	Database

8. If a message flow includes one or more Filter nodes, check the ESQL module for each node in the ESQL file to ensure that the RETURN statement correctly returns a valid expression that resolves to a Boolean value.
9. If a message flow includes a node with ESQL, the ESQL references fields in a message derived from an imported C header, and you have re-created the message model by importing the C header into the workbench, check the ESQL statements that refer to this message. The import into the Version 6.0 workbench might create a model with different naming conventions than that created by the Version 2.1 importer, and you might need to update one or more field references.
10. If you had promoted the ESQL property of any of the Version 2.1 nodes that included ESQL customization to reuse ESQL in multiple nodes, this is not maintained in the migrated Version 6.0 message flows because promotion of ESQL related properties is no longer supported. The Tasks view shows an error for each ESQL promoted property. To achieve the same effect, create an ESQL function and call that function from each node's ESQL module.
11. If you have migrated a user-defined node, only the XML interface definition file is migrated into a node .msgnode file (this defines only the terminals and properties of the node). Complete its migration and its definition manually in Version 6.0 of the product. The following steps provide an outline of the required process: for full details, see Creating the user interface representation of a user-defined node in the workbench.
    a. Create a user-defined node project and move the .msgnode file from the message flow project into the new user-defined node project. When you do this, the associated properties files are created.
    b. Optional: Complete development of the user-defined node in the Eclipse environment to create the user-defined node Eclipse plug-in (the directory structure that contains the files that make up this node). This task includes creating node resources for help, icons, and property editors and compilers, if required.

c. Check for errors in the task list. These might be generated, for example, if the node or its terminal names include the space character (not supported in Version 6.0), or if a flow embeds another migrated flow and the reference is not correct. Resolve these errors by correcting names, or by using the **Locate subflow** menu option.
d. Install the runtime code for the node (the .lil file) on the appropriate broker systems. You do not need to recompile the code for your user-defined node when you migrate it.
e. Stop and restart the broker to recognize the new or changed files.

When you have migrated your resources, see the Version 2.1 post-migration tasks for information about tasks that you might want to perform after migration.

Migrating from WebSphere MQ Integrator Version 2.1 to WebSphere Message Broker Version 6.0

If you have used the New Era of Networks Rules and Formatter support in WebSphere MQ Integrator Version 2.1 and want to continue using it after migration, migrate to WebSphere Message Broker with Rules and Formatter Extension Version 6.0 by following the instructions in “Migrating from WebSphere MQ Integrator Version 2.1 to WebSphere Message Broker with Rules and Formatter Extension Version 6.0.”

To migrate a broker domain from WebSphere MQ Integrator Version 2.1 to WebSphere Message Broker Version 6.0, follow the instructions in “Migrating from WebSphere MQ Integrator Broker Version 2.1 to WebSphere Message Broker Version 6.0” on page 16, replacing all references to “WebSphere MQ Integrator Broker” with “WebSphere MQ Integrator”.

Migrating from WebSphere MQ Integrator Version 2.1 to WebSphere Message Broker with Rules and Formatter Extension Version 6.0

To migrate a broker domain from WebSphere MQ Integrator Version 2.1 to WebSphere Message Broker with Rules and Formatter Extension Version 6.0, follow the instructions in “Migrating from WebSphere MQ Integrator Broker Version 2.1 to WebSphere Message Broker Version 6.0” on page 16, replacing all references to “WebSphere MQ Integrator Broker” with “WebSphere MQ Integrator” and all references to “WebSphere Message Broker” with “WebSphere Message Broker with Rules and Formatter Extension”.

For instructions on how to migrate the New Era of Networks Rules and Formatter support for WebSphere MQ Integrator, see the New Era of Networks documentation.

Post-migration tasks

After migrating to WebSphere Message Broker Version 6.0 from a previous version, test the broker resources and components to verify that there is no loss or unexpected change of functionality. Some changes in behavior might be caused by defects being fixed between versions.

The following topics describe what to do after migrating from Version 2.1 to Version 6.0:

- “Removing the Configuration Manager database tables after migration”
- “Issuing commands from a program after migration” on page 50
- “Setting up a command environment” on page 51

Removing the Configuration Manager database tables after migration

This topic describes one of the tasks that you might want to complete after migration.

Before you start

Before you start this task, make sure that the migration was successful and that you no longer need to retain the flexibility to restore the Version 2.1 level of code.

If you preserve the assignments, topology, and topics data in the configuration repository during a migration from WebSphere MQ Integrator Broker Version 2.1 or WebSphere MQ Integrator Version 2.1 and do not delete the Configuration Manager, the Configuration Manager database tables contain development data that the Configuration Manager no longer requires after migration. When you are sure that the migration has been successful, delete the data by performing this task.

To clean up the Configuration Manager DB2 database tables, drop the following database tables by using the **DB2 DROP TABLE** command. If the Configuration Manager tables are in a DB2 database by themselves, drop the whole database by using the **DB2 DROP DATABASE** command.

- CBROKER
- CBROKERCEG
- CCOLLECTIVE
- CCOLLECTIVECBROKER
- CDELETE
- CEG
- CEGCMSGFLOW
- CEGCMSGPROJECT
- CLOG
- CMSGFLOW
- CMSGPROJECT
- CNEIGHBOURS
- COUTSTANDING
- CSUBSCRIBE
- CTOPIC
- CTOPICCTOPIC
- CTOPOLOGY
- CTRACE
- CUUIDLOCKS

Issuing commands from a program after migration

If you have programs that issue WebSphere MQ Event Broker Version 2.1 or WebSphere MQ Integrator Broker Version 2.1 commands, this topic describes how to run these programs unchanged after you have migrated to WebSphere Message Broker Version 6.0.

UNIX systems:

The names of WebSphere MQ Event Broker Version 2.1 commands start with the prefix “wmqps”, not “mqsi”. In addition, WebSphere MQ Event Broker commands and WebSphere MQ Integrator Broker commands are stored in different directories to where WebSphere Message Broker commands are stored. If you have programs, such as shell scripts, that issue WebSphere MQ Event Broker or WebSphere MQ Integrator Broker commands, continue to use these programs unchanged after migration by running the shell script migration.sh, which is supplied with WebSphere Message Broker. migration.sh creates the required symbolic links to the WebSphere Message Broker commands.

Run migration.sh only if you have migrated from WebSphere MQ Event Broker Version 2.1 or WebSphere MQ Integrator Broker Version 2.1. You do not need to run it if you have migrated from WebSphere MQ Integrator Version 2.1.

The following table displays the fully qualified path name of migration.sh for each operating system:

Operating system	Fully qualified path name of migration.sh
AIX	/usr/opt/mqsi/sample/migration/migration.sh
HP-UX Solaris	/opt/mqsi/sample/migration/migration.sh

migration.sh works by creating symbolic links, so you do not need to run it again after applying a maintenance update to WebSphere Message Broker.

Windows systems:

The names of WebSphere MQ Event Broker Version 2.1 commands start with the prefix “wmqps”, not “mqsi”. If you have programs, such as batch programs (or batch files), that issue WebSphere MQ Event Broker commands, continue to use these programs unchanged after migration by running the batch program migration.bat, which is supplied with WebSphere Message Broker. migration.bat makes a copy of each WebSphere Message Broker command that has an equivalent WebSphere MQ Event Broker command and gives the copied command its WebSphere MQ Event Broker name.

Run migration.bat only if you have migrated from WebSphere MQ Event Broker Version 2.1. You do not need to run it if you have migrated from WebSphere MQ Integrator Broker Version 2.1 or WebSphere MQ Integrator Version 2.1.

migration.bat is in the directory *WBIMB*\sample\migration, where *WBIMB* is the fully qualified path name of the directory where you have installed WebSphere Message Broker. You must run migration.bat from this directory.

migration.bat works by copying commands, so you must run it again after every maintenance update you apply to WebSphere Message Broker.

Setting up a command environment

After you have installed the product on any of the distributed systems, you must initialize the environment before you can use any runtime component or command. A profile is provided for this purpose.

If appropriate, you can extend the initialization performed by this profile; for example, for broker or user databases, or other products that you want to use within the broker. For example, you might want to configure your broker to run with Java 1.5 instead of the default version Java 1.4.2.

You *must* ensure that you run this profile each time you run one of the administrative commands, or start a component; for example, a broker.

For information on setting up your command and runtime environment on a Windows system, see Command environment: Windows platforms

For information on setting up your command and runtime environment on Linux and UNIX systems, see Command environment: Linux and UNIX systems

Migrating from Version 5.0 products

This section describes how to migrate to WebSphere Message Broker Version 6.0 or WebSphere Message Broker with Rules and Formatter Extension Version 6.0 from the following products:

- WebSphere Business Integration Event Broker Version 5.0
- WebSphere Business Integration Message Broker Version 5.0
- WebSphere Business Integration Message Broker with Rules and Formatter Extension Version 5.0

The instructions apply to all operating systems supported by the Version 5.0 products.

This section contains the following topics:

- "Planning and pre-migration tasks" on page 52
- "Migrating from WebSphere Business Integration Event Broker Version 5.0 to WebSphere Message Broker Version 6.0" on page 57
- "Migrating from WebSphere Business Integration Message Broker Version 5.0 to WebSphere Message Broker Version 6.0" on page 57
- "Migrating from WebSphere Business Integration Message Broker with Rules and Formatter Extension Version 5.0 to WebSphere Message Broker with Rules and Formatter Extension Version 6.0" on page 70
- "Post-migration tasks" on page 70

If you are already using WebSphere MQ Publish/Subscribe, migrate your applications to use the publish/subscribe functions that are provided by WebSphere Message Broker. See "Migrating publish/subscribe applications" on page 77.

Planning and pre-migration tasks

The following topics describe what to do before migrating from Version 5.0 to Version 6.0:

- "Planning for migration from Version 5.0"
- "Backing up WebSphere Business Integration Message Broker Version 5.0 resources" on page 56

Planning for migration from Version 5.0

This section explains how to plan your migration. Complete the following steps:

1. Decide how you want to migrate the product components:
   a. Find out what is new in Version 6.0 and learn about new and changed function. These changes might affect how you want to use your migrated components in the future.
   b. Decide where you want to migrate the product components. Migrate them to a different location on the same computer or to a second computer. For example, you might want to migrate components to another location to maintain availability during the migration.

      The following topics describe different migration scenarios:
      - "Planning to migrate a small domain" on page 53
      - "Planning to migrate a large domain" on page 53
      - "Planning to migrate multiple domains" on page 54
      - "Planning to migrate a high-availability domain" on page 55
   c. Decide when you want to migrate the product components. You might want to preserve some components at the Version 5.0 level for now, and migrate them later.
   d. Decide the order in which you want to migrate your components. You can migrate components in any order, but your specific circumstances might mean that you need to migrate components in a particular order.

   "Coexistence with previous versions and other products" on page 4 shows you how WebSphere Message Broker Version 6.0 can coexist on the same computer with previous versions of the product, and how Version 6.0 components can operate with components from previous versions.
2. Decide how you want to use your existing resources with WebSphere Message Broker Version 6.0.

   Except for user-defined extensions and mapping files, you do not need to perform any tasks to migrate your development and deployment resources, such as message flow files, message set definition files, ESQL files, XML Schema files, and broker archive files. You can start using these resources with WebSphere Message Broker Version 6.0 immediately.

   All Version 5.0 and Version 5.1 user-defined node projects must be upgraded to work with the Version 6.0 Message Brokers Toolkit. Upgrade a project by cleaning it: click **Project** → **Clean**. When you clean a project, the extension point that is required by Version 6.0 to compile the ESQL files contained in the user-defined extension is created in the project plugin.xml file.

   After you have migrated Version 5.0 mapping files (.mfmap) to Version 6.0 mapping files (.msgmap) using the **mqsimigratemfmaps** command, edit them in the Version 6.0 Message Brokers Toolkit. There is no migration tool to migrate a mapping that is called from ESQL.

   After you start using your current resources in the Version 6.0 Message Brokers Toolkit, there are then restrictions on reusing these same resources with the

Version 5.0 or Version 5.1 Message Brokers Toolkit. For more information, see "Conditions for using migrated resources with previous versions of the Message Brokers Toolkit" on page 6

3. Decide whether you need to carry out any testing to ensure a successful migration.

   The purpose of testing your migration is to identify any problems that might arise during migration. For example, if problems arise you might need to restore some migrated resources to the Version 5.0 level that you backed up before you started the migration, and any post-migration changes to these resources would be lost. Migrating your development and test domains before migrating your production domain allows you to identify problems like this and develop a strategy for dealing with further problems.
4. Optional: When you are ready to migrate, run the **mqsimigratecomponents** command with the **-c** parameter. This performs a pre-migration check against the Version 5.0 components to ensure that they can be migrated. The pre-migration check identifies potential problems and allows you to correct them before proceeding with migration.

After you have planned your migration, back up your resources.

Planning to migrate a small domain:

This topic explains how to plan to migrate a small domain.

It is likely that on a small domain, many or all components are installed on a single computer. In this situation, the simplest approach is to migrate all the components on that single computer at the same time. If this is not possible, install WebSphere Message Broker Version 6.0 at a different location on the same computer, or on a second computer, and migrate each component separately, because Version 5.0 components can coexist with Version 6.0 components within a single domain. See "Coexistence with previous versions and other products" on page 4 for more information.

Migrate components in the order that best suits your environment. For example, a typical order in which to migrate the three major components might be as follows:
1. Instances of the Message Brokers Toolkit
2. Configuration Manager
3. Brokers

If you migrate instances of the Message Brokers Toolkit first, you can begin to develop message flows that use features that are new in WebSphere Message Broker Version 6.0. If you migrate the Configuration Manager, administrators can prepare your Version 6.0 broker domain for deployment. Finally, when you migrate the brokers, you can run your new message flows.

Depending on the goals of migration, the order in which you migrate components can be different. For example, if the primary reason for migration is to make use of new broker functions, you might want to migrate the brokers first. Or, for example, if you plan to move your Configuration Manager to a new operating system, you might want to migrate the Configuration Manager first to discover any operating system or environment issues early on.

Before you start to perform migration, back up your resources.

Planning to migrate a large domain:

This topic explains how to plan to migrate a large domain.

If you are using the Version 5.1 Message Brokers Toolkit, replace all references in this topic to ″Version 5.0″ with ″Version 5.1″.

On a large domain, it might not be possible to migrate all components at the same time. In this situation, the migration of the components might need to be split up. For example, the order of migration might be:

1. Migrate a test Message Brokers Toolkit and a test Configuration Manager
2. Migrate a test broker
3. Migrate more instances of the Message Brokers Toolkit
4. Migrate more brokers
5. Migrate remaining components

The migration order is flexible because Version 6.0 components can coexist with Version 5.0 components within a single domain. However, there are restrictions on coexistence. In particular, these will affect you if you plan to stagger the migration of several instances of the Message Brokers Toolkit, as some files cannot be read using the Version 5.0 Message Brokers Toolkit after they have been saved using the Version 6.0 Message Brokers Toolkit. See "Coexistence with previous versions and other products" on page 4 for more information.

Before you start to perform migration, back up your resources.

Planning to migrate multiple domains:

This topic explains how to plan to migrate multiple domains.

A typical configuration for WebSphere Message Broker is to use a set of three or more domains, consisting of development, test, and production domains.

Development domain
In a development domain, message flow developers create message flows and unit test them in a sandbox environment. The brokers in the domain are not responsible for handling business-critical data.

Test domain
Message flows in the development domain are eventually promoted to a test domain, where message flows are tested against recent, but not live, production data in a realistic broker configuration.

Production domain
When message flows in the test domain are deemed to be robust enough, they are promoted to the production domain. This is the domain that is responsible for performing actual business transactions, and message flows in the domain work with live data. Non-critical updates to production flows usually take place only at predefined service intervals.

Development and test domains must be migrated before production domains. Migrating the development domain first minimizes potential downtime associated with any migration. In addition, development domains are likely to require access to new broker functions before test and production domains. As message flows that make use of new functions are developed, the test and production domains must be migrated before the new message flows are promoted to them.

As each Message Brokers Toolkit can administer multiple domains, be careful when migrating instances of the Message Brokers Toolkit not to affect any domains that have not been migrated yet.

A Version 6.0 Message Brokers Toolkit can administer Version 5.0 domains and vice versa.

Before you start to perform migration, back up your resources.

Planning to migrate a high-availability domain:

This topic explains how to plan to migrate a high-availability domain.

High availability is the requirement for a system to be running all, or for a very high proportion of, the time. Providing a high-availability environment involves having multiple brokers so that some brokers can be used for backup purposes. The entire production domain might be duplicated so that the backup system can be switched on in the event of a problem.

Migrating in a high-availability environment presents a problem because there is a vulnerable period between stopping the previous version of the product and installing and starting WebSphere Message Broker Version 6.0. One solution is to install and configure WebSphere Message Broker Version 6.0 while the previous version of the product is running in parallel, and then switch to WebSphere Message Broker Version 6.0. Install WebSphere Message Broker Version 6.0 on either the same computers that are running the previous version of the product, or on a different set of computers.

The following sections of this topic provide a general framework for how you might want to migrate your high-availability domain. You might need to adjust these instructions to suit your specific circumstances. Refer to the product-specific migration topics for detailed migration instructions.

Preparing to migrate a high-availability domain:
To achieve high-availability migration, your environment must be configured correctly. Complete the following tasks (production and test domains are described in "Planning to migrate multiple domains" on page 54):

1. Ensure that the production domain that is to be migrated is configured for high availability. You must have at least three brokers supporting the executing applications, so that when broker 1 is being migrated, brokers 2 and 3 can provide backup support for each other. This is a standard setup for a high-availability domain at all times, and is not specific to migration.
2. Ensure that the test domain is identical to the production domain. This means that you can identify any problems during the migration of the test domain and resolve them before you migrate the production domain.

Preparing to migrate the test domain:

By migrating the test domain, you can ensure that operation is successful after migration and solve any problems before migrating the production domain. In message flow and message set development, each developer is often responsible for a particular area of functionality. Test the migration of your resources to WebSphere Message Broker Version 6.0 for each functional area in turn. The following steps describe one way in which to migrate the test domain to minimize disruption. Refer to the product-specific migration topics for detailed migration instructions.

1. After you have backed up your resources and installed Version 6.0, create a Version 6.0 Configuration Manager and broker on as many computers as needed to mirror the production domain.
2. Move one functional area to Version 6.0 by migrating the Message Brokers Toolkit and the resources that are used by the developer who is responsible for that area. Test that the migrated resources work as required by deploying them and checking the results. Resolve any problems, then migrate the remaining functional areas. Migrating the functional areas one by one means that existing development can continue on the previous version of the product.
3. In the same way that you migrate the functional areas individually, migrate the previous version's brokers individually to Version 6.0. When you have migrated one broker, test that your resources still work as required by deploying the migrated resources for which this broker is responsible. Resolve any problems and repeat for the remaining brokers. Migrating the brokers one at a time means that only small changes are made at each stage. If problems occur, other brokers that have not been migrated yet can process requests while the migration problems are resolved.

Preparing to migrate the production domain:
After you have migrated the test domain and resolved any problems, prepare to migrate the production domain. Before you migrate your resources, configure an administration system with the Version 6.0 Message Brokers Toolkit. This allows you to administer the new Version 6.0 components. Refer to the product-specific migration topics for detailed migration instructions.

When you are ready to perform migration, back up your resources.

Backing up WebSphere Business Integration Message Broker Version 5.0 resources

This topic describes how to back up your resources before migration.

Before you start:

Read the following planning topics:
- "Planning to migrate a small domain" on page 53
- "Planning to migrate a large domain" on page 53
- "Planning to migrate multiple domains" on page 54
- "Planning to migrate a high-availability domain" on page 55

Before you carry out any migration tasks, back up your WebSphere Business Integration Message Broker Version 5.0 resources by completing the following steps:
1. Back up the configuration repository.
2. Back up the broker database tables.
3. Back up other resources, such as message flow files, message set definition files, ESQL files, mapping files, XML Schema files, and broker archive (bar) files.
4. Back up your Message Brokers Toolkit resources by copying your workspace directory to a different location.

For detailed instructions on how to back up these resources, see the WebSphere Business Integration Message Broker Version 5.0 information center.

After you have backed up your WebSphere Business Integration Message Broker Version 5.0 resources, proceed to one of the following topics:

- “Migrating from WebSphere Business Integration Event Broker Version 5.0 to WebSphere Message Broker Version 6.0”
- “Migrating from WebSphere Business Integration Message Broker Version 5.0 to WebSphere Message Broker Version 6.0”
- “Migrating from WebSphere Business Integration Message Broker with Rules and Formatter Extension Version 5.0 to WebSphere Message Broker with Rules and Formatter Extension Version 6.0” on page 70

Migrating from WebSphere Business Integration Event Broker Version 5.0 to WebSphere Message Broker Version 6.0

To migrate from WebSphere Business Integration Event Broker Version 5.0 to WebSphere Message Broker Version 6.0, follow the instructions in “Migrating from WebSphere Business Integration Message Broker Version 5.0 to WebSphere Message Broker Version 6.0,” replacing all references to ″WebSphere Business Integration Message Broker Version 5.0″ with ″WebSphere Business Integration Event Broker Version 5.0″.

In some places, the instructions for migrating from WebSphere Business Integration Message Broker Version 5.0 to WebSphere Message Broker Version 6.0 give information about the file types to which the instructions apply. There are file types in WebSphere Message Broker that do not exist in WebSphere Event Broker, so ignore any that are specific to WebSphere Message Broker.

For example, if the instructions ask you to perform an action on message flow files, message set definition files, ESQL files, mapping files, XML Schema files, and broker archive files, you need to perform the action on message flow files and broker archive files only, because the other files types apply to WebSphere Message Broker only.

Migrating from WebSphere Business Integration Message Broker Version 5.0 to WebSphere Message Broker Version 6.0

This topic describes how to migrate from WebSphere Business Integration Message Broker Version 5.0 to WebSphere Message Broker Version 6.0.

Before you start

Before you start migration:

- ensure that your installation of WebSphere Business Integration Message Broker Version 5.0 is at service level Fix Pack 4 or later
- read about coexistence with previous versions and other products
- plan your migration
- back up your Version 5.0 resources

Each of the tasks listed at the end of this topic explains how to migrate a component. Each task also tells you at which point during the migration of that component you need to install WebSphere Message Broker Version 6.0.

To migrate a single component, complete the task for that component. To migrate all components, complete all of the tasks listed, but plan your migration so that you apply the tasks as required by your situation. For example, because each task includes steps for installing WebSphere Message Broker Version 6.0, you might want to carry out the tasks in parallel, rather than by completing one task before starting the next task.

The order in which you migrate components does not matter, so complete the tasks in any order. The following list of tasks are presented in a typical order of migration:
- Migrating the Message Brokers Toolkit
- Migrating a Configuration Manager
- Migrating a broker
- Migrating a User Name Server

After you have completed the migration, see the post-migration tasks for information about tasks that you might want to perform after migration.

Migrating the Message Brokers Toolkit from WebSphere Business Integration Message Broker Version 5.0 to WebSphere Message Broker Version 6.0

If you are using the Version 5.1 Message Brokers Toolkit, follow the instructions in this topic, replacing all references to ″Version 5.0 Message Brokers Toolkit″ with ″Version 5.1 Message Brokers Toolkit″.

When you start using your resources in the Version 6.0 Message Brokers Toolkit, there are restrictions to using these same resources again with the Version 5.0 or Version 5.1 Message Brokers Toolkit. For more information, see “Conditions for using migrated resources with previous versions of the Message Brokers Toolkit” on page 6.

To migrate the Version 5.0 Message Brokers Toolkit to Version 6.0, complete the following steps:
1. Install WebSphere Message Broker Version 6.0 in a different location from WebSphere Business Integration Message Broker Version 5.0.
2. Migrate any Version 5.0 mappings using the **mqsimigratemfmaps** command. (There is no migration tool to migrate a mapping that is called from ESQL.)
3. When you start the Version 6.0 Message Brokers Toolkit for the first time, you are prompted to enter a workspace location. Enter the directory where the Version 5.0 or Version 5.1 Message Brokers Toolkit workspace that you want to migrate is located. In Version 5.0, the default location is <BrokersInstallPath>/ eclipse/workspace.
   - Version 5.0 message flows are converted to the Version 6.0 format automatically when you save them for the first time. The Version 5.0 Message Brokers Toolkit cannot read message flows that have been saved in Version 6.0 format.
   - If your message flows use MQe nodes, follow the instructions in “Migrating a message flow that contains WebSphere MQ Everyplace nodes” on page 106. If your message flows use XML style sheets, follow the instructions in “Migration of style sheets and XML files” on page 113.
   - Version 5.0 message sets are converted to the Version 6.0 format automatically when you save them for the first time. The Version 5.0 Message Brokers Toolkit cannot read message sets that have been saved in Version 6.0 format.
4. Clean and rebuild your workspace.

5. To migrate user-defined nodes from Version 5.0 or Version 5.1, import the user-defined node project into the Version 6.0 workbench and rebuild the project. If you are migrating user-defined nodes from Version 5.0, perform the following additional steps:
   a. Modify the `<requires>` element in the plugin.xml file in the user-defined node project root to match the following example. Ensure that you delete all extraneous entries from the Version 5.0 or Version 5.1 plugin.xml file so that it matches the following example exactly:
      ```
      <requires>
       <import match="greaterOrEqual" plugin="com.ibm.etools.mft.api" version="6.0.0"/>
      </requires>
      ```
   b. Modify the ″org.eclipse.help.contexts″ extension in the same plugin.xml file to match the following example:
      ```
      <extension point="org.eclipse.help.contexts">
       <contexts file="HelpContexts.xml"/>
      </extension>
      ```
   c. Restart the Message Brokers Toolkit using the **-clean** option.

If you have configured MCA users on the WebSphere MQ channels to the Configuration Manager, you might encounter problems when you try to connect to the migrated Configuration Manager from the Version 6.0 Message Brokers Toolkit. To resolve this, configure access control lists (ACLs) for users who are running the Version 6.0 Message Brokers Toolkit. Follow the instructions in Considering security for the workbench.

When you have migrated the Message Brokers Toolkit, ensure that you have migrated the rest of the components:
- Migrate the Configuration Manager
- Migrate the broker
- Migrate the User Name Server

then see the post-migration tasks for information about tasks that you might want to perform after migration.

Migrating a Configuration Manager from WebSphere Business Integration Message Broker Version 5.0 to WebSphere Message Broker Version 6.0

In WebSphere Message Broker Version 6.0, the Configuration Manager no longer uses an external database to store the domain configuration but uses an internal repository instead. The Configuration Manager is also now available on more operating systems than in WebSphere Business Integration Message Broker Version 5.0.

From the following list of topics in this section, select the topic that is appropriate to your environment.
- Migrating to a different computer that has DB2 installed
- Migrating to a different computer that does not have DB2 installed
- Migrating on the same computer

Migrating a Version 5.0 Configuration Manager to Version 6.0 on the same computer:

This topic describes one of three ways to migrate a Version 5.0 Configuration Manager to Version 6.0.

Before you start

Read about "Coexistence with previous versions and other products" on page 4.

The following steps describe how to replace the Version 5.0 Configuration Manager with a Version 6.0 Configuration Manager:

1. Install WebSphere Message Broker Version 6.0 in a location other than where WebSphere Business Integration Message Broker Version 5.0 is installed.
2. Stop the Version 5.0 Configuration Manager.
3. Use the **mqsimigratecomponents** command to migrate your Version 5.0 Configuration Manager.
4. Start the Version 6.0 Configuration Manager.

If you do not have additional hardware available but want to confirm the successful migration of the Configuration Manager before replacing the Version 5.0 Configuration Manager, migrate the Configuration Manager to a different location on the same computer. If you use this method, you can also change the name of the Configuration Manager. Complete the following steps:

1. Install WebSphere Message Broker Version 6.0 in a location other than where WebSphere Business Integration Message Broker Version 5.0 is installed.
2. Stop the Version 5.0 Configuration Manager.
3. Create a Version 6.0 Configuration Manager using the **mqsicreateconfigmgr** command. Make sure that the Version 6.0 Configuration Manager has a different name and a different queue manager name to the Version 5.0 Configuration Manager. Use the **-n** parameter of the **mqsicreateconfigmgr** command to specify the Version 5.0 Configuration Manager's database.
4. Configure WebSphere MQ to allow the Version 6.0 Configuration Manager to communicate with the broker network. For example, you might need to configure channels and transmission queue definitions.
5. Start the Version 6.0 Configuration Manager.
6. In order to associate all brokers in the domain with the new Configuration Manager, so that the brokers publish their status messages to the correct queue manager, a complete topology deploy of the new Configuration Manager is required. Deploy the topology using either the Message Brokers Toolkit or the command-line interface. When the topology deploy has completed re-subscribing to the brokers in the domain, the brokers are managed by the new Configuration Manager; do not use the previous Configuration Manager. You can now delete the previous Configuration Manager.

If you have configured MCA users on the WebSphere MQ channels to the Configuration Manager, you might encounter problems when you try to connect to the migrated Configuration Manager from the Version 6.0 Message Brokers Toolkit. To resolve this, configure access control lists (ACLs) for users who are running the Version 6.0 Message Brokers Toolkit. Follow the instructions in Considering security for the workbench.

Before you migrate your brokers, or make any configuration changes, ensure that your Configuration Manager has been migrated correctly by performing the following checks:

1. Start the migrated Configuration Manager.
2. Start the Message Brokers Toolkit and connect it to the Configuration Manager's domain.
3. Check the list of domain components and see if any errors are produced.

4. Check the list of alerts for the domain and see if anything has stopped.
5. Check that you can open the Event Log.
6. Optional: If you have a broker running, stop and start a message flow, and check for a successful response; check that the Alerts view is updated when the message flow is stopped, and clear when the message flow is started.

If these steps are successful, your Configuration Manager has been migrated successfully. If they are unsuccessful, resolve any problems before you migrate further components or make configuration changes.

When you have migrated the Configuration Manager, ensure that you have migrated the rest of the components:
- Migrate the Message Brokers Toolkit
- Migrate the broker
- Migrate the User Name Server

then see the post-migration tasks for information about tasks that you might want to perform after migration.

Migrating a Version 5.0 Configuration Manager to Version 6.0: to a different computer that has DB2 installed:

This topic describes one of three ways to migrate a Version 5.0 Configuration Manager to Version 6.0.

Before you start

Read about "Coexistence with previous versions and other products" on page 4.

An existing Windows Configuration Manager can be migrated to a Version 6.0 Configuration Manager on any of the supported operating systems via a JDBC Type 4 Universal DB2 connection.

To migrate a Configuration Manager to a different computer that has a JDBC Type 4 Universal DB2 connection installed, complete the following steps:
1. Install WebSphere Message Broker Version 6.0 on the computer to which you are migrating your Version 5.0 Configuration Manager.
2. Add the following files to the environment in which you are going to create the new Configuration Manager. On z/OS, update the BIPCPROF file and submit the BIPGEN job. On all other operating systems, update the local environment.
   a. Add the following files to the CLASSPATH:

      db2 install/jcc/classes/sqlj.zip
      db2 install/jcc/classes/db2jcc.jar
      db2 install/jcc/classes/db2jcc_javax.jar
      db2 install/jcc/classes/db2jcc_license_cisuz.jar
   b. Add to the computer specific environment variable for libraries (for example LIBPATH on z/OS):

      db2 install/jcc/lib
   c. Add to The PATH:

      db2 install/jcc/bin

   where *db2 install* is where DB2 is installed at your location (for example /usr/lpp/db2710/db2710).

3. Stop the Version 5.0 Configuration Manager.
4. Create a Version 6.0 Configuration Manager on the second computer by issuing the **mqsicreateconfigmgr** command (or the BIPCRCM job on z/OS), specifying the database name, user name, and password required to access the Version 5.0 Configuration Manager database. For example:

   -u (userid)
      The database userid on the Windows computer for the Configuration Manager

   -p (password)
      The database password on the Windows computer for the Configuration Manager

   -n (database name)
      //server:port/database name

   where:
   - *//server* is the IP address of the Windows computer (for example `//9.20.235.197`)
   - *:port* is the port number of DB2 on Windows
   - */database name* is the name of the Windows Configuration Manager database (for example `/MQSICMDB`)

   For example: `//9.20.235.197:50000/MQSICMDB`

   Use different queue manager names for the two Configuration Managers to maintain uniqueness in the WebSphere MQ network.

   When you create the Version 6.0 Configuration Manager, domain information from the Version 5.0 Configuration Manager database is copied automatically to the Version 6.0 Configuration Manager internal repository so it might take a few minutes to migrate the database.
5. On the second computer, configure WebSphere MQ to allow the Version 6.0 Configuration Manager to communicate with the broker network. For example, you might need to configure channels, transmission queues, and remote queue manager definitions.
6. Start the Version 6.0 Configuration Manager.
7. In order to associate all brokers in the domain with the new Configuration Manager, so that the brokers publish their status messages to the correct queue manager, a complete topology deploy is required on the second computer. Deploy the topology using either the Message Brokers Toolkit or the command-line interface.

   When the topology deploy has completed re-subscribing to the brokers in the domain, the brokers are managed by the new Configuration Manager on the new system; do not use the previous Configuration Manager.
8. If necessary, uninstall DB2 and the Version 5.0 broker.

Before you migrate your brokers, or make any configuration changes, ensure that your Configuration Manager has been migrated correctly by performing the following checks:
1. Start the migrated Configuration Manager.
2. Start the Message Brokers Toolkit and connect it to the Configuration Manager's domain.
3. Check the list of domain components and see if any errors are produced.
4. Check the list of alerts for the domain and see if anything has stopped.
5. Check that you can open the Event Log.

6. Optional: If you have a broker running, stop and start a message flow, and check for a successful response; check that the Alerts view is updated when the message flow is stopped, and clear when the message flow is started.

If these steps are successful, your Configuration Manager has been migrated successfully. If they are unsuccessful, resolve any problems before you migrate further components or make configuration changes.

When you have migrated the Configuration Manager, ensure that you have migrated the rest of the components:

- Migrate the Message Brokers Toolkit
- Migrate the broker
- Migrate the User Name Server

then see the post-migration tasks for information about tasks that you might want to perform after migration.

Migrating a Version 5.0 Configuration Manager to Version 6.0: to a different computer that does not have DB2 installed:

This topic describes one of three ways to migrate a Version 5.0 Configuration Manager to Version 6.0.

Before you start

Read about “Coexistence with previous versions and other products” on page 4.

To migrate a Configuration Manager to a different computer that does not have DB2 or a JDBC client for DB2 installed, complete the following steps:

1. Migrate the Version 5.0 Configuration Manager to Version 6.0 on the same computer by completing the following steps:
   a. Install WebSphere Message Broker Version 6.0 in a location other than where WebSphere Business Integration Message Broker Version 5.0 is installed.
   b. Stop the Version 5.0 Configuration Manager.
   c. Launch a Version 6.0 Command Console and enter the **mqsimigratecomponents** command to migrate your Version 5.0 Configuration Manager.
   d. Start the Version 6.0 Configuration Manager.

      When you start the Version 6.0 Configuration Manager for the first time, it automatically detects domain information in the DB2 database of the Version 5.0 Configuration Manager, and migrates it into the internal repository of the Version 6.0 Configuration Manager. You do not need to do anything, and the DB2 database is not modified.
2. Stop the Version 6.0 Configuration Manager, and make a copy of its internal repository using the **mqsibackupconfigmgr** command.
3. Install WebSphere Message Broker Version 6.0 on the second computer.
4. On the second computer, create a Version 6.0 Configuration Manager by issuing the **mqsicreateconfigmgr** command (or the BIPCRCM job on z/OS).

   You must maintain unique queue manager names in your WebSphere MQ network. When you create the new Configuration Manager, choose a new queue manager name.

5. On the second computer, configure WebSphere MQ to allow the Version 6.0 Configuration Manager to communicate with the broker network. For example, you might need to configure channels, transmission queues, and remote queue manager definitions.
6. On the second computer, issue the **mqsirestoreconfigmgr** command (or the BIPRSCM job on z/OS) to overwrite the contents of the empty Version 6.0 Configuration Manager repository with the repository that you backed up from the original computer.
7. On the second computer, start the Version 6.0 Configuration Manager.
8. In order to associate all brokers in the domain with the new Configuration Manager, so that the brokers publish their status messages to the correct queue manager, a complete topology deploy is required on the second computer. Deploy the topology using either the Message Brokers Toolkit or the command-line interface.

   When the topology deploy has completed re-subscribing to the brokers in the domain, the brokers are managed by the new Configuration Manager on the new system; do not use the previous Configuration Manager.

Before you migrate your brokers, or make any configuration changes, ensure that your Configuration Manager has been migrated correctly by performing the following checks:
1. Start the migrated Configuration Manager.
2. Start the Message Brokers Toolkit and connect it to the Configuration Manager's domain.
3. Check the list of domain components and see if any errors are produced.
4. Check the list of alerts for the domain and see if anything has stopped.
5. Check that you can open the Event Log.
6. Optional: If you have a broker running, stop and start a message flow, and check for a successful response; check that the Alerts view is updated when the message flow is stopped, and clear when the message flow is started.

If these steps are successful, your Configuration Manager has been migrated successfully. If they are unsuccessful, resolve any problems before you migrate further components or make configuration changes.

When you have migrated the Configuration Manager, ensure that you have migrated the rest of the components:
- Migrate the Message Brokers Toolkit
- Migrate the broker
- Migrate the User Name Server

then see the post-migration tasks for information about tasks that you might want to perform after migration.

Migrating a broker from WebSphere Business Integration Message Broker Version 5.0 to WebSphere Message Broker Version 6.0

To migrate a broker from WebSphere Business Integration Message Broker Version 5.0 to WebSphere Message Broker Version 6.0, see the appropriate topic for your operating system:
- Distributed systems
- z/OS

Migrating a Version 5.0 broker to Version 6.0 on distributed operating systems:

You can migrate a Version 5.0 broker directly to Version 6.0. However, if the lack of availability of a broker during migration is an issue, move the Version 5.0 broker's application logic onto a Version 6.0 broker. Select the topic that is appropriate to your environment.

If you are using the Version 5.1 Message Brokers Toolkit, replace all references to ″Version 5.0″ with ″Version 5.1″.
- "Migrating a Version 5.0 broker to Version 6.0"
- "Migrating a Version 5.0 broker's application logic to another installation"

Migrating a Version 5.0 broker to Version 6.0:

Before you start

Before migrating a broker, ensure that you do not have any aggregations in progress. When you migrate a broker to Version 6.0, any live data that is being stored for aggregations in progress will be lost.

If the broker runs in a locale that is not listed in the Locales topic, check that the code page is one of the supported code pages and that the locale is set up correctly.

To migrate a Version 5.0 broker on distributed operating systems to Version 6.0 at the same location, complete the following steps:
1. Install WebSphere Message Broker Version 6.0 in the same location as WebSphere Business Integration Message Broker Version 5.0.
2. Stop the Version 5.0 broker.
3. Stop any channels to the Version 5.0 broker.
4. Update the .odbc.ini file to Version 6.0 format by following the instructions in "Updating your ODBC definitions when migrating" on page 29.
5. Launch a Version 6.0 Command Console and enter the **mqsimigratecomponents** command to migrate your Version 5.0 components.
6. Start the Version 6.0 broker.

Migrating a Version 5.0 broker's application logic to another installation:

Before you start

To complete this task you must have either a Version 5.0 or a Version 6.0 Message Brokers Toolkit available.

Note:

> Before migrating a broker, ensure that you do not have any aggregations in progress. When you migrate a broker to Version 6.0 any live data that is being stored for aggregations in progress will be lost.
>
> If the broker runs in a locale that is not listed in the Locales topic, check that the code page is one of the supported code pages and that the locale is set up correctly.

To migrate a Version 5.0 broker on distributed operating systems to Version 6.0 at a different location on the same computer, or to a different computer, complete the following steps:

1. Install WebSphere Message Broker Version 6.0 either as a new instance on the computer where WebSphere Business Integration Message Broker Version 5.0 is installed, or on a different computer. For detailed instructions, see the Installation Guide.
2. Update the .odbc.ini file to Version 6.0 format by following the instructions in "Updating your ODBC definitions when migrating" on page 29.
3. Create a Version 6.0 broker and start it. Ensure that the Version 6.0 broker's name is different from the name of the Version 5.0 broker.

   Brokers cannot share queue managers. You cannot migrate a broker to a queue manager that is already being used by an existing broker.
4. Add the new broker to the domain by completing the following tasks:
   a. Create a Configuration Manager on Windows
   b. Create a domain connection
   c. Connect to the broker domain
   d. Add a broker to a broker domain
5. Write a list of the execution groups that you have on the Version 5.0 broker, and create these same execution groups on the new broker. Use either the Version 5.0 or Version 6.0 Message Brokers Toolkit to do this.
6. Deploy the message flows and message sets of the Version 5.0 broker to the Version 6.0 broker. Use either the Version 5.0 or Version 6.0 Message Brokers Toolkit to do this.
7. Configure any other relevant properties of the Version 5.0 broker on the Version 6.0 broker. For example, you might need to configure properties that you set using the **mqsichangeproperties** command, or as a result of using certain nodes, such as Publication or SCADA nodes.
8. Stop the Version 5.0 broker.
9. Remove the Version 5.0 broker from the workbench.
10. Redeploy the topology.
11. Delete the Version 5.0 broker.

When you have migrated the broker, ensure that you have migrated the rest of the components:

- Migrate the Message Brokers Toolkit
- Migrate the Configuration Manager
- Migrate the User Name Server

then see the post-migration tasks for information about tasks that you might want to perform after migration.

Updating your ODBC definitions when migrating:

As part of migrating a broker, you must create ODBC definitions for broker and user databases, with database drivers that are appropriate for the new product version. You must do this before you run the **mqsimigratecomponents** command. The way that you do this depends on your operating system:

Windows

ODBC definitions are stored by the operating system. There is only one repository of ODBC data source names. You do not need to change DB2 and SQL Server data source definitions. Change Oracle and Sybase data source definitions according to the instructions in "Changing the 32-bit ODBC connection and XA resource manager definitions for a migrated broker" on page 29.

Linux and UNIX

ODBC definitions are stored in a text file. For previous versions of the product, this file was located in /var/mqsi/odbc/.odbc.ini. When working with Version 6.0, create a new text file under this directory, but do not replace the existing file. This means that you do not affect existing brokers. Set the environment variable ODBCINI to point to the new file; perform this step as part of the setup for a Version 6.0 Command Console.

In your new Version 6.0 ODBC configuration file, ensure that you list all of the data sources that will be used by brokers that you want to migrate. For information on how to create a new ODBC.INI file, see Connecting to a database from Linux and UNIX systems. If you are using XA to enable coordinated transactions, you must also update your qm.ini file for WebSphere MQ, as described in "Changing the 32-bit ODBC connection and XA resource manager definitions for a migrated broker" on page 29.

If you revert to a previous version of WebSphere Message Broker, you must reverse the changes that you make to the ODBC definitions.

- On UNIX, reset the ODBCINI environment variable to point to the previous version of the odbc.ini file.
- On Windows, use the Control Panel to adjust the ODBC settings.

Update the odbc.ini after you have run the **mqsimigratecomponents** command, but before you restart the broker.

Migrating a Version 5.0 broker to Version 6.0 on z/OS:

Before you start

- Ensure that you are familiar with the steps involved in creating a broker on z/OS.
- The JCL (Job Control Language) uses the **mqsimigratecomponents** command to migrate a broker on z/OS. This command takes many parameters, which you must understand fully before attempting to migrate the broker.
- Before migrating a broker, ensure that you do not have any aggregations in progress. When you migrate a broker to Version 6.0 any live data that is being stored for aggregations in progress will be lost.
- If the broker runs in a locale that is not listed in the Locales topic, check that the code page is one of the supported code pages and that the locale is set up correctly.

To migrate a Version 5.0 broker to Version 6.0 on z/OS, complete the following steps:

1. Stop the Version 5.0 broker.
2. Back up the broker database tables.
3. Create a new broker PDSE.
4. Copy all broker JCL from the Version 6.0 installed SBIPPROC and SBIPSAMP PDSEs to the new broker PDSE and customize them all. See Customizing the broker JCL for more information.

When you customize the new Version 6.0 JCL for this migration, do not make the ++*HOME*++ JCL variable the same value as the existing Version 5.0 component directory, otherwise the previous ENVFILE will be overwritten when you submit the BIPGEN job. Set the ++*HOME*++ variable to the Started Task user ID home directory (for example, /u/mq01brk).

a. Customize the BIPEDIT file using values that are defined in the broker's Version 5.0 mqsicompcif file. "Equivalent customization variables in mqsicompcif and BIPEDIT" on page 123 lists the customization variables that are used in the Version 5.0 mqsicompcif file, and the equivalent variables in the Version 6.0 BIPEDIT file.
b. Copy any additional changes that you have made to the environment file, `ENVFILE`, and the ODBC initialization file, `dsnaoini`, to BIPBPROF and BIPDSNAO in the component data set. Submit the BIPGEN job to create the environment file `ENVFILE`.
c. Customize and submit the BIPMGCMP job. This migrates the registry, queues and broker database. As part of the database migration, database tables are created, or deleted, and dropped, so you must you have the correct DB2 privileges. These privileges would be the same as those required for running the **mqsicreatebroker** command.

5. Copy the started task JCL (BIPBRKP) to the procedures library. When you copy the started task, keep a second copy of the original in a safe place for backup purposes.
6. The verification program will run when you start the Version 6.0 broker.

When you have migrated the broker, ensure that you have migrated the rest of the components:

- Migrate the Message Brokers Toolkit
- Migrate the Configuration Manager
- Migrate the User Name Server

then see the post-migration tasks for information about tasks that you might want to perform after migration.

Migrating a User Name Server from Version 5.0 to Version 6.0

To migrate a User Name Server from WebSphere Business Integration Message Broker Version 5.0 to WebSphere Message Broker Version 6.0, see the appropriate topic for your operating system:

- Distributed systems
- z/OS

Migrating a Version 5.0 User Name Server to Version 6.0 on distributed systems:
If the lack of availability of a User Name Server during migration is an issue, migrate your User Name Server to a different location on the same computer or on a second computer.

1. Install WebSphere Message Broker Version 6.0 either as a new instance on the computer where WebSphere Business Integration Message Broker Version 5.0 is installed, or on a different computer.
2. Stop the Version 5.0 User Name Server.
3. Launch a Version 6.0 Command Console and enter the **mqsimigratecomponents** command to migrate the User Name Server.
4. Start the Version 6.0 User Name Server.

When you have migrated the User Name Server, ensure that you have migrated the rest of the components:

- Migrate the Message Brokers Toolkit
- Migrate the Configuration Manager
- Migrate the broker

then see the post-migration tasks for information about tasks that you might want to perform after migration.

Migrating a Version 5.0 User Name Server to Version 6.0 on z/OS:

Before you start

- Ensure that you are familiar with the steps involved in creating a User Name Server on z/OS.
- The JCL uses the **mqsimigratecomponents** command to migrate a User Name Server on z/OS. This command takes many parameters, which you must understand fully before attempting to migrate the User Name Server.

To migrate a Version 5.0 User Name Server to Version 6.0 on z/OS, complete the following steps:

1. Stop the Version 5.0 User Name Server.
2. Create a new User Name Server PDSE.
3. Copy all User Name Server JCL from the Version 6.0 installed SBIPPROC and SBIPSAMP PDSEs to the new User Name Server PDSE and customize them all.

   When you customize the new Version 6.0 JCL for this migration, do not make the ++*HOME*++ JCL variable the same value as the existing Version 5.0 component directory, otherwise the previous ENVFILE will be overwritten when you submit the BIPGEN job. Set the ++*HOME*++ variable to the Started Task user ID home directory (for example, /u/mq01uns).
   a. Customize the BIPEDIT file using values that are defined in the User Name Server's Version 5.0 mqsicompcif file. "Equivalent customization variables in mqsicompcif and BIPEDIT" on page 123 lists the customization variables that are used in the Version 5.0 mqsicompcif file, and the equivalent variables in the Version 6.0 BIPEDIT file.
   b. Copy any additional changes that you have made to the environment file, `ENVFILE`, to BIPUPROF in the component data set. Submit the BIPGEN job to create the environment file, `ENVFILE`.
   c. Customize and submit the BIPMGCMP job.
4. Copy the started task JCL (BIPUNSP) to the procedures library. When you copy the started task, keep a second copy of the original in a safe place for backup purposes.
5. The verification program runs when you start the Version 6.0 User Name Server.

When you have migrated the User Name Server, ensure that you have migrated the rest of the components:

- Migrate the Message Brokers Toolkit
- Migrate the Configuration Manager
- Migrate the broker

then see the post-migration tasks for information about tasks that you might want to perform after migration.

Migrating from WebSphere Business Integration Message Broker with Rules and Formatter Extension Version 5.0 to WebSphere Message Broker with Rules and Formatter Extension Version 6.0

To migrate a broker domain from WebSphere Business Integration Message Broker with Rules and Formatter Extension Version 5.0 to WebSphere Message Broker with Rules and Formatter Extension Version 6.0, follow the instructions in "Migrating from WebSphere Business Integration Message Broker Version 5.0 to WebSphere Message Broker Version 6.0" on page 57, replacing all references to ″WebSphere Message Broker″ with ″WebSphere Message Broker with Rules and Formatter Extension″.

For instructions on how to migrate the Rules and Formatter Extension, see the New Era of Networks documentation.

You must migrate the Rules and Formatter extension immediately after migrating from WebSphere Business Integration Message Broker Version 5.0 to WebSphere Message Broker Version 6.0. Do not carry out any operations between migrating WebSphere Message Broker and migrating the Rules and Formatter Extension.

Migrating from WebSphere Business Integration Message Broker with Rules and Formatter Extension Version 5.0 to WebSphere Message Broker Version 6.0

Migrate from WebSphere Business Integration Message Broker with Rules and Formatter Extension Version 5.0 to WebSphere Message Broker Version 6.0 only if you have not used the Rules and Formatter functions in WebSphere Business Integration Message Broker with Rules and Formatter Extension Version 5.0, or you do not want to continue using them after the migration.

If you have deployed any message flows that use the Rules and Formatter function that is available in WebSphere Business Integration Message Broker with Rules and Formatter Extension Version 5.0, remove these message flows from the affected brokers before the brokers are migrated to WebSphere Message Broker Version 6.0.

After you have removed these message flows from the affected brokers, follow the instructions in "Migrating from WebSphere Business Integration Message Broker Version 5.0 to WebSphere Message Broker Version 6.0" on page 57.

Post-migration tasks

After migrating to WebSphere Message Broker Version 6.0 from a previous version, test the broker resources and components to verify that there is no loss or unexpected change of functionality. Some changes in behavior might be caused by defects being fixed between versions.

The following topics describe what to do after migrating from Version 5.0 to Version 6.0:
- "Removing unwanted files and database tables after migration"
- "Setting up a command environment" on page 51

Removing unwanted files and database tables after migration

Before you start

Before you start this task, make sure that the migration was successful and that you no longer need to retain the flexibility to use WebSphere Business Integration Message Broker Version 5.0 code and resources.

To remove unwanted files and database tables after migration, complete the following steps:

1. If it is no longer required, remove WebSphere Business Integration Message Broker Version 5.0 from the computer. Refer to the WebSphere Business Integration Message Broker Version 5.0 information center for instructions about uninstalling the product.
2. If they are no longer required for use with Version 6.0, also remove the prerequisite software products used by Version 5.0, such as DB2 and WebSphere MQ. For instructions, refer to the documentation for those products.
3. If you did not remove DB2, you might want to remove the Configuration Manager DB2 database tables used by Version 5.0.

   In Version 6.0, the Configuration Manager no longer uses an external database to store the domain configuration but uses an internal repository instead. This means that you cannot use your Version 5.0 Configuration Manager DB2 database tables with the Version 6.0 Configuration Manager.

   To remove the Configuration Manager DB2 database tables, drop the following database tables by using the DB2 **DROP TABLE** command. If the Configuration Manager tables are in a DB2 database by themselves, drop the whole database by using the DB2 **DROP DATABASE** command.

   - CACLGROUPS
   - CBROKER
   - CBROKERCEG
   - CCOLLECTIVE
   - CCOLLECTIVECBROKER
   - CDELETE
   - CEG
   - CEGCMSGFLOW
   - CEGCMSGPROJECT
   - CLOG
   - CMSGFLOW
   - CMSGPROJECT
   - CNEIGHBOURS
   - COUTSTANDING
   - CPROXY
   - CPROXYCRESOURCE
   - CSUBSCRIBE
   - CTOPIC
   - CTOPICCTOPIC
   - CTOPOLOGY
   - CTRACE
   - CUUIDLOCKS

Setting up a command environment

After you have installed the product on any of the distributed systems, you must initialize the environment before you can use any runtime component or command. A profile is provided for this purpose.

If appropriate, you can extend the initialization performed by this profile; for example, for broker or user databases, or other products that you want to use within the broker. For example, you might want to configure your broker to run with Java 1.5 instead of the default version Java 1.4.2.

You *must* ensure that you run this profile each time you run one of the administrative commands, or start a component; for example, a broker.

For information on setting up your command and runtime environment on a Windows system, see Command environment: Windows platforms

For information on setting up your command and runtime environment on Linux and UNIX systems, see Command environment: Linux and UNIX systems

Upgrading from WebSphere Event Broker Version 6.0 to WebSphere Message Broker Version 6.0

Before you start

Before you start this task, complete the following steps:
1. Make sure that workbench users have saved all WebSphere Event Broker resources and closed all workbench sessions.
2. Stop the Configuration Manager, User Name Server, and all the brokers.
3. Read about "Coexistence with previous versions and other products" on page 4.

To upgrade a broker domain from WebSphere Event Broker Version 6.0 to WebSphere Message Broker Version 6.0, complete the following steps:
1. On each computer where the workbench or a broker can run, install WebSphere Message Broker over WebSphere Event Broker.
2. Start the Configuration Manager, User Name Server, and all the brokers.

The upgrade is now complete and the broker domain is ready for use.

Restoring migrated components to previous versions

You can restore or revert migrated components to previous versions of WebSphere Message Broker. For some components, you restore them to the previous version and their original state; for other components, you revert them to the previous version, but keep any changes that you have made to them, such as deployed message flows.

You might want to restore or revert components if you encounter a problem after you have migrated your broker domain to Version 6.0. By restoring or reverting components to the previous level, you can fix the problem with Version 6.0. There is no need to revert the entire domain; revert only those components that are causing a problem. For example, if a message flow does not work as expected in Version 6.0, revert that broker until you can resolve the problem.

You can revert components to a previous version only if they were migrated from that version originally. You cannot create a new Version 6.0 broker, then revert it to a previous level. Nor can you create a Version 2.1 broker, migrate it to Version 6.0, then revert it to Version 5.0.

The topics in this section explain how to restore or revert components and resources that you have migrated from Version 5.0 or Version 2.1 products back to their original state:
- “Restoring components and resources to Version 5.0”
- “Restoring components and resources to Version 2.1” on page 75

Restoring components and resources to Version 5.0

This topic explains how to restore components and resources that you have migrated from Version 5.0 products back to their original state.

Restrictions

When you revert a broker from Version 6.0 to Version 5.0, message sets are deleted, so you must redeploy them. You are reminded to do this by warning message BIP8688, which appears when you run the **mqsimigratecomponents** command.

After migration to Version 6.0, if you have deployed any message flows that use new Version 6.0 functions, such as new nodes or new ESQL syntax, those message flows will not start if your broker is reverted to Version 5.0. Other message flows in your broker that do not use new Version 6.0 functions will continue to run.

Version 5.0 and Version 5.1 Message Brokers Toolkit source files use a new format in the Version 6.0 Message Brokers Toolkit. The files are migrated to the new format when you save them using the Version 6.0 Message Brokers Toolkit. After this, you can no longer use the files with the Version 5.0 or Version 5.1 Message Brokers Toolkit. If you have created message flows in Version 6.0, you cannot revert them to Version 5.0. For detailed information, see “Conditions for using migrated resources with previous versions of the Message Brokers Toolkit” on page 6.

Migrating resources back to Version 5.0

The following sections describe how to restore the Message Brokers Toolkit and your runtime components to Version 5.0.

Restoring the Message Brokers Toolkit to Version 5.0

To restore the Message Brokers Toolkit to Version 5.0, perform the following steps:
1. Stop the Version 6.0 Message Brokers Toolkit.
2. Restore the Version 5.0 workspace from the backup that you took before migration.
3. Restart the Version 5.0 Message Brokers Toolkit.

Any changes that you made in the Version 6.0 Message Brokers Toolkit cannot be used in the Version 5.0 Message Brokers Toolkit.

Restoring runtime components to Version 5.0

Use the **-s** and **-t** parameters of the **mqsimigratecomponents** command to migrate components from Version 6.0 to Version 5.0. Specify Version 6.0 for the source version parameter (**-s**) and Version 5.0 for the target version parameter (**-t**). See the **mqsimigratecomponents** command topic for detailed information about these parameters and the format to use when specifying version numbers.

To revert z/OS runtime components to Version 5.0, perform the following steps:

1. Submit the BIPMGCMP job to call the **mqsimigratecomponents** command, specifying the **-s** and **-t** parameters as described above.
2. Replace the started task JCL in USER.PROCLIB with the Version 5.0 copy that you backed up.

Reverting a broker to Version 5.0

To revert a migrated broker to its Version 5.0 state, perform the following steps:

1. Stop the Version 6.0 broker.
2. Revert the broker to Version 5.0 using the **mqsimigratecomponents** command, as shown in the following example:
   ```
   mqsimigratecomponents Broker -t 5.0.0.4
   ```
   Warning message BIP8688 might be displayed, warning you to redeploy your message sets.
3. Reverse the changes that you made to the ODBC definitions when you migrated to Version 6.0.
   - On UNIX, reset the ODBCINI environment variable to point to the previous version of the odbc.ini file.
   - On Windows, use the Control Panel to adjust the ODBC settings.
4. Restart the broker using a Version 5.0 command window.

If you migrate to Version 6.0, deploy a message set to the Version 6.0 broker, then migrate back to Version 5.0, Version 5.0 is unable to recognize the message set that was deployed by Version 6.0. In this case, any message sets that Version 5.0 is unable to use are deleted and a warning message is displayed for each message set, prompting you to redeploy it to Version 5.0 following successful migration.

Reverting a User Name Server to Version 5.0

There are no functional changes in the User Name Server between WebSphere Business Integration Message Broker Version 5.0 and WebSphere Message Broker Version 6.0. To revert a User Name Server to Version 5.0, issue the **mqsimigratecomponents command**, as shown in the following example:

```
mqsimigratecomponents UserNameServer -t 5.0.0.4
```

Restoring a Configuration Manager to Version 5.0

When you migrate a Configuration Manager from Version 5.0 to Version 6.0, the DB2 database is neither changed nor deleted. Version 6.0 does not use DB2 to store data. If you migrate from Version 6.0 back to Version 5.0, the original DB2 database is used again, and if you made any changes after migration to Version 6.0, these changes are not reverted to Version 5.0. As a result, you will lose any domain changes that you made after migration to Version 6.0.

If you deployed any broker configuration changes after migration to Version 6.0, these are lost when you restore the Configuration Manager to Version 5.0. You must rebuild your brokers by deleting them and redeploying them in order to maintain a consistent state. Check carefully that your Configuration Manager works correctly before deploying for the first time after migration to Version 6.0.

If you changed the Configuration Manager's queue manager during migration to Version 6.0, you must keep the new queue manager if you restore using the **mqsimigratecomponents** command. If the new queue manager is on an operating system other than Windows, you cannot use the **mqsimigratecomponents** command to restore; instead, you must restore from a backup.

Here is an example of the command to restore the Configuration Manager to Version 5.0 state:

```
mqsimigratecomponents ConfigMgr -t 5.0.0.4
```

Restoring components and resources to Version 2.1

This topic explains how to restore components and resources that you have migrated from Version 2.1 products back to their original state.

Restrictions

You can restore resources to Version 2.1 only on operating systems that were supported by Version 2.1. You cannot restore resources to Version 2.1 state on the following operating systems:

- Linux Intel
- Linux s390
- Linux PPC
- Solaris x86_64

If you have used the following features in Version 6.0, restoring resources to Version 2.1 will result in loss of data:

- Multicast
- MQe nodes
- Quality of Protection (QOP)

When you revert a broker from Version 6.0 to Version 2.1, message sets are deleted, so you must redeploy them. You are reminded to do this by warning message BIP8688, which appears when you run the **mqsimigratecomponents** command.

When you revert a broker from Version 6.0 to Version 2.1 on z/OS, retained publications are deleted.

After migration to Version 6.0, if you have deployed any message flows that use new Version 6.0 functions, such as new nodes or new ESQL syntax, those message flows will not start if your broker is reverted to Version 2.1. Other message flows in your broker that do not use new Version 6.0 functions will continue to run.

Due to coexistence requirements, you can restore the Configuration Manager to Version 2.1 only if you have not migrated any brokers to Version 6.0. If you restore the Configuration Manager to Version 2.1, you must also restore the Message Brokers Toolkit to Version 2.1.

Restoring resources to Version 2.1

The following sections describe how to restore the Message Brokers Toolkit and runtime components to Version 2.1.

Restoring the Message Brokers Toolkit to Version 2.1

If you restore the Message Brokers Toolkit to Version 2.1, you must also restore the Configuration Manager to Version 2.1 at the same time. Any changes that you made after migration to Version 6.0 will be lost.

1. Migrate the Configuration Manager back to Version 2.1.
2. Restart the Version 2.1 Message Brokers Toolkit.

Restoring runtime components to Version 2.1

Use the **-s** and **-t** parameters of the **mqsimigratecomponents** command to migrate components from Version 6.0 to Version 2.1. Specify Version 6.0 for the source version parameter (**-s**) and Version 2.1.0.8 for the target version parameter (**-t**). See the **mqsimigratecomponents** command topic for detailed information about these parameters and the format to use when specifying version numbers.

To revert z/OS runtime components to Version 2.1, perform the following steps:

1. Submit the BIPMGCMP job to call the **mqsimigratecomponents** command, specifying the **-s** and **-t** parameters as described above.
2. Replace the started task JCL in USER.PROCLIB with the Version 2.1 copy that you backed up.

Reverting a broker to Version 2.1

To revert a migrated broker to its Version 2.1 state, perform the following steps:

1. Stop the Version 6.0 broker.
2. Revert the broker to Version 2.1 using the **mqsimigratecomponents** command, as shown in the following example:
   ```
   mqsimigratecomponents Broker -t 2.1.0.8
   ```
   Warning message BIP8688 might be displayed, warning you to redeploy your message sets.
3. Reverse the changes that you made to the ODBC definitions when you migrated to Version 6.0.
   - On UNIX, reset the ODBCINI environment variable to point to the previous version of the odbc.ini file.
   - On Windows, use the Control Panel to adjust the ODBC settings.
4. Restart the broker.

If you migrate to Version 6.0, deploy a message set to the Version 6.0 broker, then migrate back to Version 2.1, Version 2.1 is unable to recognize the message set that was deployed by Version 6.0. In this case, any message sets that Version 2.1 is unable to use are deleted and a warning message is displayed for each message set, prompting you to redeploy it to Version 2.1 following successful migration.

Reverting a User Name Server to Version 2.1

There are no functional changes in the User Name Server between WebSphere MQ Integrator Broker Version 2.1 and WebSphere Message Broker Version 6.0. To revert a User Name Server to Version 2.1, issue the **mqsimigratecomponents** command, as shown in the following example:

```
mqsimigratecomponents UserNameServer -t 2.1.0.8
```

Restoring a Configuration Manager to Version 2.1

If you restore the Configuration Manager to Version 2.1, you must also restore the Message Brokers Toolkit and any brokers to Version 2.1.

The Version 6.0 Configuration Manager does not use the DB2 database that was used by the Version 2.1 Configuration Manager; if you restore to Version 2.1, the original database is used again. This means that any configuration changes that you made after you migrated the Configuration Manager to Version 6.0 will be lost. This can leave your domain in an inconsistent state, forcing you to rebuild and redeploy all of your brokers in order to regain a consistent state. After you have migrated your Configuration Manager to Version 6.0, check carefully that the Message Brokers Toolkit and Configuration Manager are working correctly before you attempt any further configuration changes or before you migrate any brokers.

If you have moved the Configuration Manager's queue manager during migration, you must keep the new queue manager when you restore, unless you restore your entire domain from a backup. If you have moved the Configuration Manager onto an operating system other than Windows, you cannot restore it to Version 2.1 unless you restore your domain from a backup.

Given these warnings, you can restore a Version 6.0 Configuration Manager to Version 2.1 using the command in the following example:

```
mqsimigratecomponents ConfigMgr -t 2.1.0.8
```

Your Version 6.0 Message Brokers Toolkit will no longer work; you must use a Version 2.1 Message Brokers Toolkit instead.

Migrating publish/subscribe applications

If you are already using WebSphere MQ Publish/Subscribe, you can migrate your applications to use the publish/subscribe functions provided by WebSphere Message Broker.

For more information, refer to the following topics:

- "Planning for migration" on page 78
- "Running two independent broker networks" on page 78
- "Creating and operating a heterogeneous network" on page 79
- "Migrating WebSphere MQ brokers" on page 85
- "Migrating a WebSphere MQ broker network" on page 88

Planning for migration

If you are already using WebSphere MQ Publish/Subscribe, you can migrate your applications to use the publish/subscribe functions provided by WebSphere Message Broker.

You can also migrate individual WebSphere MQ Publish/Subscribe brokers to create replacement WebSphere Message Broker brokers, with support for their client applications remaining intact.

These two possibilities offer you a number of advantages:

- Publications from within the WebSphere MQ Publish/Subscribe network can be targeted by WebSphere Message Broker subscribers. This includes messages originating in environments not yet supported by WebSphere Message Broker.
- Message flows can be created and deployed on WebSphere Message Broker brokers to:
  - Analyze the information that is flowing around your enterprise.
  - Create and execute additional business logic dependent upon the content of the publications.
  - Consolidate the information within your enterprise in the form of new publications, that can then be republished as a series of additional topics available to both WebSphere Message Broker and WebSphere MQ Publish/Subscribe clients.

There are three possible scenarios for exploiting the two networks:

1. You can choose to have two independent broker networks, and therefore have two separate broker domains for publish/subscribe applications. See "Running two independent broker networks."
2. You can connect the two networks to allow publications and subscriptions to flow throughout the integrated network. Further details are provided in "Creating and operating a heterogeneous network" on page 79.
3. You can selectively and gradually migrate individual brokers from WebSphere MQ Publish/Subscribe to WebSphere Message Broker. For more guidance on this option, see "Migrating WebSphere MQ brokers" on page 85 and "Migrating a WebSphere MQ broker network" on page 88.

Before you can make this choice, and create your migration plan, you must be aware of the differences between the two products. These differences are described in WebSphere MQ Publish/Subscribe.

Running two independent broker networks

If you already have an WebSphere MQ Publish/Subscribe broker network, you can continue to use this network unchanged. The introduction of WebSphere Message Broker Version 6.0 to your environment, and the creation of brokers in that broker domain, does not affect your WebSphere MQ Publish/Subscribe broker domain until you take specific action to connect the two networks.

If you want to run in this mode with two separate, independent networks, you do not have to take any specific actions. You can retain your existing WebSphere MQ Publish/Subscribe network, and install and configure a WebSphere Message Broker Version 6.0 network, without any interaction.

Your existing applications can continue to work unchanged. However, there can be no interchange of publications in this scenario.

You must be aware that a single queue manager cannot support both a WebSphere MQ Publish/Subscribe broker and a WebSphere Message Broker Version 6.0 broker. If you have brokers of both types on the same system, each broker must have its own dedicated queue manager.

You can implement this scenario while you assess the new product and the extra functions contained within the publish/subscribe support. It also lets you plan for the extent of integration or migration, or both, that you require, without affecting your current environment.

However, you must ensure that you have the required level of WebSphere MQ for your WebSphere MQ Publish/Subscribe system. WebSphere Message Broker Version 6.0 requires WebSphere MQ to be at least Version 5.3.

Creating and operating a heterogeneous network

You can integrate your existing WebSphere MQ Publish/Subscribe broker network with a WebSphere Message Broker broker network to create a mixed, heterogeneous network. This enables publications and subscriptions to be propagated through one logical network, made up of two or more physical networks. As a result, subscribers to the WebSphere MQ Publish/Subscribe brokers can target information being published to the WebSphere Message Broker broker network, and subscribers to the WebSphere Message Broker brokers can target information being published to the WebSphere MQ Publish/Subscribe broker network.

In a heterogeneous network, any new stream that is created for a WebSphere MQ Publish/Subscribe broker is supported by any WebSphere MQ Publish/Subscribe brokers that are connected to it. However, the new stream is not supported by WebSphere MQ Publish/Subscribe brokers that are separated by a WebSphere Message Broker broker from the WebSphere MQ Publish/Subscribe broker where the new stream was created.

WebSphere MQ Publish/Subscribe brokers and WebSphere Message Broker brokers can be running on Windows or other platforms.

There are two ways in which a WebSphere Message Broker broker can be joined to the WebSphere MQ Publish/Subscribe network; it can be joined either as a leaf node (that is, as a child of an existing WebSphere MQ Publish/Subscribe broker) or as a parent node (that is, as the parent of an existing WebSphere MQ Publish/Subscribe broker).

Every WebSphere Message Broker broker that you integrate into an WebSphere MQ Publish/Subscribe network must have a minimum of two queues available:

- SYSTEM.BROKER.DEFAULT.STREAM

  This queue supports the default publication stream. You must create this queue on every broker. You must also create and deploy a message flow that services this stream queue.
- SYSTEM.BROKER.INTER.BROKER.COMMUNICATIONS

  This queue is used by the broker to communicate with neighboring WebSphere MQ Publish/Subscribe brokers. You must create this queue on every broker, however the message flow for this queue is created internally by the broker.

Adding a broker as a leaf node

The steps described in this topic assume that you are joining:

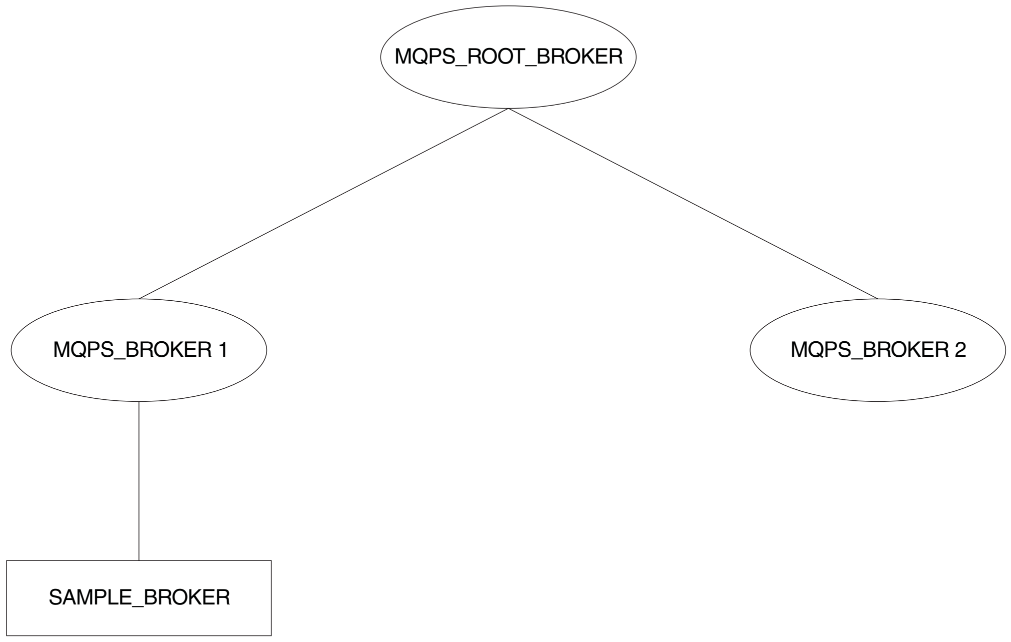
- A WebSphere Message Broker broker named WBRK_BROKER. When this broker was created, the same name was specified for the queue manager.
- A WebSphere MQ Publish/Subscribe broker network with a root broker MQPS_ROOT_BROKER, and two leaf brokers MQPS_BROKER1 and MQPS_BROKER2.

Substitute the names of your brokers for these examples wherever they are used.

All commands shown must be issued on the system on which the appropriate resource is defined. WebSphere MQ commands (for example, the command to define a queue) are shown in MQSC format. For more information about WebSphere MQ commands, refer to your WebSphere MQ documentation.

The following steps describe what you should do to add, as a leaf node within your WebSphere MQ Publish/Subscribe broker network, a WebSphere Message Broker broker that you have already created.

This is shown in the following diagram. WebSphere Message Broker broker **WBRK_BROKER** is joined to the WebSphere MQ Publish/Subscribe network, with broker **MQPS_BROKER1** as its parent broker.



1. Ensure that the WebSphere Message Broker broker's default execution group is successfully deployed. This execution group is deployed the first time you deploy a newly created WebSphere Message Broker broker. You can check the status of both the execution group and the broker from the topology view in the workbench.
2. Define the queue required to support interbroker communications with WebSphere MQ Publish/Subscribe neighbors on the WebSphere Message Broker broker's queue manager:

```
define qlocal(SYSTEM.BROKER.INTER.BROKER.COMMUNICATIONS) noshare
```

3. Stop the WebSphere Message Broker broker:

   ```
   mqsistop WBRK_BROKER
   ```

4. Restart the WebSphere Message Broker broker:

   ```
   mqsistart WBRK_BROKER
   ```

   When the WebSphere Message Broker broker is restarted, the presence of the interbroker queue (defined above) enables the broker to receive and process messages on this queue.

5. Create the resources required on the WebSphere Message Broker broker to support the default WebSphere MQ Publish/Subscribe stream:

   a. Create the default stream queue:

      ```
      define qlocal(SYSTEM.BROKER.DEFAULT.STREAM) noshare
      ```

   b. Create a message flow based on the supplied publish/subscribe message flow:

      1) Start up the workbench and select the designer view.
      2) If you have not already imported and saved the default message flows supplied, import these now. This enables you to reuse the default publish/subscribe flow here. Click **File** → **Import** and open the file, called `SamplesWorkspaceForImport`, in the `examples` subdirectory within the WebSphere Message Broker home directory. This might take a few minutes to complete.

         If you prefer, you can create your own message flow.
      3) Make a copy of the supplied message flow and rename it.
      4) Check the properties of the nodes in the message flow. You must set the appropriate input (stream) queue property for the MQInput node. Check that the other properties of the nodes are set correctly for your requirements.
      5) Finally, check in your changes and deploy the message flow to the default execution group of the broker WBRK_BROKER.

      You can find full details of how to complete these steps in the online help for the workbench.

6. If you are using additional streams in the WebSphere MQ Publish/Subscribe network, you must also enable these on the WebSphere Message Broker broker. Although the WebSphere Message Broker broker is able to support all the streams of its WebSphere MQ Publish/Subscribe neighbors, you need only define queues, and define and deploy message flows, for those streams requested by WebSphere Message Broker subscriber clients.

   a. Create a local queue on the WebSphere Message Broker broker's queue manager for each stream on which messages are to be processed. For example:

      ```
      define qlocal(STREAM.X) noshare
      ```

   b. Create and deploy a message flow to read and process the WebSphere MQ Publish/Subscribe messages that are sent to each stream (publication) queue.

      You can use the supplied publish/subscribe message flow as the basis for each new message flow. Each MQInput node representing a non-default stream must have the property *implicitStreamNaming* set (this is the default setting).

7. Ensure that the WebSphere MQ Publish/Subscribe broker is running. If it is not, you can start it using the start command:

   ```
   strmqbrk MQPS_BROKER1
   ```

8. Ensure that the WebSphere MQ connection between the two brokers is enabled; you must start the listeners for the receiver channels, and you must then start the sender channels.
9. Join the WebSphere Message Broker broker to the WebSphere MQ Publish/Subscribe network as a child of the WebSphere MQ Publish/Subscribe broker:
   ```
   mqsijoinmqpubsub WBRK_BROKER -p MQPS_BROKER1
   ```
10. Verify the success of the join command to ensure that the WebSphere MQ Publish/Subscribe broker is an active neighbor:
    ```
    mqsilistmqpubsub WBRK_BROKER
    ```
    If the join command has completed successfully, you see a response to the list command that is similar to:
    ```
    BIP8090I: WebSphere MQ Publish/Subscribe neighbor WBRK_BROKER
          is active

    BIP8091I: Common stream    SYSTEM.BROKER.DEFAULT.STREAM
    BIP8091I: Common stream    STREAM.X
    ```

Adding a broker as a parent node

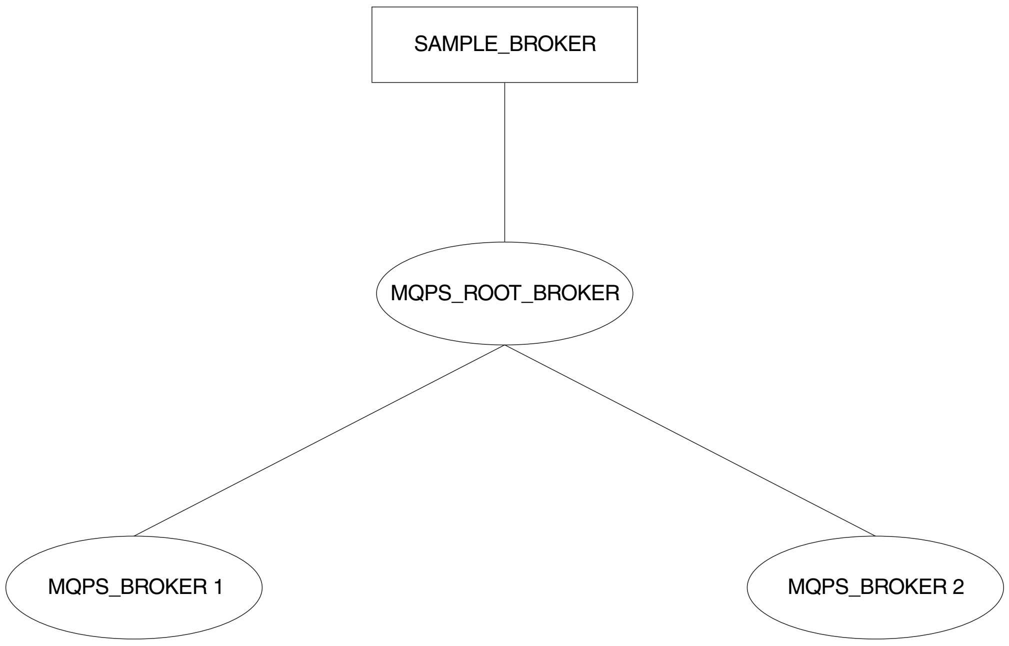
The steps described in this topic assume that you are joining:
- A WebSphere Message Broker broker named WBRK_BROKER. When this broker was created, the same name was specified for the queue manager.
- A WebSphere MQ Publish/Subscribe broker network with a root broker MQPS_ROOT_BROKER, and two leaf brokers MQPS_BROKER1 and MQPS_BROKER2.

Substitute the names of your brokers for these examples wherever they are used.

All commands shown must be issued on the system on which the appropriate resource is defined. WebSphere MQ commands (for example, the command to define a queue) are shown in MQSC format. For more information about WebSphere MQ commands, refer to your WebSphere MQ documentation.

The following steps describe what you should do to add, as a parent node within your WebSphere MQ Publish/Subscribe broker network, a WebSphere Message Broker broker that you have already created.

This is shown in the following diagram. WebSphere Message Broker broker **SAMPLE_BROKER** is joined to the WebSphere MQ Publish/Subscribe network as the new parent node (that is, as the parent of the original parent node **MQPS_ROOT_BROKER**).



1. Ensure that the WebSphere Message Broker broker's default execution group is successfully deployed. This execution group is deployed the first time you deploy a newly created WebSphere Message Broker broker. You can check the status of both the execution group and the broker from the topology view in the workbench.
2. Define the queue required to support interbroker communications with WebSphere MQ Publish/Subscribe neighbors on the WebSphere Message Broker broker's queue manager:

   ```
   define qlocal(SYSTEM.BROKER.INTER.BROKER.COMMUNICATIONS) noshare
   ```
3. Stop the WebSphere Message Broker broker:

   ```
   mqsistop SAMPLE_BROKER
   ```
4. Restart the WebSphere Message Broker broker:

   ```
   mqsistart SAMPLE_BROKER
   ```

   When the WebSphere Message Broker broker is restarted, it is enabled to receive and process messages on the interbroker queue.
5. Create the resources required on the WebSphere Message Broker broker to support the default WebSphere MQ Publish/Subscribe stream:
   a. Create the default stream queue:

      ```
      define qlocal(SYSTEM.BROKER.DEFAULT.STREAM) noshare
      ```
   b. Create a message flow for publish/subscribe, either your own, or one based on the supplied publish/subscribe message flow:
      1) Start up the workbench and select the designer view.
      2) Make a copy of the supplied message flow and rename it; you must import this default message flow before you can access and use it.
      3) Check the properties of the nodes in the message flow. You must set the appropriate input (stream) queue property for the MQInput node. Check that the other properties of the nodes are set correctly for your requirements.
      4) Finally, deploy the message flow to the default execution group of the broker SAMPLE_BROKER.

You can find full details of how to complete these steps in the online help for the workbench.

6. If you are using additional streams in the WebSphere MQ Publish/Subscribe network, you must also enable these on the WebSphere Message Broker broker. Although the WebSphere Message Broker broker is able to support all the streams of its WebSphere MQ Publish/Subscribe neighbors, you need only define queues, and define and deploy message flows, for those streams requested by WebSphere Message Broker subscriber clients.
   a. Create a local queue on the WebSphere Message Broker broker's queue manager for each stream on which messages are to be processed. For example:
      ```
      define qlocal(STREAM.X) noshare
      ```
   b. Create and deploy a message flow to read and process the WebSphere MQ Publish/Subscribe messages that are sent to each stream (publication) queue.

      You can use the supplied publish/subscribe message flow as the basis for each new message flow. Each MQInput node representing a non-default stream must have the property *implicitStreamNaming* set.
7. Enter the following WebSphere MQ Publish/Subscribe command against the broker that is the current WebSphere MQ Publish/Subscribe parent broker, to terminate its activities:
   ```
   endmqbrk -c -m MQPS_ROOT_BROKER
   ```
   This requests a controlled shutdown (-c). When the shutdown has completed, the broker can be restarted. You can request an immediate shutdown, by specifying -iinstead of -c, if you need to force this shutdown to complete.
8. Ensure that the WebSphere MQ connection between the two brokers is active; you must start the listeners for the receiver channels, and then start the sender channels.
9. Enter the following WebSphere MQ Publish/Subscribe command against the current WebSphere MQ Publish/Subscribe parent broker to restart it:
   ```
   strmqbrk -m MQPS_ROOT_BROKER -p SAMPLE_BROKER
   ```
   If the queue manager associated with the WebSphere Message Broker broker SAMPLE_BROKER has not been created with the same name as the broker, you must specify the queue manager name here after the `-p` flag, not the broker name.
10. Verify the success of the integration:
    ```
    mqsilistmqpubsub SAMPLE_BROKER
    ```
    If the WebSphere Message Broker broker has been integrated into the WebSphere MQ Publish/Subscribe network successfully, you see a response that is like the following:
    ```
    BIP8090I: WebSphere MQ Publish/Subscribe neighbor SAMPLE_BROKER
       is active

    BIP8091I: Common stream    SYSTEM.BROKER.DEFAULT.STREAM
    BIP8091I: Common stream    STREAM.X
    ```

Deleting brokers

If you have a mixed broker network, you must take particular care to maintain the integrity of the network if you need to remove or delete a broker from the network:

- When you issue the mqsideletebroker command to delete a WebSphere Message Broker broker, the WebSphere MQ Publish/Subscribe brokers that are neighbors of this WebSphere Message Broker broker are not automatically informed of its deletion. Remove the WebSphere Message Broker broker from the network using the clear commands mqsiclearmqpubsub (at the WebSphere Message Broker broker) and clrmqbrk (at its WebSphere MQ Publish/Subscribe neighbors) before you delete it.

- If you delete a WebSphere Message Broker broker before you remove it from the network, and it has a parent WebSphere MQ Publish/Subscribe broker, the parent broker continues to attempt to send publication and subscription messages to it. You can correct this behavior by issuing the clrmqbrk command at the parent. For example, if you issue:

  ```
  mqsideletebroker -m WBRK_CHILD_BROKER
  ```

  while the WebSphere Message Broker broker is still known to its parent WebSphere MQ Publish/Subscribe broker, you can then issue the command

  ```
  clrmqbrk -m MQPS_PARENT_BROKER -c WBRK_CHILD_BROKER
  ```

  to the parent broker to clean up the network.
- When you issue the dltmqbrk command to delete an WebSphere MQ Publish/Subscribe broker that is a child of a WebSphere Message Broker broker, the WebSphere Message Broker broker receives notification of the deletion.

  Therefore you do not have to issue the mqsiclearmqpubsub command to remove knowledge of the deleted child at the WebSphere Message Broker parent broker.

  For example, if you want to delete the child broker MQPS_CHILD_BROKER you must issue the following single command:

  ```
  dltmqbrk -m MQPS_CHILD_BROKER
  ```

You are prevented from deleting a WebSphere MQ Publish/Subscribe broker that is a parent of any broker; the dltmqbrk command fails.

Migrating WebSphere MQ brokers

Migrating WebSphere MQ brokers

When you plan the migration of one or more WebSphere MQ brokers, you must take account of the differences between WebSphere MQ Publish/Subscribe and WebSphere Message Broker. This might mean that you have to make some changes to your applications, your topics, or both, before you start migration.

The information here tells you the steps you must take to migrate a single broker. "Migrating a WebSphere MQ broker network" on page 88 tells you how to migrate a WebSphere MQ broker network.

These steps result in the replacement of the WebSphere MQ brokers by WebSphere Message Broker brokers.

Each replacement WebSphere Message Broker broker must be created on the same queue manager as the WebSphere MQ broker that it replaces. Because the WebSphere MQ broker shares the same name as the queue manager that supports it, you must specify the WebSphere MQ broker name as the queue manager parameter on the mqsicreatebroker command (the `-q` flag).

Migration involves the transfer of the following state information from the WebSphere MQ broker to the WebSphere Message Broker broker:

- Subscriptions.

  All client subscriptions are exported from all streams except SYSTEM.BROKER.ADMIN.STREAM.
- Retained publications.

All retained publications in MQRFH format are exported from all streams except SYSTEM.BROKER.ADMIN.STREAM.

- Local publishers.

  Registrations for all publishers that are producing local publications are exported from all streams except SYSTEM.BROKER.ADMIN.STREAM.
- Related brokers.

  If the broker is part of a multi-broker hierarchy, details of all its relations are exported. This includes the names of all streams that the broker to be migrated has in common with the relation.
- Streams.

  On WebSphere MQ Publish/Subscribe, streams are the queues from which publications are read by a broker.

This information is exported as a series of messages that are sent from the WebSphere MQ broker to its replacement. When migration is complete, the WebSphere MQ broker is deleted automatically, and cannot be recreated.

The workbench and migration

If you are migrating a WebSphere MQ broker, you cannot fully deploy it in your WebSphere Message Broker broker domain until migration has completed successfully. You should not deploy additional execution groups or message flows until after you have successfully migrated the WebSphere MQ Publish/Subscribe broker.

Use the *Broker Topology* editor to define the WebSphere Message Broker broker, and deploy the topology. Create an empty .bar file and drag it onto the default execution group. You are now ready to start the migration.

If migration fails, and you want to revert to your WebSphere MQ broker, you must delete the WebSphere Message Broker. See Deleting a broker.

Migrating a single broker

When you migrate a WebSphere MQ broker that is not part of a network, you are replacing it in the network and assigning all the function that was previously supported by that broker to a WebSphere Message Broker broker.

You must shutdown the WebSphere MQ broker before you start migration, and ensure that all applications that use the broker are also quiesced.

Preparing for the migration

Before you can migrate a broker, you need to do some preparation.

1. Identify the WebSphere MQ broker that you are going to migrate.

   The steps used here assume you have chosen the name WBRK_BROKER for your new WebSphere Message Broker broker, and that the WebSphere MQ broker you are migrating is currently hosted by the queue manager MQPS_BROKER1.
2. Back up the queue manager hosting the WebSphere MQ broker.

   Ensure that this backup is complete backup before you start the migration process. This allows you to retrieve the old WebSphere MQ broker after successful migration, if you should need to do so for any reason. The *WebSphere MQ System Administration* book describes this backup process.

3. Quiesce any applications that are registered with the broker.

   Any messages generated during the migration exercise are queued and could cause performance or capacity problems. Quiescing the applications as well as the broker ensures that publish/subscribe traffic is only generated when there is a broker ready to process it.
4. End your WebSphere MQ broker operation:
   ```
   endmqbrk MQPS_BROKER1
   ```

Preparing the replacement broker

You are now ready to work with the new broker.

1. Create a WebSphere Message Broker broker.

   You must create the new broker on the system on which the queue manager MQPS_BROKER1 is defined. You must select the migration option (flag `-m`) on the command.
   ```
   mqsicreatebroker WBRK_BROKER -q MQPS_BROKER1
   -i mqbroker -a sample -n WBRKBKDB -m
   ```
2. Start the new WebSphere Message Broker broker:
   ```
   mqsistart WBRK_BROKER
   ```
3. Configure the broker in the workbench.

   Create the new broker in the broker domain topology from the *Topology* view in the workbench. Save and deploy the topology. Create a default execution group and drag an empty .bar file onto the default execution group.

Migrating the WebSphere MQ broker

The new WebSphere Message Broker broker is ready to receive migration data for the WebSphere MQ broker that it replaces.

1. Migrate the WebSphere MQ broker function to the replacement WebSphere Message Broker broker by issuing the following command:
   ```
   migmqbrk -m MQPS_BROKER1
   ```
   This command is supplied as part of the WebSphere MQ Publish/Subscribe package on the Web. You must ensure you have the latest level of this command, and the *WebSphere MQ Publish/Subscribe User's Guide* that describes its use.

   The command retrieves the persistent information (subscriptions and retained publications) from the WebSphere MQ broker, and sends it in specially constructed messages to the queue SYSTEM.BROKER.INTERBROKER.QUEUE on the new WebSphere Message Broker broker.

   The message flow that services this queue (deployed when you deployed the broker and its default execution group) receives these messages and records the information. When all the messages have been processed, the message flow is terminated and cannot be re-initialized.

   The migration command can only be re-invoked if the whole process of migration has not completed successfully. If any error occurs, for any reason, the WebSphere MQ broker is recoverable and can be restarted. You can then continue to use it. The WebSphere Message Broker broker also exists, but has not recorded any migration information. You can delete and create this broker to restart the migration process.

   If the whole process succeeds, the WebSphere MQ broker no longer exists and cannot be recovered.

   You receive the following message on successful completion of migration:

```
WebSphere MQ broker has been successfully migrated
```

When you have successfully migrated all the WebSphere MQ brokers that you plan to migrate, delete or rename the file `strmqbrk.exe`. This prevents any WebSphere MQ brokers from starting accidentally.

Deploying the stream queues

The new WebSphere Message Broker broker is now set up to take over from the WebSphere MQ broker. You must create and deploy the message flows that it needs to activate the streams; you do not need to define the stream queues, because these are already defined to the queue manager. The queue definitions are not deleted when the migration takes place, and the same queue manager is used by the WebSphere MQ broker and the WebSphere Message Broker broker that has replaced it.

You can create the message flows you need by following these steps:

1. Start the workbench and select the message flow view.
2. For each stream, including the default stream:
   a. Build a basic publish/subscribe message flow by copying and renaming the supplied publish/subscribe message flow.
   b. Check the properties of the nodes in each message flow that you create.

      You must set the input queue name (the stream queue) property in the input node. You must also set the implicitStreamNaming property for every non-default stream queue input node.
   c. Finally, assign the message flow to an execution group of the broker WBRK_BROKER, check in your changes, and deploy the broker.

Migrating a WebSphere MQ broker network

The procedure that you must follow to migrate a WebSphere MQ broker that is part of a multi-broker network is basically the same as that needed to migrate a single broker.

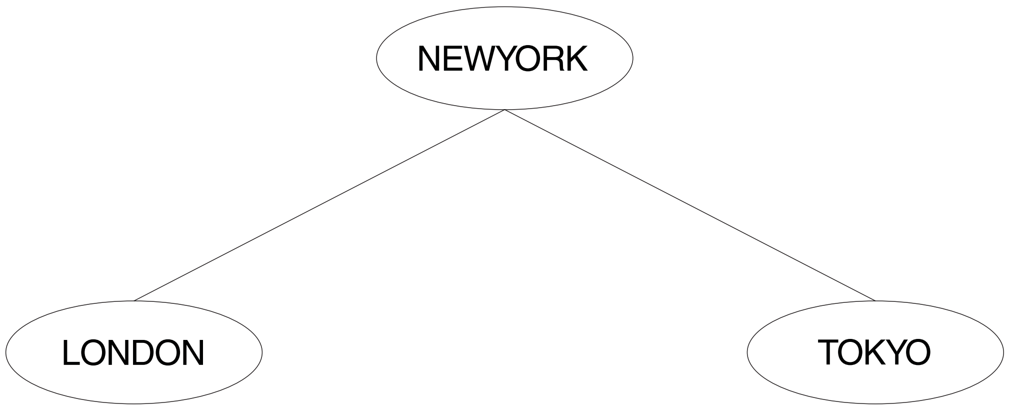
Before you start the migration, you must consider:

- The order in which you migrate the brokers.
- The place of each broker in the network, and the relationships it has with its neighbors.
- The use of collectives in the WebSphere Message Broker network.

Refer to “Planning for migration” on page 78 for more information.

The following sequence of figures illustrates the migration of a network of three brokers. The actions taken to migrate the network assumes that the three brokers are migrated one at a time, and that all three are to be grouped in a single collective in the WebSphere Message Broker broker domain.

The WebSphere MQ broker network that is to be migrated has three brokers, the root (NEWYORK) and two children (LONDON and TOKYO).



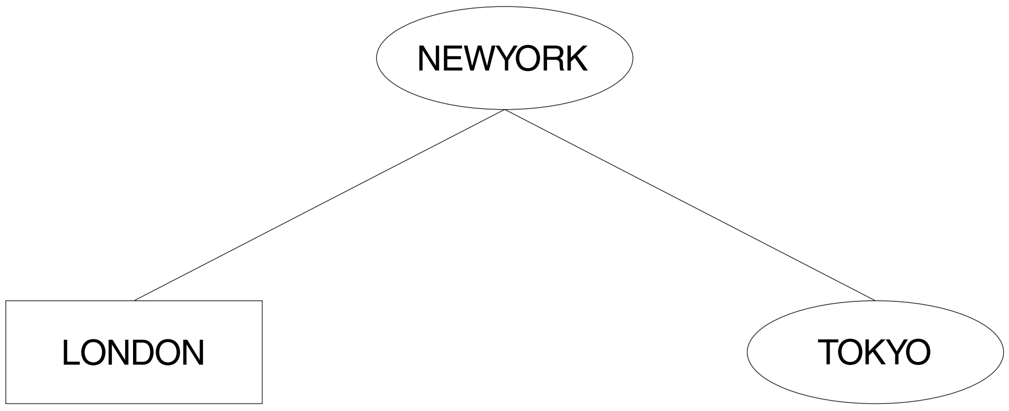
These brokers do not have to be migrated in any particular order. This example shows the migration being done in the following order:
1. LONDON
2. NEWYORK
3. TOKYO

The migration is completed in a number of separate steps. Each step is best taken when network traffic is low (for example, at weekends). The whole migration is planned in three stages:

Stage 1: migration of the LONDON broker

The steps that you need to take to migrating a single broker within a network are exactly the same as those that you need to take to migrate a standalone WebSphere MQ broker. See "Migrating WebSphere MQ brokers" on page 85.
1. Quiesce all client applications at both the LONDON and NEWYORK brokers. This ensures that no publications are missed by any subscribers while the topology is being changed.
2. Quiesce all other brokers in the network (in this example, the TOKYO broker). This guarantees that no publications are delivered while the topology is being changed.

After the migration of the LONDON broker you have a mixed network, consisting of two WebSphere MQ brokers (NEWYORK and TOKYO) and one WebSphere Message Broker broker:



The connection between the LONDON and NEWYORK brokers is a WebSphere MQ connection. The workbench only recognizes WebSphere Message Broker brokers, and therefore only LONDON has been defined to it. A WebSphere Message Broker connection cannot be created at this stage.

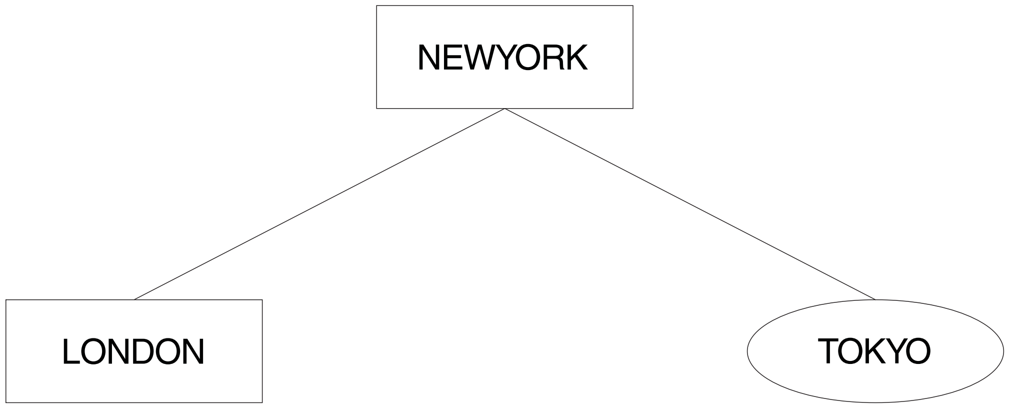
This mixed network is in a perfectly valid state. It can remain in this state until you are ready to do the next stage of the migration.

Stage 2: migration of the NEWYORK broker

Follow the step-by-step procedure for migrating a single broker for broker NEWYORK. This is described in "Migrating WebSphere MQ brokers" on page 85.

1. Quiesce all client applications at all brokers to which NEWYORK is a neighbor; in this network, LONDON and TOKYO. This ensures that no publications are missed by any subscribers while the topology is being changed.
2. Quiesce all the brokers in the network. This guarantees that no publications are delivered while the topology is being changed.

The network now contains two WebSphere Message Broker brokers (LONDON and NEWYORK), and one WebSphere MQ broker (TOKYO):



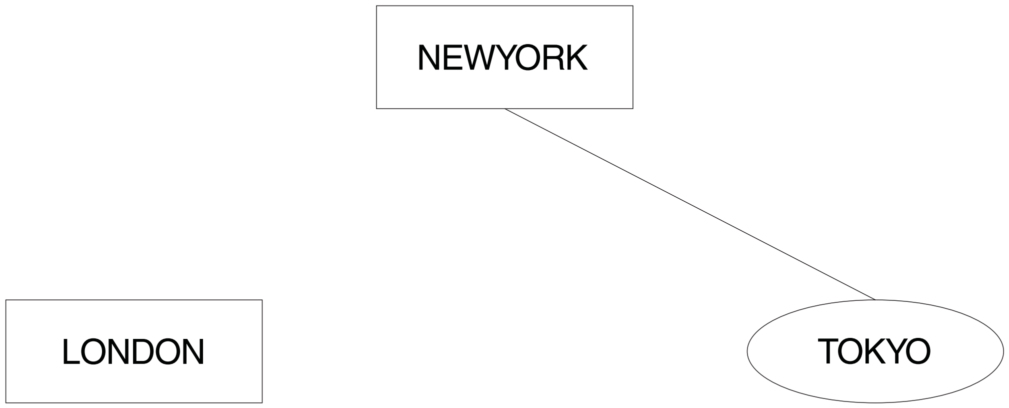
The LONDON and NEWYORK brokers are still connected by a WebSphere MQ connection. They can remain connected in this way for as long as necessary. However, to develop applications that use the functions provided by WebSphere Message Broker you must join the two WebSphere Message Broker brokers together using the workbench.

The connection can be upgraded to a WebSphere Message Broker connection by first removing the original WebSphere MQ connection between LONDON and NEWYORK.

To remove this connection, issue the WebSphere Message Broker command mqsiclearmqpubsub at both brokers:

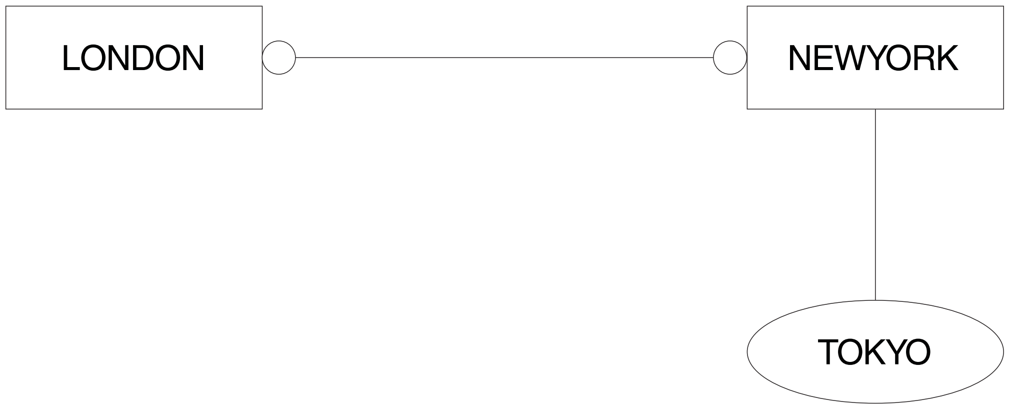
```
mqsiclearmqpubsub NEWYORK -n LONDON
mqsiclearmqpubsub LONDON -n NEWYORK
```

The network now looks like this:



Now use the workbench to define the relationship between the two brokers, LONDON and NEWYORK. Both brokers are already defined, but the collective to which they are to be assigned is not. You can define this collective from the *Topology* view, and assign the two brokers to it. All brokers in a collective are assumed to be connected, so you do not have to make those connections using the workbench.

The new topology can now be deployed. The connection between LONDON and NEWYORK is now implemented using WebSphere Message Broker functions. The network is now:



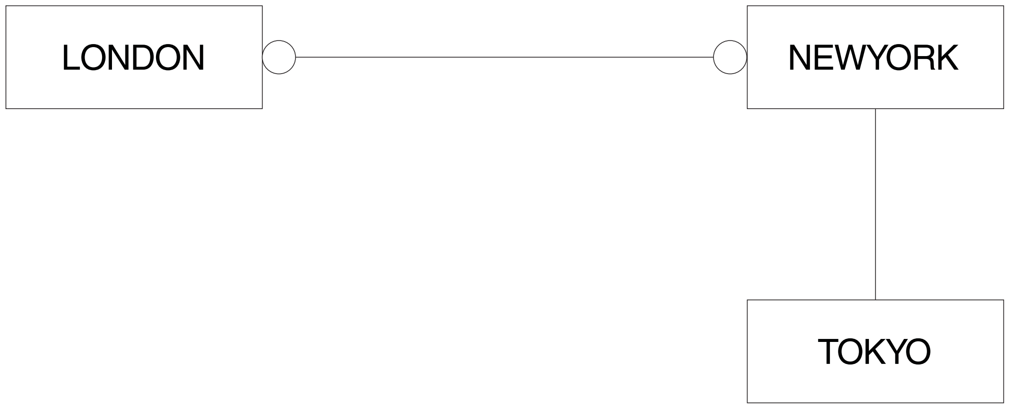
The two brokers, LONDON and NEWYORK, are no longer in a parent-child relationship but are neighbors within a collective. The topology of the WebSphere Message Broker network is not based on a hierarchical structure as was the WebSphere MQ network.

Now that LONDON and NEWYORK form a collective, there is no root node left in the WebSphere MQ network. NEWYORK is the gateway between the WebSphere MQ broker (TOKYO) and the WebSphere Message Broker collective of brokers.

Stage 3: migration of the TOKYO broker

The final WebSphere MQ broker, TOKYO, is now ready to be migrated. Follow the procedure described in “Migrating WebSphere MQ brokers” on page 85.

The network is now:



The WebSphere MQ connection between TOKYO and NEWYORK can now be broken. This is done by using the following commands:

```
mqsiclearmqpubsub NEWYORK -n TOKYO
mqsiclearmqpubsub TOKYO -n NEWYORK
```

Now use the workbench to add the TOKYO broker to the WebSphere Message Broker network, and to the collective. The operation of a collective requires that all brokers have direct physical connections with each other (via WebSphere MQ).

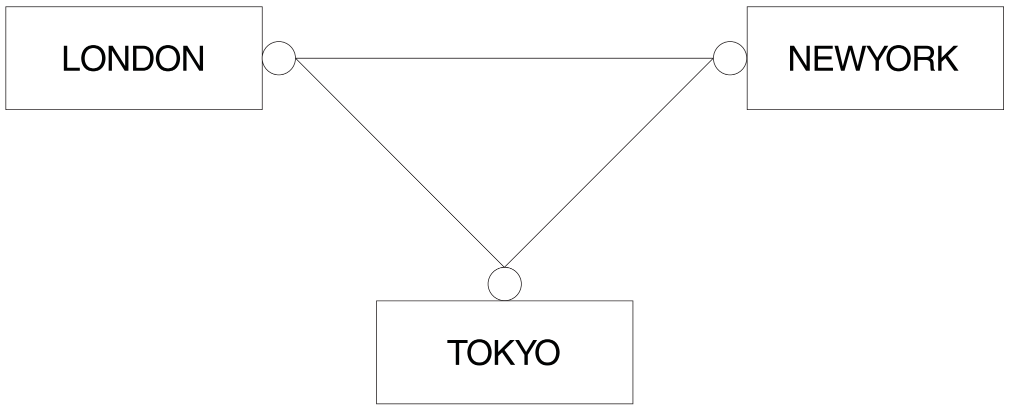
Before the topology of the new WebSphere Message Broker network can be deployed, a WebSphere MQ connection between LONDON and TOKYO is required. A series of WebSphere MQ commands must be invoked to define the channels and transmission queues that support two-way traffic.

When you have completed the migration of all the brokers in the collective, you have removed the single point of failure at the NEWYORK broker. Subscribers on the LONDON broker can receive publications from the TOKYO broker even when the NEWYORK broker is not running.

Before migration, traffic between brokers was always routed through NEWYORK, the root node, which was therefore the single point of failure.

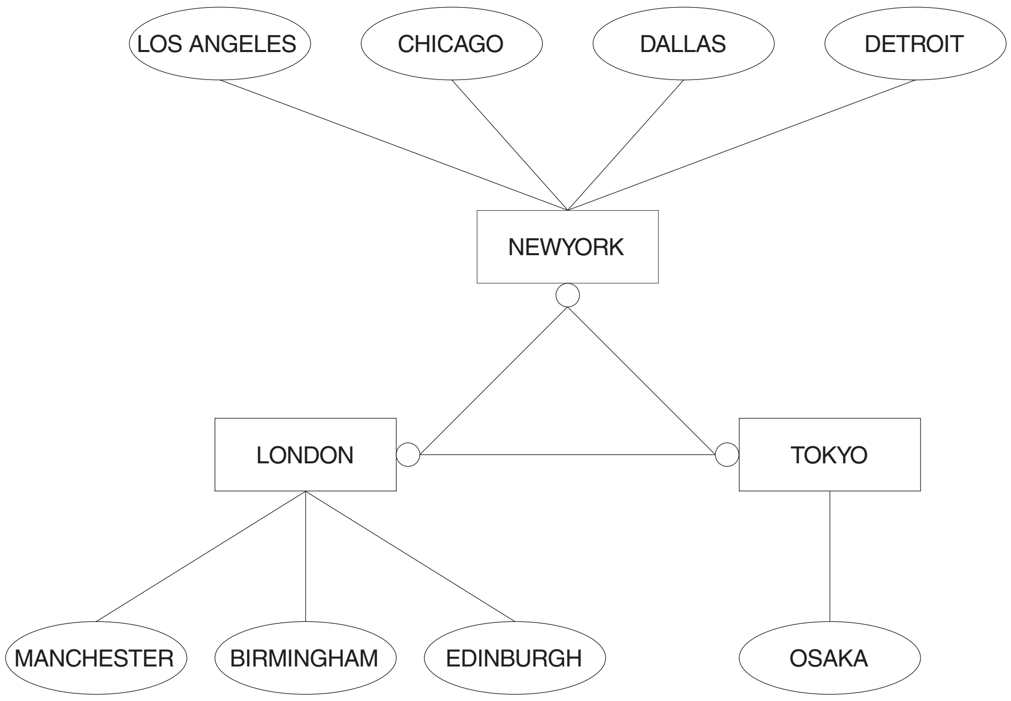
For further details of connecting brokers to each other, see Configuring WebSphere Message Broker. For more general information about distributed WebSphere MQ networks, refer to *WebSphere MQ Intercommunication*.

When all migration and associated tasks have been completed, the network comprises a single collective, containing three WebSphere Message Broker brokers connected as equals.



A network of migrated brokers

The following diagram shows a mixed network of WebSphere Message Broker and WebSphere MQ brokers. Brokers NEWYORK, LONDON, and TOKYO have been migrated to form a WebSphere Message Broker collective. All the other brokers remain as WebSphere MQ brokers.



Part 2. Reference

Migration and upgrade

This section contains the following topics:

Supported migration and upgrade paths

The following table shows the supported migration and upgrade paths to the Version 6.0 products:

From ...	You can migrate to ...
MQSeries® Integrator Version 2.0.2	• Migration from MQSeries Integrator Version 2.0.2, or an earlier release, is not supported.
WebSphere MQ Event Broker Version 2.1 (at any service level)	• WebSphere Event Broker Version 6.0 • WebSphere Message Broker Version 6.0
WebSphere MQ Integrator Broker Version 2.1 (at CSD02 or later)[1]	• WebSphere Message Broker Version 6.0
WebSphere MQ Integrator Version 2.1 (at CSD02 or later)[1]	• WebSphere Message Broker Version 6.0[2] • WebSphere Message Broker with Rules and Formatter Extension Version 6.0
WebSphere MQ Event Broker Version 2.1 (at any service level) and WebSphere MQ Integrator Broker Version 2.1[3] (at CSD02 or later)[1]	• WebSphere Message Broker Version 6.0
WebSphere MQ Event Broker Version 2.1 (at any service level) and WebSphere MQ Integrator Version 2.1[3] (at CSD02 or later)[1]	• WebSphere Message Broker Version 6.0[2] • WebSphere Message Broker with Rules and Formatter Extension Version 6.0
WebSphere Business Integration Event Broker Version 5.0	• WebSphere Event Broker Version 6.0 • WebSphere Message Broker Version 6.0
WebSphere Business Integration Message Broker Version 5.0	• WebSphere Message Broker Version 6.0
WebSphere Business Integration Message Broker with Rules and Formatter Extension Version 5.0	• WebSphere Message Broker Version 6.0 [4] • WebSphere Message Broker with Rules and Formatter Extension Version 6.0
WebSphere MQ Publish/Subscribe	• WebSphere Event Broker Version 6.0 • WebSphere Message Broker Version 6.0
WebSphere Event Broker Version 6.0	• WebSphere Message Broker Version 6.0

Notes:

1. If you have applied CSD02, CSD03, CSD04, or CSD05 to Version 2.1, you must apply an additional APAR (IY45459) in order for migration to be successful. APAR IY45459 is included in CSD06.
2. This migration path is possible only if you have not used the New Era of Networks Rules and Formatter support in WebSphere MQ Integrator Version 2.1, or you do not want to continue using it after migration.
3. Two broker domains are merged into one broker domain during migration.
4. This migration path is possible only if you have not used the Rules and Formatter function in WebSphere Business Integration Message Broker with Rules and Formatter Extension Version 5.0, or you do not want to continue using it after migration.

Message flow migration notes

This section provides reference information to assist you when you migrate message flows to WebSphere Message Broker Version 6.0. It contains the following topics:

- "Migrating message flows from Version 2.1"
- "Migration of message mappings from Version 5.0" on page 100
- "Migrating a user-defined node" on page 104
- "Migrating a message flow that contains WebSphere MQ Everyplace nodes" on page 106
- "Migration of style sheets and XML files" on page 113

If you are using the Version 5.1 Message Brokers Toolkit, replace all references in these topics to ″Version 5.0″ with ″Version 5.1″.

Migrating message flows from Version 2.1

Use the **mqsimigratemsgflows** command to migrate message flows from Version 2.1 to Version 6.0. **It is not necessary to use this command when migrating from Version 5.0 to Version 6.0.**

This topic contains the following sections:

- "Conditions for using the mqsimigratemsgflows command"
- "Custom property editor" on page 99
- "Promoted property name" on page 99
- "Converting Version 2.1 names that are not valid" on page 100
- "Mapping of node type" on page 100

Conditions for using the mqsimigratemsgflows command

- Export message flows from Version 2.1 to an export file. You can use SupportPac utilities to create the files, and these exported files are the input to the migration process.
- Message flows that are found in multiple export files are migrated each time that they occur. When migrating a message flow, any previous flow with the same name is overwritten without warning.
- Message flows and their related subflows should be exported to the same file. This is the default Version 2.1 export behavior. This helps you to identify and resolve references to the other message flows and subflows, and to rename

objects consistently. The migration uses the name of the message flow rather than the UUID to link flows and subflows and to name the message flow file.

- If a message flow refers to a subflow that cannot be located, but appears to be in the project, it is possible that it has been migrated separately, and was referenced internally in an inconsistent manner due to renaming. To correct the situation, export the message flows again in the same export file.
- User-defined nodes should be exported to the same export file as the message flows that are using them. If you do not do this, the message flows that have references to the user-defined nodes refer to these as `.msgflow` files rather than `.msgnode` files. The migration report specifies if this has occurred.
- User-defined nodes that are also input nodes (for example, those nodes that accept messages), must have their new ″input node″ system property set after migration using the Message Node editor. This allows the Rational Application Developer to recognize these nodes on the same level as, for example, MQInput.
- When migration is complete, open the workbench. If the project is closed, open it; if the project is open, refresh it and clean all the projects by clicking `Project -> Clean`.
- ESQL code is copied from the Version 2.1 message flow and put in a MODULE. There is no change to the code.
- ESQL filter expressions are migrated in the following manner:
  - If a RETURN statement is found in the code, a module is created, just like any other compute expression, field mapping or database statement.
  - If a RETURN statement is not found, the ESQL code is wrapped around a RETURN statement.

  Review the results of this migration in case the detection of the RETURN statement did not work as expected.

Custom property editor

When a user-defined node or a promoted property has a Property editor, the XML attribute is type=″MyType″ and there exists a class <*package*>MyTypePropertyEditor.class.

The Property editors themselves (written in Java code) are not migrated. However, if new ones are created (using the Eclipse SWT toolkit) with the same class name, the new editor is found and loaded without the need to change the migrated flow.

Promoted property name

In Version 2.1, when a promoted property is created through the drag-and-drop process, the property name (xmi.label) is set to be the translation of the attribute name. The original attribute name must not contain spaces otherwise it is rejected by the broker. However, promoted attributes are never sent to the broker, so they might have contained spaces in Version 2.1.

When the message flow is migrated, the original name is lost and only the translation is kept. The promoted attribute can override several attributes, so the original name must correspond to the translated name.

The solution is to generate a suitable attribute name by replacing spaces or other offending characters with the unicode representation. The *propertyName* attribute of the propertyDescriptor is set to `key=Property.<`*`the translated attribute name`*`>`. The UI returns <*the translated attribute name*>.

However, migrated message flows have not retained the attribute system name, only the translated name. It is therefore difficult or impossible to locate the original attribute. For example, a DataSource promoted property is not shown as translated if the message flow is shipped as a plug-in flow and another user flow promotes the property from the plug-in flow.

Converting Version 2.1 names that are not valid

Message flows and properties can contain names that are not valid in Version 6.0. If this situation arises, the following transformation occurs. Each offending character is replaced with a series of characters representing its unicode code point. For example, an exclamation mark (″!″) is replaced with X0026. This is explained in the report file that is generated.

This transformation is deterministic. If a message flow is migrated on another occasion, and the message flow refers to a flow with a character that is not valid, both names are transformed in the same way.

These transformations do not result in conflicting names except in extremely rare circumstances. A conflict might occur because a Unicode code point sequence occurs in a name precisely where the corresponding character occurs in another name, which is otherwise identical. In this case, rename one of these message flows or properties and re-export the flows. Select a new name that does not contain a Unicode code point sequence (’Xnnnn’) and rename the message flow in the Control Center before you migrate. Never rename a .msgflow file in the file system; always use the Control Center or the workbench to perform renaming tasks.

Mapping of node type

Version 2.1 nodes are converted to Version 6.0 nodes as follows:

Version 2.1 node	Version 6.0
Compute	Compute
Database	Database
DataDelete	Database
DataInsert	Database
DataUpdate	Database
Extract	Compute
Filter	Filter
Warehouse	Database

Migration of message mappings from Version 5.0

Use the **mqsimigratemfmaps** command to migrate message mappings. This command is part of the Message Brokers Toolkit, not the run time on Windows and Linux, and is not available in the command path by default. You can find the command under the Eclipse directory of the tooling installation. There are some restrictions on migrating message mappings.

The following table lists the mapping functions that are supported in Version 5.0 but not supported in Version 6.0, and shows the error messages that you might see. Mappings that contain these Version 5.0 functions cannot be migrated to Version 6.0, and must be re-created and redeployed using another node, such as a Java Compute node. Alternatively, try to migrate as much of the mapping as possible using the migration command, view the error report to see details of the functions that could not be migrated, and create a new node that will execute the non-migrated functions.

Supported in Version 5.0	Migration utility error message
Expressions that involve multiple instances of a repeating source element, for example: `src_msg.e[1] + src_msg.e[2] -> tgt_msg.e`	`Error:102: Unexpected index ’2’ encountered for target mappable ’e’. The expected index is ’1’. Migration currently provides no support for expressions involving more than one instance of the same repeating-element.`
ESQL field references that contain the asterisk wildcard character ″*″. For example: `src_msg.e.* or src_msg.e.*[]`	`Error:130: ESQL field-reference ’src_msg.e.*’ cannot be migrated. Migration currently provides no support for field-references containing ’*’.`
Dynamic ESQL field references. For example: `src_msg.e.{'a' || 'b'}`	`Error:131: ESQL field-reference ’src_msg.e.{’a’ || ’b’}’ cannot be migrated. Migration currently provides no support for dynamic field-references.`
ESQL expressions that contain a reference to the temporary index-variable ″#I″. For example: `src_msg_e || "#I" -> tgt_msg.e`	`Error:128: ESQL expressions containing the variable ’#I’ anywhere other than the index of a repeating-element cannot be handled by the migration.`
Expressions within an index of a repeating element. For example: `src_msg.e[src_msg.a] or src_msg.e["#I" +5] or src_msg.e[< 3]`	`Error:116: ESQL field-reference ’src_msg.e[< 3]’ cannot be migrated. Migration currently provides no support for indexes other than the variable ’#I’ and plain integer indexes.`
Aggregation functions MIN, MAX, and COUNT, used with the ESQL SELECT expression. For example: `SELECT MAX("#T".FIRSTNAME) FROM Database.CUSTOMER AS "#T" WHERE "#T".CUSTOMERID = 7`	`Error:135: The ESQL expression ’SELECT MAX(″#T″.FIRSTNAME) FROM Database.CUSTOMER AS ″#T″ WHERE ″#T″.CUSTOMERID = 7’ could not be migrated. The expression contains syntax which has no direct equivalent in the new map-script language.`
ESQL’s IN operator. For example: `src_msg.e IN (1, 2, 3)`	`Error:135: The ESQL expression ’SELECT MAX(″#T″.FIRSTNAME) FROM Database.CUSTOMER AS ″#T″ WHERE ″#T″.CUSTOMERID = 7’ could not be migrated.`

Restrictions on migrating message mappings

There are certain scenarios where the migration of .mfmap files is not supported. This topic explains why migration is not automatic in these situations, and provides instructions for how to complete a successful migration.

The programming model for message maps is different between Version 5.0 (where the file format is .mfmap) and Version 6.0 (where the format is .msgmap). Version 5.0 message maps have a procedural programming model, which is essentially an alternative ESQL, where you describe all the steps that are required to perform a transformation. Version 6.0 uses a declarative programming model, where you describe the result of the transformation, and the tools determine how to achieve that result.

Most migration failures result from message maps that contain too much information about the steps that perform the transformation, and not enough information about the desired result. For these message maps, migration is enabled by changing the .mfmap file so that specific ″how to″ sections are separated into an ESQL function or procedure that can be called by the message map. The .mfmap file calls the ESQL instead of containing it as an expression. The **mqsimigratemfmaps** command then migrates the .mfmap file, but calls the ESQL instead of logging a migration error.

A limitation is that ESQL (the run time for .mfmap and .msgmap files) cannot define functions that return complex element (or REFERENCE) values. The following procedure explains how to work around this complex element target limitation; in many cases, it means that the map must be rewritten as ESQL. For more examples and information about calling ESQL from maps, see the WebSphere Message Brokers mapping sample at **Help** → **Samples Gallery** → **Technology samples** → **Message Brokers** → **Message Map**.

1. Determine whether you can define an ESQL function for the .mfmap file.
   a. When the target value is a complex element, or in ESQL terms a REFERENCE, the individual mapping must be rewritten in the .msgmap file. Delete the mapping from the .mfmap file, and proceed to Step 4.
   b. Use a function for all other cases: CHAR string, numbers, date and time. Proceed to Step 2.
2. Determine the source parameters and returns type for your function.
   a. For each source path in the mapping, there must be one parameter in the function or procedure. For a function, all parameters are unchangeable. The type of the parameter must match the source data type.
   b. The function return type is the ESQL datatype identified above.
3. Update the .mfmap file to enable migration. Change the .mfmap file to invoke the function in the mapping, passing the source parameters to the function in the order in which they were listed in step 2a.
4. Re-run the **mqsimigratemfmaps** command to migrate the modified .mfmap file.
5. Repeat Steps 1 to 4 until there are no errors in the migration log.
6. Start the Version 6.0 Message Brokers Toolkit and open the migrated .msgmap file.
   a. For ESQL that is migrated as functions, there should be no errors.
   b. For complex element targets, rewrite the mapping using the Version 6.0 features.

The following examples illustrate migration of .mfmap files to .msgmap files.

- To migrate a multiple reference to a repeating source expression:

  ```
  src_msg.e[1] + src_msg.e[2]
  ```

  compute the result in an ESQL function like:

  ```
  CREATE FUNCTION addOneAndTwo(IN src_msg)
  BEGIN
   RETURN src_msg.e[1] + src_msg.e[2];
  END;
  ```

In the .msgmap file, call the ESQL function addOneAndTwo using the parent element src_msg as a parameter.

- An expression that does not use element names:

  ```
  src_msg.*
  ```

  or

  ```
  src_msg.*[]
  ```

  can be processed using a function that takes the parent of the repeating field:

  ```
  CREATE FUNCTION processAny(IN src_msg)
  BEGIN
   DECLARE nodeRef REFERENCE TO src_msg.e.*;
   DECLARE result <dataType> <initialValue>;
   WHILE LASTMOVE nodeRef DO
    --expression goes here
    SET result = result;
   END WHILE;
   RETURN RESULT;
  END;
  ```

  In the .msgmap file, call the ESQL function using the parent element src_msg as a parameter.

- Expressions that dynamically compute element names:

  ```
  src_msg.{'a' || 'b'}
  ```

  can be processed by ESQL functions that process the parent of the repeating field:

  ```
  CREATE FUNCTION processDynamicName(IN src_msg)
  BEGIN
   RETURN src_msg.{'a' || 'b'};
  END;
  ```

  In the .msgmap file, call the ESQL function using the parent element src_msg as a parameter.

- Expressions that use the select MIN, MAX, and COUNT functions:

  ```
  SELECT MAX("#T".FIRSTNAME)
   FROM Database.CUSTOMER AS "#T"
   WHERE "#T".CUSTOMERID = custId
  ```

  can be processed by ESQL functions that process the parent of the repeating field:

  ```
  CREATE FUNCTION processMAX(IN custId)
  BEGIN
   RETURN
   SELECT MAX("#T".FIRSTNAME)
    FROM Database.CUSTOMER AS "#T"
    WHERE "#T".CUSTOMERID = custId
  END;
  ```

  In the .msgmap file, call the ESQL function using the element custId as a parameter.

- .mfmap files that use mfmap index variables in expressions:

  ```
  e || "#I"
  ```

  must be rewritten entirely in ESQL. By definition, there must be a complex repeating parent element, and this is not supported by ESQL functions.

- Expressions that use source expressions to compute values:

```
src_msg.e[src_msg.a]
```

must be rewritten using if rows, msgmap:occurrence() functions, and ESQL functions:

```
for src_msg.e
 if
  condition msgmap:occurrence(src_msg/e) = src_msg/a
```

- For expressions that use index expressions to compute values:

  ```
  src_msg.e["#I" +5]
  src_msg.e[< 3]
  ```

  the entire .mfmap file must be rewritten in ESQL, because the .msgmap files do not yet support indexed access to repeating fields.
- .mfmap files that use ROW expressions to compute values:

  ```
  src_msg.e IN (1, 2, 3)
  ```

  must be rewritten in ESQL, because .msgmap files do not support ESQL ROW expressions.

Migrating a user-defined node

You must complete the following steps to migrate a user-defined node to WebSphere Message Broker Version 6.0:

1. "Migrating the Message Brokers Toolkit representation of the user-defined node"
2. "Migrating the runtime user-defined node code" on page 105

Migrating the Message Brokers Toolkit representation of the user-defined node

You can deploy a user-defined node that is written in the Version 5.0 Message Brokers Toolkit on the Version 6.0 Message Brokers Toolkit. Before you can deploy the user-defined node, you must migrate the Message Brokers Toolkit representation of the user-defined node to the Version 6.0 Message Brokers Toolkit.

To migrate from the Version 5.1 Message Brokers Toolkit to the Version 6.0 Message Brokers Toolkit:

1. Import the user-defined node project into the Version 6.0 Message Brokers Toolkit.
2. Select your user-defined node project in the Package Explorer, and click **Project> Clean Project**.

To migrate from the Version 5.0 Message Brokers Toolkit to the Version 6.0 Message Brokers Toolkit:

1. Import the user-defined node project into the Version 6.0 Message Brokers Toolkit.
2. Select your user-defined node project in the Package Explorer, and click **Project> Clean Project**.
3. Modify the `<requires>` element in the plugin.xml file in the user-defined node project root to match the following:

   ```
   <requires>
           <import match="greaterOrEqual" plugin="com.ibm.etools.mft.api" version="6.0.0"/>
   </requires>
   ```

4. Modify the ″org.eclipse.help.contexts″ extension in the same plugin.xml file to match the following:

   ```
   <extension point="org.eclipse.help.contexts">
    <contexts file="HelpContexts.xml"/>
   </extension>
   ```

When you have migrated your user-defined nodes, you do not need to migrate any message flows that contain the user-defined node.

You must now complete the step "Migrating the runtime user-defined node code."

Migrating the runtime user-defined node code

Before you start this step you must have Migrated the Message Brokers Toolkit representation of the user-defined node.

To migrate the runtime user-defined node code:

1. Put a copy of your compiled or packaged user-defined extension file on every broker system from which you intend to use it.
   - If you are migrating a Java user-defined node, then you can build the user-defined extension file once and distribute it to each of your systems .
   - If you are migrating a C user-defined node and all of your brokers are of the same machine type, you can build the user-defined extension file once and distribute it to each of your systems.
   - If you are migrating a C user-defined node and you have a cluster that consists of various machine types, for example one AIX, one Solaris, and one Windows broker, you must build the files separately on each machine type.
2. Specify the directory in which to put the file, by using either the **mqsichangebroker** or **mqsicreatebroker** command.

   In previous versions, the .lil or .jar file was saved in the installation directory. Do not save the .lil or .jar file in the WebSphere Message Broker installation directory.

   For C user-defined extensions, store the .pdb file that corresponds to the .lil file, in the chosen directory. The .pdb file provides symbolic information that is used by WebSphere Message Broker when displaying stack diagnostic information in the event of access violations or other software malfunctions.
3. Stop and start each broker to ensure that the existence of a new file is detected.

   There are two situations where a broker restart is not necessary:
   - If you have created an execution group in the Message Brokers Toolkit, and there is nothing yet deployed to it, you can add the .lil, .pdb, and .jar file to your chosen directory.
   - If something has already been deployed to the execution group that you want to use, add the .lil, .pdb, or .jar file to your chosen directory and then use the **mqsireload** command to restart the group. It is not possible to overwrite an existing file on the Windows operating system when the broker is running because of the file lock that is put in place by the operating system.

   Use these two approaches with caution because any execution group that is connected to the same broker will also detect the new .lil, .pdb, and .jar files when that execution group is restarted, or when something is first deployed to that execution group. By using the more conventional way of restarting the broker, you ensure that anyone with an interest in a particular execution group is made aware that recent changes have been made to the broker.

These two situations assume that you have already completed the previous step, and have therefore used either the **mqsichangebroker** command or the **mqsicreatebroker** command to notify the broker of the directory in which the user-defined extension files have been placed.

When you have installed a user-defined node, it is referred to by its schema and name, just like a message flow.

Migrating a message flow that contains WebSphere MQ Everyplace nodes

The support for connecting WebSphere MQ Everyplace device clients to WebSphere Message Broker has changed in WebSphere Message Broker Version 6.0.

This topic summarizes the changes, and the subsequent topics describe the situation in more detail:

1. Migrating an MQe message flow (this topic)
2. "Designing MQe connections" on page 107
3. "Deploying an MQe message flow" on page 108
4. "Configuring after MQe message flow deployment" on page 110

Previous versions

- You can use either of these connection configurations:
  1. MQe device client <--*connects to* --> MQe gateway <--*connects to* --> Broker (using MQInput and MQOutput nodes)
  2. MQe device client <--*connects to* --> Broker (using MQeInput and MQeOutput nodes)
- The MQe code level contained within the MQe nodes in the broker is at a fixed version, and you can not update it independently.

Version 6.0

- The MQeInput and MQeOutput nodes do not exist (the broker has no MQe code within it).
- You can use only this connection configuration:
  1. MQe device client <--*connects to* --> MQe gateway <--*connects to* --> Broker (using MQInput and MQOutput nodes)
- You must install WebSphere MQ Everyplace separately. The MQe code is now all contained in that separate installation and it can be upgraded independently to newer versions.
- You can still import or create a message flow with MQe nodes in it, and save it and deploy it to a broker:
  – The bar files generated will be exactly the same as before.
  – When you deploy to a previous version broker, the flow will work just as before.
  – When you deploy to a Version 6.0 broker, the broker run time interprets the bar file differently, changing the MQeInput and MQeOutput nodes into MQInput and MQOutput nodes, ignoring some MQe node attributes and reinterpreting others.

  This way, you can migrate your existing MQe message flows unchanged to WebSphere Message Broker Version 6.0 (you must configure the MQe gateway according to the reinterpreted attributes).

Attention: The use of message flows that contain MQeInput and MQeOutput nodes in WebSphere Message Broker Version 6.0 is deprecated. The behavior that is described here is intended only for when you are deploying from Version 6.0 to a previous version, and to provide a route for migration. Redesign your flows to remove the MQe nodes and replace them with MQ nodes that are configured to your own specifications and coordinated with your MQe Gateway configuration.

- If you are using MQeOutput nodes with the *Destination Mode* property set to `Destination List`, when you migrate your MQe flows to a Version 6.0 broker, change the *Destination Mode* property to `Reply to Queue` then redeploy the modified flows. You can use either the Version 6.0 or previous version of the Message Brokers Toolkit to perform the redeployment.

If you want to use MQe connections with WebSphere Message Broker Version 6.0 see the subsequent topics for more details, starting with "Designing MQe connections."

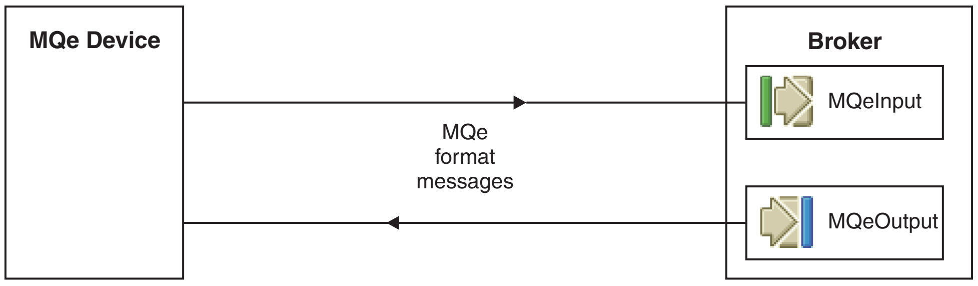
For help with configuring WebSphere MQ Everyplace, see the documentation supplied with that product.

Designing MQe connections

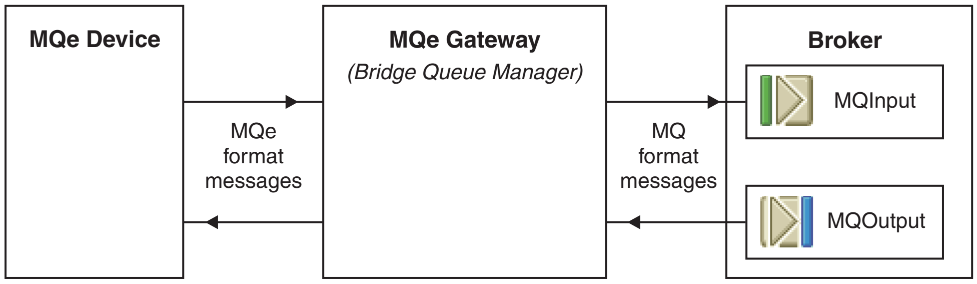
This topic is part of a set of related topics that are best read in total and in sequence:
1. "Migrating a message flow that contains WebSphere MQ Everyplace nodes" on page 106
2. Designing MQe connections (this topic)
3. "Deploying an MQe message flow" on page 108
4. "Configuring after MQe message flow deployment" on page 110

This topic describes the connection possibilities between WebSphere MQ Everyplace device clients and WebSphere Message Broker.

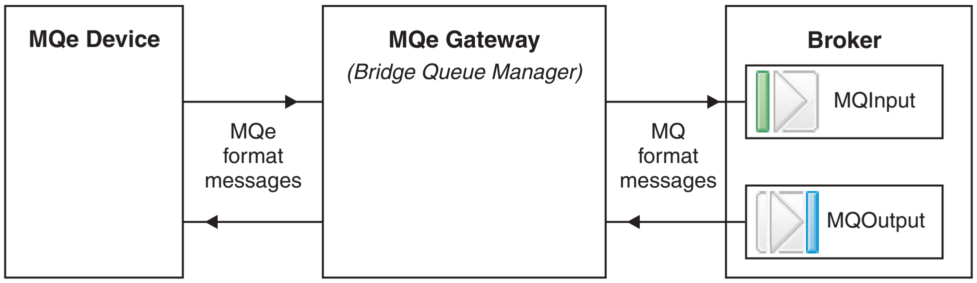
The following two diagrams illustrate the possibilities visually, and a table follows with the details.
1. The connection options for previous versions:
   a. Directly



   b. Via MQe Gateway



2. The connection options for Version 6.0:
   - Via MQe Gateway



The following table lists connection possibilities between WebSphere MQ Everyplace device clients and WebSphere Message Broker

Connection considerations	Previous versions	Version 6.0
MQe devices connect directly to the broker	Yes	No
MQe devices connect to an MQe Gateway, then that connects to the broker	Yes	Yes
The format of messages into and out of the broker is:	MQe format or MQ format	MQ format only

The next topic in this set is “Deploying an MQe message flow.”

For help with configuring WebSphere MQ Everyplace, see the documentation supplied with that product.

Deploying an MQe message flow

This topic is part of a set of related topics that are best read in total and in sequence:
1. “Migrating a message flow that contains WebSphere MQ Everyplace nodes” on page 106
2. “Designing MQe connections” on page 107
3. Deploying an MQe message flow (this topic)
4. “Configuring after MQe message flow deployment” on page 110

The following table describes the deployment possibilities of a message flow that contains MQeInput and MQeOutput nodes.

Attention: The use of message flows that contain MQeInput and MQeOutput nodes in WebSphere Message Broker Version 6.0 is deprecated. The behavior that is described here is intended only for when you are deploying from Version 6.0 to a previous version, and to provide a route for migration. Redesign your flows to remove the MQe nodes and replace them with MQ nodes that are configured to your own specifications and coordinated with your MQe Gateway configuration.

Deploy from	**Deploy to**	**Result**
Previous version	Previous version	Works as before
Previous version	Version 6.0	Not possible - instead you must import your old message flow into your Version 6.0 Message Brokers Toolkit and then deploy from there.[1]
Version 6.0	Previous version	Works as before.[2]
Version 6.0	Version 6.0	Works when you set up an appropriately configured MQe gateway. When the broker receives the bar file, the MQeInput and MQeOutput nodes in the message flow are converted into MQInput and MQOutput nodes that are re-configured as described below.

Notes:

1. When you save the flow in Version 6.0 you see an error message saying that the use of MQe nodes is now deprecated. You can override the error and continue to save and deploy.
2. The bar files generated by Version 6.0 contain XML that is exactly the same as previous versions.

The following list describes what happens to all the MQe node attributes when the node is converted into an MQ node. (For a summary of the key attributes, see the next topic in this set, "Configuring after MQe message flow deployment" on page 110).

MQeInput and MQeOutput node properties panels	**Converted MQ node properties**
General panel attributes	
• Trace • Trace Filename • Use Config File • Config Filename • Queue Manager Name • Queue Name	All discarded except: • Queue Manager Name • Queue Name Do your tracing in the normal way for WebSphere Message Broker, as described in the Using trace topic. When this flow is traced, the MQ nodes are seen, not the MQe nodes in the message flow in the deployed bar file that have been converted.

MQeInput and MQeOutput node properties panels	Converted MQ node properties
Registry panel attributes	
• Registry type • Directory • PIN • Certificate Request PIN • Keyring Password • Certificate Host • Certificate Port	All discarded These attributes concern MQe registry and security, and you configure that on your MQe gateway.
Listener panel attributes	
• Listener Type • Host Name • Port • Time Interval	All discarded You configure these attributes on your MQe gateway and your broker queue manager.

To use such a deployment, create a suitable configuration on the broker queue manager for the converted nodes to work, as described in the next topic, "Configuring after MQe message flow deployment."

For help with configuring WebSphere MQ Everyplace, see the documentation supplied with that product.

Configuring after MQe message flow deployment

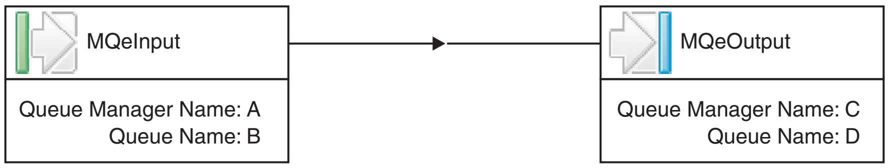
This topic is part of a set of related topics that are best read in total and in sequence:
1. "Migrating a message flow that contains WebSphere MQ Everyplace nodes" on page 106
2. "Designing MQe connections" on page 107
3. "Deploying an MQe message flow" on page 108
4. Configuring after MQe message flow deployment (this topic)

When you deploy a message flow that contains MQeInput and MQeOutput nodes on WebSphere Message Broker Version 6.0 the nodes are converted to MQInput and MQOutput nodes (for full details see "Deploying an MQe message flow" on page 108).

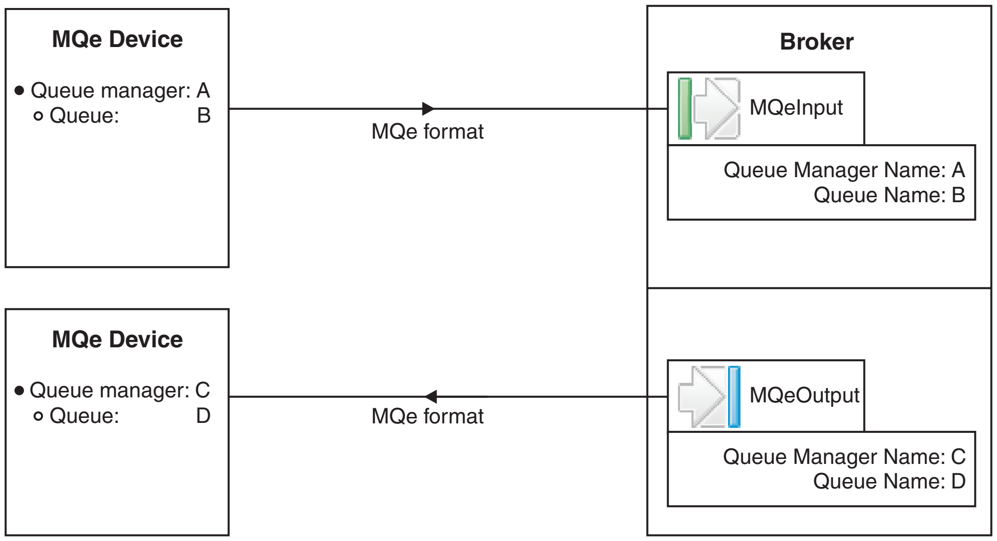
Attention: The use of message flows that contain MQeInput and MQeOutput nodes in WebSphere Message Broker Version 6.0 is deprecated. The behavior that is described here is intended only for when you are deploying from Version 6.0 to a previous version, and to provide a route for migration. Redesign your flows to remove the MQe nodes and replace them with MQ nodes that are configured to your own specifications and coordinated with your MQe Gateway configuration.

When the MQe nodes are converted into MQ nodes at deployment time, the *queue manager name* and *queue name* attributes are preserved, and you must create a configuration on the broker queue manager to handle them.

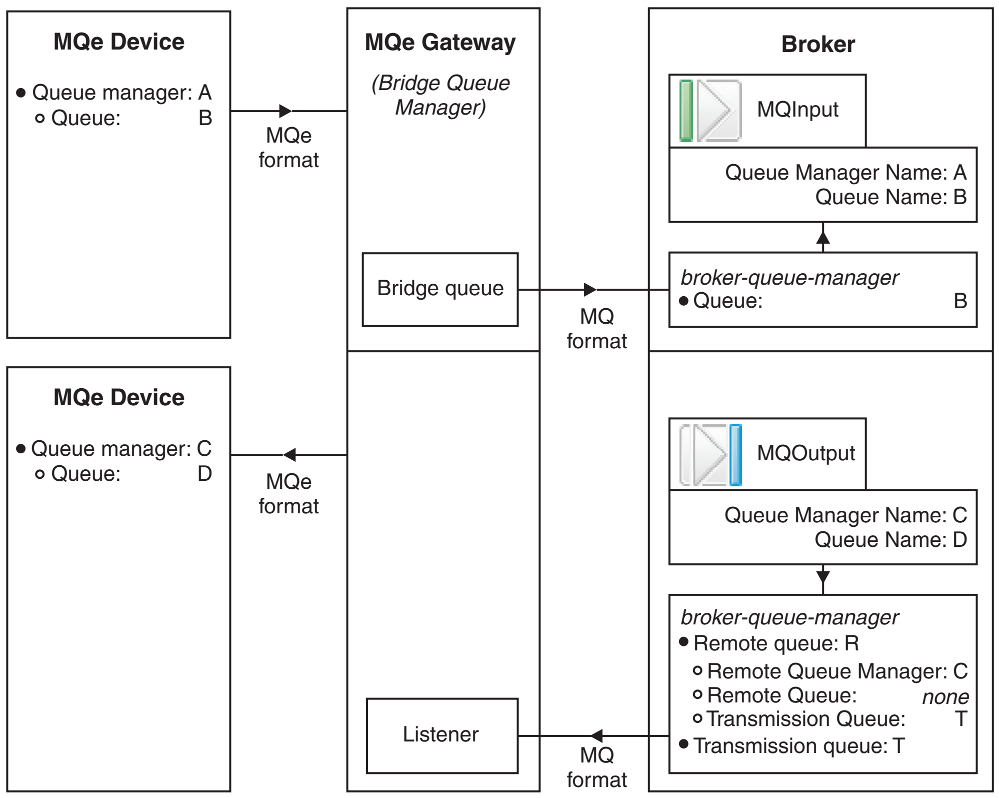
For example, suppose you have a message flow like this simplified one that contains MQeInput and MQeOutput nodes:



The following diagram shows this message flow connected directly to MQe devices, as it would be used on a previous version broker:



When you deploy this message flow to a Version 6.0 broker, create a configuration as shown here:



Configuration details:

1. Configure your broker queue manager with the following parameters:
   - A local queue named B
   - A remote queue definition R with these attributes:
     - Remote Queue Manager = C
     - Remote Queue = *none* (leave blank)
     - Transmission Queue = T

   This remote queue definition will match all messages that are destined for any queue on queue manager C and put them to the transmission queue.
2. Configure your MQe gateway with the following parameters:
   - Provision for accepting messages from an MQe device with queue manager A and queue B, transforming the format from MQe to MQ, and putting them onto a bridge queue
   - A bridge queue that puts messages to queue B on the broker queue manager
   - A listener that reads messages from transmission queue T on the broker queue manager
   - Provision for transforming the message format of the messages on queue T from MQ to MQe format, and sending them to the MQe device with queue manager C and queue D

For help with configuring WebSphere MQ Everyplace, see the documentation supplied with that product.

Migration of style sheets and XML files

If you do not intend to use the new deployment feature, and your XMLTransformation nodes reference style sheets with partially-qualified paths (relative referencing), you might need to migrate both the message flows and the style sheets. If your XMLTransformation nodes reference style sheets with fully-qualified paths (absolute referencing), migration is not necessary.

You can continue to deploy your style sheets and XML files manually into the broker file systems, but relatively-referenced style sheets and XML files (and their dependants) must be stored in the *broker work path*/{XSL,XML}/external directories.

In Version 2.1 and Version 5.0, any relative references to manually-deployed style sheets are relative to the location of the file directory where the message broker is started. In Version 6.0, any relative references to manually-deployed style sheets are relative to the directory *broker work path*/{XSL,XML}/external on the broker file system. This means that you must complete the following actions:

1. Move the relatively-referenced style sheets (and their descendants), to the correct directory structure under *broker work path*/{XSL}/external.
2. Move all their dependent XML files to the correct directory structure under *broker work path*/{XML}/external.

For example, a relatively-referenced principal style sheet, a/b.xsl, must now be stored as *broker work path*/{XSL}/external/a/b.xsl on the broker file system. An XML file that is referenced by a/b.xsl as c/d.xml must be stored as *broker work path*/{XML}/external/a/c/d.xml.

Version 6.0 treats an embedded style sheet in the same way as a file that is loaded from the root of the broker’s deployed storage, so move any descendant style sheets or XML files of embedded style sheets to the *broker work path*/{XSL,XML}/external tree. If this is not possible, you can reference them with a fully-qualified path (absolute reference).

Message set migration notes

This topic provides information that you need to know when migrating message sets to WebSphere Message Broker Version 6.0. It contains the following sections:

- “Migrating from Version 2.1”
- “Migrating from Version 5.0” on page 114
- “General migration information” on page 114
- “Behavioral changes in the MRM parser” on page 115

If you are using the Version 5.1 Message Brokers Toolkit, replace all references in this topic to ″Version 5.0″ with ″Version 5.1″.

Migrating from Version 2.1

To migrate message sets from Version 2.1 to Version 6.0, use the **mqsimigratemsgsets** command to convert your Version 2.1 message set export files (.mrp) into Version 6.0 message set projects. Before you run the command, refer to the “Migrating Message Sets from Version 2.1” on page 117 topic, which provides detailed notes on its operation.

Migrating from Version 5.0

To migrate message sets from Version 5.0 to Version 6.0, no migration commands are necessary. The content of a Version 5.0 message set project can be read by the Version 6.0 Message Brokers Toolkit and is converted automatically to Version 6.0 format when it is modified and saved for the first time.

General migration information

The following information is applicable whether you have migrated from Version 2.1 or Version 5.0.

- The CWF property Repeat Count has been superseded by the logical property Max Occurs, bringing the CWF physical format in line with TDS and XML physical formats, which already use Max Occurs to determine the number of repeats. A warning appears in the Problems view of the message editor for each element or group that had a CWF Repeat Count value set. Right-click the warning and click **Quick Fix** to resolve the issue.

  Note: In Version 5.0, if the CWF property Repeat Count is not set *and* Min Occurs is not equal to Max Occurs, the number of repeats is taken to be one. In Version 6.0, the number of repeats is taken to be Max Occurs. It is not possible to issue a warning for this situation. Such a message model is not created by the COBOL or C importers, so it can occur only if you have created a CWF message model using the Version 5.0 message editor.
- The TDS physical format prior to Version 6.0 included embedded message identification by Message Key. The Message Key technique has been deprecated and is superseded by a technique called Message Identity. A warning appears in the Problems view of the message editor for each message that has a TDS Message Key value, and for each element or attribute that has TDS Interpret Element Value property set to Message Key. Right-click the warning and click **Quick Fix** to resolve the issue.

  Continue to use TDS Message Key if the message set will ever be deployed to a Version 5.0 or Version 2.1 broker, because these brokers do not support the Message Identity technique of embedded message identification.
- The TDS physical format property Data Pattern is now validated by the Message Brokers Toolkit to ensure that it is a valid XML Schema regular expression. Errors appear in the Problems view of the message editor, and you must correct them manually using the editor. In particular, there have been some errata issued against the XML Schema specification concerning the escaping of meta-characters, which can cause validation errors to appear. To correct the errors, you might need to escape the hyphen character ("-") or the curly brace characters ("{" and "}"), using the backslash character ("\"); for example "\{" or "\-". If you receive such errors when using an IBM-supplied message set, contact IBM to obtain a corrected version of the message set.
- The default value or fixed value of an element or attribute, and enumeration values of a simple type, are now checked against the length, maximum length and minimum length facets of the simple type, if supplied. If a value does not comply with the facets, an error appears in the Problems view of the message editor. Right-click the error and click **Quick Fix** to resolve the issue, if a quick fix is available. Otherwise, correct the problem manually using the editor. This error is most likely to be encountered if you import a COBOL copybook into the Version 5.0 Fix Pack 2 or earlier version of the Message Brokers Toolkit.
- The CWF and TDS Encoding Null and Encoding Null Value properties have been removed from local attributes, global attributes and attribute references. In

XML Schema, only elements can have a Nillable property, therefore a null value cannot be carried by a data field that is modelled as an attribute. If you had specified CWF or TDS Encoding Null values for attributes in Version 5.0, they were ignored.

- The TDS Repeating Element Delimiter property has been removed from local attributes and attribute references. In XML Schema, attributes cannot repeat. If you had specified TDS Repeating Element Delimiter values for attributes in Version 5.0, they were ignored.
- For an element or attribute with simple type xsd:gMonth, the default value for the CWF, XML, and TDS Datetime Format property is now ″--MM″. In Version 5.0, the default value is ″--MM--″. This has been corrected to comply with an XML Schema errata.
- When you are migrating, you do not have to redeploy your message sets because message dictionaries are upgraded in the broker database to Version 6.0 format by the **mqsimigratecomponents** command. However, if you do not redeploy a message set, and you observe a ParserException, such as a ″checked vector error″ or an ″access violation″, when you first exercise a message flow that uses the message set, you will have to redeploy the message set. This can happen if the message dictionary contains previously undetected bad data. In Version 6.0, a message dictionary is checked on first use; in Version 2.1 and Version 5.0, this is not the case.

Behavioral changes in the MRM parser

The following information is applicable whether you have migrated from Version 2.1 or Version 5.0:

- XML physical format. Version 6.0 is now sensitive to xsi:type attributes in the XML document and modifies its behavior accordingly. xsi:type attributes are no longer treated as self-defining attributes, so they appear in the message tree with the name ‘type’ instead of ‘@type’. If your message flow logic is sensitive to xsi:type attributes in the message tree, change your message flow to comply with the new behavior. To retain pre-Version 6.0 logic in your message flows, set the environment variable MQSI_IGNORE_XSI_TYPE to any value, and the pre-Version 6.0 behavior will be adopted.
- XML physical format. DOCTYPE information in an XML document does not appear in the logical message tree when parsed, but it is preserved from input document to output document because the MRM keeps a copy of the DOCTYPE internally. Prior to Version 6.0, this was an exact copy of the bit stream. In Version 6.0, the copying process causes some formatting white space to be lost. For example, the element declaration below in an input DOCTYPE:

  ```
  <!ELEMENT e0 (e1|e2)+>
  ```

  will appear in the output as:

  ```
  <!ELEMENT e0 (e1|e2)+>
  ```

  The new behavior is consistent with the way that the XML physical format processes white space in all other XML constructs.
- TDS physical format. For a group with logical property Composition set to Choice and TDS property Data Element Separation set to Fixed Length, the parser always assumes that the length of the choice data is that of the longest child of the group, and will always read and write that number of characters. Ensure that your messages comply with this, otherwise the parser will treat data that follows the choice as part of the choice. Prior to Version 6.0, the parser did not enforce this rule.

- TDS physical format. If TDS element property Encoding Null is set to NullPadFill, it is only acted upon if the corresponding Length property is being used actively when parsing or writing. Prior to Version 6.0, NullPadFill was acted upon whether or not the Length property was being used actively.
- TDS physical format. TDS property Data Element Separation is set to All Elements Delimited or Variable Length Elements Delimited. A complex element is now always treated as having variable length, regardless of its own Data Element Separation setting. This means that in the message bit stream, a Delimiter is expected between the complex element and any subsequent element, and a Repeating Element Delimiter is expected between all instances of a complex element, even if the complex element has a fixed length. Prior to Version 6.0, the parser did not enforce this rule, and for Variable Length Elements Delimited, the writer did not output delimiters when the length of the complex element was known.
- TDS physical format. TDS property Data Element Separation is set to Use Data Pattern. If the regular expression that is specified for a data pattern returns a zero length match, this is now treated as an element being present with a zero-length value. Prior to Version 6.0, a zero-length match was treated as an element being omitted.
- TDS physical format. TDS property Data Element Separation is set to Tagged Delimited. An extraneous delimiter that occurs after the last child of the group in the bit stream is no longer tolerated and will cause a parsing exception. Prior to Version 6.0, the parser did not enforce this rule. If this is a problem, consider modelling the extraneous delimiter using the TDS Group Terminator property.
- TDS physical format. TDS property Data Element Separation is set to Tagged Delimited. In Version 6.0, when parsing a Tagged Delimited group, the parser attempts to make sense of groups where the members appear out of order in the message bit stream, even if the group Composition property is set to Sequence or OrderedSet. However, if the group contains either an embedded message or a complex element or group that cannot be identified from the bit stream, all the members of the group must appear in the bit stream in the same order that they are defined in the model. If they appear out of order, the group will not be parsed correctly and will have unpredictable results. One symptom of this is the appearance of self-defining elements in the message tree, caused by a failure to match an element to the model.

  A specific example of this is where your message contains an embedded message and you are using either the Message Key or Message Identity technique to identify the embedded message. If the element that is providing the message key or message identity value fails to be matched with the model, the parser will not know whether to interpret its value as a message key or message identity.

  Prior to Version 6.0, the parser attempted to make sense of all out-of-order Tagged Delimited groups, with a consequent reduction in performance. In Version 6.0, if this is a problem, consider modeling the unordered content of the group as an embedded child group with Composition set to UnorderedSet.

  A complex element or group can be identified from the bit stream if it provides a group indicator, a tag, or a data pattern, or if its child members provide a group indicator, tag or data pattern.

  Despite its name, there are circumstances where members of a Tagged Delimited group do not need to provide a tag; specifically, if the member is an embedded message or is a complex element or group.
- TDS physical format. The TDS Data Element Separation property is set to Tagged Encoded Length. When writing, the encoded length is output as the number of characters in the data, to match the interpretation when parsing. Prior

to Version 6.0, the encoded length was output as the number of bytes in the data, which did not match the interpretation when parsing if the data was not SBCS.

- TDS physical format. When writing, if the number of repeats of an element or group is fixed in the model but the actual occurrences in the logical message tree exceed the number of repeats, and the Data Element Separation property is not one of the ″Tagged″ separations, the extra occurrences are discarded and do not appear in the output. Prior to Version 6.0, the extra occurrences appear in the output. If the extra repeats are expected to appear in the output, specify `maxOccurs=-1` to indicate that the number of repeats is unbounded.
- All physical formats. When writing dateTime fields that have been given a Format property of I, the output form depends on the logical type of the field, as described in DateTime as string data. Prior to Version 6.0, the output form was incorrectly the full yyyy-MM-dd'T'HH:mm:ssZZZ form.
- Validation of input and output messages has been improved to provide additional detection of invalid messages. In Version 6.0, it is therefore possible for a message to be flagged as invalid when it was incorrectly not flagged in previous versions.

Migrating Message Sets from Version 2.1

To migrate message sets from Version 2.1 to Version 6.0 use the **mqsimigratemsgsets** command. **It is not necessary to use this command when migrating from Version 5.0 to Version 6.0.**

This topic contains the following sections:

- "Conditions of the **mqsimigratemsgsets** command"
- "What the **mqsimigratemsgsets** command creates" on page 119
- "Prefixed identifiers" on page 120
- "Embedded simple type" on page 121
- "Compound type with an MRM base type" on page 121
- "Mapping of MRM simple types to schema simple types" on page 122

Conditions of the mqsimigratemsgsets command

Do not modify the message set file manually between export from Version 2.1 and import into WebSphere Message Broker Version 6.0 because this causes errors, indicated by those warning and error messages in the report: BIP0141, BIP0142 to BIP0157, and BIP0163.

Each new message definition file `.mxsd` is an annotated XML schema model, and each artifact in the message set is re-created and retains its existing properties in the new model, with the following exceptions:

- The *Name* in the WebSphere Message Broker Version 6.0 model is the *Identifier* from the Version 2.1 model. If the object is an element with a prefixed identifier, the prefix is removed, as the prefix in Version 2.1 was a way of indicating that this element was local.
- *Label*, *Short* and *Long Description*, and *History* are merged into a single *Documentation* property for WebSphere Message Broker Version 6.0
- Each compound type becomes an xsd:complexType and an associated xsd:group in WebSphere Message Broker Version 6.0.

The `xsd:complexType` that is created in this way can be local or global. The default is local, but it becomes global if any one of the following conditions is true:

  - The compound type is unreferenced
  - The compound type has a Version 2.1 MRM base type
  - The compound type is referenced by more than one element
  - A message is based on the compound type
  - The **-g** parameter is specified

  The `xsd:group` that is created in this way can be local or global. The default is local, but it becomes global if any of the following conditions are true:

  - The compound type is embedded directly within another compound type.
  - The compound type has a Version 2.1 MRM base type.
  - The compound type has a Version 2.1 type composition of `simpleUnorderedSet` and a Version 2.1 type content of `closed`.

  The type composition of `simpleUnorderedSet` is dropped from the model in WebSphere Message Broker Version 6.0.

  - If type content is `closed` then it is replaced by composition `all` with a BIP0191 warning message.
  - Otherwise, it is replaced by composition `unorderedSet` with a BIP0192 warning message.
  - Type composition of `empty` is replaced by an empty `sequence` with a BIP0193 warning message.
- Embedded messages with *minOccurs* or *maxOccurs* not equal to 1 have the occurs corrected to 1 with a BIP0162 warning message.
- Each element becomes an `xsd:element`. The `xsd:element` that is created in this way can be local or global, depending on the following criteria applied in the specified order:
  1. If the element is unreferenced, it is global.
  2. If the element has a prefixed identifier and is a member of exactly one compound type, it is local.
  3. If the element has a prefixed identifier and the **-pl** parameter is specified, it is local.
  4. If the element is of a compound type with a Version 2.1 MRM base type, it is local.
  5. If the element is a member of more than one compound type, it is global.
  6. If the **-g** parameter is specified, it is global.
  7. Otherwise, the element is local.

  The type of the `xsd:element` that is created in this way can be:

  - A global `xsd:complexType`
  - An anonymous local `xsd:complexType`
  - A schema built-in `xsd:simpleType`. (See "Mapping of MRM simple types to schema simple types" on page 122.)
  - An anonymous `xsd:simpleType` that restricts a schema built-in `xsd:simpleType`
- Any value constraints that belong to the element are processed as follows:
  - A Default constraint sets the default attribute of the `xsd:element`.
  - A Null Permitted constraint sets the nillable attribute of the `xsd:element`.

  - A Date Template constraint changes the `xsd:simpleType` of the `xsd:element`. (See “Mapping of MRM simple types to schema simple types” on page 122.)
  - All others restrict the `xsd:simpleType` of the `xsd:element` by the application of `xsd:facets`.

  Any unreferenced value constraint is discarded with a BIP0158, BIP0159, or BIP0160 warning message.

  Any value constraint that results in an illegal `xsd:facet` being created is not migrated, but gives a BIP0165 warning message.

  Note: For elements of type STRING only, value constraints *MinInclusive* and *MaxInclusive* are instead imported as hidden annotations for documentation purposes, because there is no equivalent `xsd:facet`.
- Element qualifiers are discarded with a once-only BIP0167 warning message.
- Each message becomes a message *and* an associated global `xsd:element`.
- A message that used element qualifiers has the qualification discarded with a BIP0166 warning message.
- Some physical format properties that were redundant in Version 2.1 have been removed from the new model. If one of these properties is encountered with a non-default value, a BIP0164 (or other more specific) warning message is issued.
- The TDS message set level property *Century window* default value was always set to 53 in Version 2.1. For messaging standard SWIFT this is incorrect, so the default for SWIFT only is changed to 80. This is reflected in an imported model.

What the mqsimigratemsgsets command creates

For each `.mrp` file encountered, a new message set project is created with a name that is derived from the message set name and level in Version 2.1. The utility does this by adding a suffix to the message set name for all Level values other than 1. This process restores the one-to-one mapping and enables the broker to locate just one message set, given the name.

For example, a Version 2.1 message set with name SWIFT and Level 1, migrates in Version 6.0 to the message set name SWIFT, whereas a Version 2.1 message set with name SWIFT and Level 2 migrates in Version 6.0 to SWIFT_2.

A message set folder and associated `messageSet.mset` file is created within the new project. The following points apply to the content of the message set:
- The message set folder name is the same as the new project.
- The message set identifier is the identifier of the original Version 2.1 message set.
- The message set is created specifying that namespaces are not supported.
- All physical format layers are re-created in the new message set.
- Any COBOL language binding layer is discarded with a BIP0174 warning message.
- Any C language binding layer is discarded with a BIP0173 warning message.
- Any finalized state is discarded with a BIP0170 warning message.
- Any basing information is discarded with a BIP0172 warning message.
- Any freeze timestamp is discarded with a BIP0169 warning message. Note that you always receive this message because the Version 2.1 export operation freezes the message set.

Each message category that is encountered in the `.mrp` file results in a new `.category` file being created. Note:

- Each transaction is replaced by the equivalent message category, containing all the messages in the transaction.
- Each transaction category is replaced by the equivalent message category, containing all the messages in all the transactions that are referenced by the transaction category.

A single message definition `.mxsd` file is created within the message set with the same name as the message set and in the default (notarget) namespace, unless the **-part** parameter is present.

If **-part** is specified, multiple `.mxsd` files can be created, if:

- The number of messages, elements and compound types in the file exceeds 1000, and
- The `.mrp` file can be partitioned into wholly disjoint subsets.

In Version 2.1, all elements and compound types are global. In Version 6.0, `xsd:elements` and `xsd:complex types` can be global or local. When you migrate a Version 2.1 message set, you will probably find that many elements and compound types that were global in Version 2.1 have been converted into local `xsd:elements` and `xsd:complex types` in Version 6.0, according to the rules stated above.

There are two reasons why you might want to override this behavior:

- You prefer the Version 2.1 organization of your message set, where all objects are global. If you specify the **-g** parameter, this organization is retained as far as is practical.
- You make use of compound type *Type content* with a value of `Open defined`.

This means that the valid content of a compound type can be any object in the message set, subject to *Type composition* property rules. Typically in this case, the compound type has not been modelled with any explicit content.

This can cause the **mqsimigratemsgsets** command to incorrectly make certain elements local rather than global. If you use `Open defined` and you find that after migration a runtime validation error BIP5372E occurs, when previously it did not, rerun the **mqsimigratemsgsets** command with the **-g** parameter.

Prefixed identifiers

In Version 2.1, a prefixed identifier was intended to indicate that an element is local. However, it is possible that an element with a prefixed identifier is actually used in more than one compound type, making it global. If this is so, a global `xsd:element` is created according to the preceding rules. A BIP0195 warning message is also issued, because this is a misuse of prefixed identifiers, and can result in duplicate global `xsd:elements` being created. For example, A^X and B^X, but both are used more than once, resulting in two global `xsd:elements` with name X.

If duplicates are created, run the **mqsimigratemsgsets** command again, specifying the **-pl** parameter. This forces all referenced elements with prefixed identifiers to be created as *local* `xsd:elements`.

Embedded simple type

Embedded simple types within compound types need special treatment because the schema model is unable to cope with this construct. As embedded simple types are deprecated, replace them by taking advantage of the `mixed` attribute of the containing `xsd:complex` type.

Embedded simple types were introduced primarily to model a complex XML element that contained data values that were interspersed between child elements. Each such data value was modelled explicitly by an embedded simple type, which acted as a placeholder for the value and also supplied its simple type.

In XML schema, there is no exact equivalent. The closest is the mixed attribute of `xsd:complexType`. However, this only states that text can appear before or between child elements. It implies nothing about the location or data type of the text.

To retain this semantic, a schema extension is introduced, called the Embedded Simple Type. This is an unnamed local `xsd:element` of the appropriate simple type. The type itself is a restriction of the real underlying `xsd:simple type`, with a special name (commencing `ComIbmMrm_Anon`).

Compound type with an MRM base type

This situation produces a BIP0161 warning message and needs special treatment because the schema model is unable to cope with this compound construct. Compound elements are deprecated, so replace the use of compound elements with the use of normal elements that reference the global `xsd:complexType`, as described in 1 below, and take advantage of the mixed attribute.

Such compound types were introduced primarily to model a complex XML element that contained a data value as well as child elements. So an element of such a complex type has both complex content like a normal complex element, but also has a value like a simple element (the MRM base type information).

In XML schema, there is no exact equivalent. The closest is use of the mixed attribute of the `xsd:complexType`. But this just says that text can appear before and between (or just between) child elements. It implies nothing about the location or data type of the text.

There is no exact equivalent in XML schema, so the following steps have been taken:

1. A global `xsd:complexType` and a global `xsd:group` are created in the usual way. The `xsd:complexType` additionally has the mixed attribute set and its content is just a reference to the global `xsd:group`. This models the complex content, but loses the MRM base type information.
2. A specific schema extension, the Compound Element is introduced. This is an amalgamation of a complex element and a simple element. It is implemented in schema terms as an element with an anonymous complex type, the content of that type being:
   - A local `xsd:element` of the appropriate simple type (to model the MRM base type information). The simple type is a restriction of the real underlying `xsd:simple type`, with a special name (commencing `ComIbmMrm_BaseValue`)
   - A group reference to the global `xsd:group` created in 1 above.

A compound element is created for each element that referenced the compound type. Note that this can be done only if the element was itself a member of another compound type.

The combination of these two things means that meaningful use of such compound types within a message is preserved, because the MRM base type information is lost only when it was never used actively in a message.

The special data types that are created by the situations described in the preceding sections, commencing `ComIbmMrm`, are defined in an XML schema called `.wmq21.mxsd`, which is included in each message definition file that is created by the **mqsimigratemsgsets** command.

Mapping of MRM simple types to schema simple types

The simple type mapping is as follows:

MRM type	Schema type
BINARY	`xsd:hexBinary`
BOOLEAN	`xsd:boolean`
DECIMAL	`xsd:decimal`
DATETIME	`xsd:dateTime` (see the following table)
FLOAT	`xsd:float`
INTEGER	`xsd:int`
STRING	`xsd:string`

For DATETIME type, the simple type mapping is potentially changed by the presence of a Date Template value constraint as follows:

MRM DATETIME Date Template	Schema type
CCYY-MM-DDThh:mm:ss.s	`xsd:dateTime`
CCYY-MM-DD	`xsd:date`
CCYY-MM	`xsd:gYearMonth`
CCYY	`xsd:gYear`
--MM-DD	`xsd:gMonthDay`
--MM	`xsd:gMonth`
---DD	`xsd:gDay`
Thh:mm:ss.s	`xsd:time`

If the Date Template is not in the preceding list, the DATETIME is mapped to either an `xsd:time` or an `xsd:dateTime` with a BIP0175 warning message, depending on whether or not the Date Template had just a time component. However, note that this mapping can cause errors to appear in the task list after the import.

If the element concerned also had Version 2.1 `Default Value`, `Min Inclusive`, `Max Inclusive`, or `Enumeration` value constraints, the values for these do not match the lexical space for an `xsd:time` or `xsd:dateTime`, and so fail validation. These must be corrected manually using the editor.

The same task list error also appears for any Version 2.1 DATETIME type that supplied a `Default Value`, `Min Inclusive`, `Max Inclusive`, or `Enumeration` constraint where the value was not fully specified. For example, Date Template CCYY-MM, Enumeration 2003 was allowed in Version 2.1 as it was interpreted as 2003-01 at runtime. However, in the new model, the value must match the lexical space of the simple type and so must include the -01.

Equivalent customization variables in mqsicompcif and BIPEDIT

One of the steps involved in migrating a broker or User Name Server to Version 6.0 on z/OS is to customize the Version 6.0 BIPEDIT file using values that are defined in the Version 2.1 or Version 5.0 mqsicompcif file.

The following table lists the customization variables that are used in the Version 2.1 and Version 5.0 mqsicompcif file, and the equivalent variables in the Version 6.0 BIPEDIT file. You can also look at the sample BIPEDIT file.

Version 2.1 and Version 5.0 mqsicompcif customization variables	Version 6.0 BIPEDIT customization variables
DB2_SAMPLE_PGMNAME	++DB2SAMPLEPROGRAM++
DB2_SAMPLE_PGMPLAN	++DB2SAMPLEPROGRAMPLAN++
DB2_DSNACLI_PLANNAME	++DB2DSNACLIPLAN++
DB2_SUBSYSTEM	++DB2SUBSYSTEM++
DB2_LOCATION	++DB2LOCATION++
DB2_TABLE_OWNER	++DB2TABLEOWNER++ ++DB2CURRENTSQLID++ (These can be different)
DB2_STORAGE_GROUP	++DB2STORAGEGROUP++
DB2_STOR_GROUP_VCAT	++DB2LOCATION++
DB2_STOR_GROUP_VOL	Default value in BIPCRDB
DB2_DATABASE	++DB2DATABASE++
DB2_BUFFER_POOL	++DB2BUFFERPOOL++ ++DB2INDEXBP++ ++DB2LOBBP++ (These can be different)
COMPONENT_PDS	++COMPONENTDATASET++
COMPONENT_KEY	++STARTEDTASKNAME++
STPN	++STARTEDTASKNAME++
USER_PROCLIB_PDS	N/A
STEPLIB_NAME	Defined in each JCL
JAVAHOME	++JAVA++
CLASSPATH	Default value in profile
PATH	Default value in profile
LIBPATH	Default value in profile
LILPATH	Default value in profile
DISTHUB_PATH	Default value in profile
LANG	++LOCALE++
LC_ALL	++LOCALE++

Version 2.1 and Version 5.0 mqsicompcif customization variables	Version 6.0 BIPEDIT customization variables
TZ	++TIMEZONE++
NLSPATH	Default value in profile
CONSOLE_NLSPATH	Default value in profile
MC_MESSAGES	Default value in profile
DB2_CONVERSION	++DB2CONVERSION++
USE_ARM	Default value in profile
ARM_ELEMENTNAME	Default value in profile
ARM_ELEMENTTYPE	Default value in profile
_CEE_RUNOPTS	N/A
NNSY_ROOT	Default value in profile
NNSY_CATALOGUES	Default value in profile
NN_CONFIG_FILE_PATH	NN_CONFIG_FILE_PATH in BIPBRPOF

//BIPXXXXX JOB 5655-G97,
// MSGLEVEL=(1,1),
// MSGCLASS=A,
// NOTIFY=&SYSUID
	Defined in each JCL

mqsicreate(broker/usernameserver) command	++INSTALL++
mqsicreate(broker/usernameserver) command	++COMPONENTDIRECTORY++
mqsicreate(broker/usernameserver) command	++COMPONENTNAME++
N/A	++HOME++
N/A	++OPTIONS++
STEPLIB	++WMQHLQ++
mqsicreate(broker/usernameserver) command	++QUEUEMANAGER++
N/A	++COMPONENTPROFILE++
N/A	++XMLTOOLKIT++
STEPLIB	++DB2HLQ++
STEPLIB	++DB2RUNLIB++
N/A	++MQPATH++

Assignments configuration data in an export file

This topic describes how to find the assignments configuration data in the XML file that is generated when you export everything in a Version 2.1 Control Center workspace by clicking **File → Export All in Workspace**. Specifically, the topic tells you how to find the following configuration data for each broker in the broker domain:

- The name of the broker
- The name of each message set that is assigned to the broker
- The name of each execution group within the broker

- For each execution group within the broker, the name of each message flow that is assigned to the execution group
- For each message flow assigned to an execution group, the following properties:
  – Additional instances
  – Commit count
  – Commit interval
  – Coordinated transaction

Here is an example of an export file. Only the beginning of the file is shown.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE XMI SYSTEM "mqsi.dtd">
<XMI xmi.version="1.0">
  <XMI.header>
    <XMI.documentation>
      <XMI.owner>name</XMI.owner>
Broker/a3916f02-f500-0000-0080-8818935dcec8
ExecutionGroup/23bd6f02-f500-0000-0080-8818935dcec8
MessageProcessingNodeType/24c55cb6-ea00-0000-0080-c5b38dd899ad
MessageProcessingNodeType/7e8b6bb6-ea00-0000-0080-c5b38dd899ad
MessageProcessingNodeType/c0656ab6-ea00-0000-0080-c5b38dd899ad
MessageProcessingNodeType/575960b6-ea00-0000-0080-c5b38dd899ad
TopicRoot/TopicRoot
PubSubTopology/PubSubTopology
MessageProcessingNodeType/2f11692e-e900-0000-0080-c5b38dd899ad
MessageProcessingNodeType/8322692e-e900-0000-0080-c5b38dd899ad
MessageProcessingNodeType/4e2f692e-e900-0000-0080-c5b38dd899ad
MessageProcessingNodeType/ff3c692e-e900-0000-0080-c5b38dd899ad
MessageProcessingNodeType/fba95b0e-ef00-0000-0080-aed9bbfe32dc
MessageProcessingNodeType/fc149e14-ef00-0000-0080-aed9bbfe32dc
MessageProcessingNodeType/857e100f-ef00-0000-0080-aed9bbfe32dc
    </XMI.documentation>
  </XMI.header>
  <XMI.content>
    <Broker icon="images/Broker.gif" creator="" version="" collectionPath=""
          versionTimestamp="" longDescription="" status="" versionCreator=""
          creationTimestamp="" shortDescription=""
          xmi.uuid="a3916f02-f500-0000-0080-8818935dcec8"
          xmi.id="a3916f02-f500-0000-0080-8818935dcec8" xmi.label="fred_tp">
      <Broker_queueManager value="ANAME" encoded="false"
            xmi.label="queueManager" type="String" valueMandatory="true"/>
      <AssignedMsgProject icon="images/MRProject.gif" collectionPath="MRM"
            status="" xmi.label="MQSISTR_MRM2">
        <MRProjectRef icon="images/MRProject.gif" xml:link="simple"
              xmi.label="MRProject" type="MRProject" refType="MRProject"
              title="MQSISTR_MRM2"
              href="MRM/MRProject/71bb0097-e700-0000-0080-abda7687be69"/>
      </AssignedMsgProject>
      <ExecutionGroupRef icon="images/ExecutionGroup.gif" xml:link="simple"
            xmi.label="ExecutionGroup" type="ExecutionGroup"
            refType="ExecutionGroup"
            href="ExecutionGroup/23bd6f02-f500-0000-0080-8818935dcec8"
            title="default"/>
    </Broker>
    <ExecutionGroup icon="images/ExecutionGroup.gif" creator="" version=""
          collectionPath="" versionTimestamp="" longDescription="" status=""
          versionCreator="" creationTimestamp="" shortDescription=""
          xmi.uuid="23bd6f02-f500-0000-0080-8818935dcec8"
          xmi.id="23bd6f02-f500-0000-0080-8818935dcec8" xmi.label="default">
      <AssignedMessageFlow icon="images/MessageProcessingNodeType.gif" status=""
            xmi.label="mqsistrmrmswiftflow">
        <AssignedMessageFlow_additionalInstances value="3" encoded="false"
              xmi.label="additionalInstances" type="Integer"
              valueMandatory="false"/>
        <AssignedMessageFlow_commitCount value="5" encoded="false"
```

```
            xmi.label="commitCount" type="Integer" valueMandatory="false"/>
      <AssignedMessageFlow_commitInterval value="1" encoded="false"
            xmi.label="commitInterval" type="Integer" valueMandatory="false"/>
      <AssignedMessageFlow_coordinatedTransaction value="no" encoded="false"
            xmi.label="coordinatedTransaction" type="yes_no"
            valueMandatory="false"/>
      <MessageProcessingNodeTypeRef icon="images/MessageProcessingNodeType.gif"
            xml:link="simple" xmi.label="MessageProcessingNodeType"
            type="MessageProcessingNodeType" refType="MessageProcessingNodeType"
            href="MessageProcessingNodeType/24c55cb6-ea00-0000-0080-c5b38dd899ad"
            title="mqsistrmrmswiftflow"/>
    </AssignedMessageFlow>
  </ExecutionGroup>
  ...
 </XMI.content>
</XMI>
```

The following sections describe how to find the assignments configuration data in an export file.

The name of the broker

For each broker in the broker domain, there is a Broker element within the XMI.content element. The name of the broker is the value of the Broker element's `xmi.label` attribute.

In the example, there is only one broker. The name of the broker is fred_tp.

The name of each message set that is assigned to the broker

For each message set that is assigned to a broker, there is an AssignedMsgProject element within the Broker element. The name of the message set is the value of the AssignedMsgProject element's `xmi.label` attribute.

In the example, only one message set is assigned to the broker fred_tp. The name of the message set is MQSISTR_MRM2.

The name of each execution group within the broker

For each execution group within a broker, there is an ExecutionGroupRef element within the Broker element. The name of the execution group is the value of the ExecutionGroupRef element's `title`.

In the example, the broker fred_tp has only one execution group, which is the default execution group.

For each execution group in the broker domain, there is an ExecutionGroup element within the XMI.content element. You can determine the broker to which an execution group belongs by examining the value of the ExecutionGroup element's `xmi.uuid` attribute. This value forms part of the value of the `href` attribute on the Broker element's corresponding ExecutionGroupRef element.

In the example, there is only one execution group in the broker domain, the default execution group for the broker fred_tp. The value of the `xmi.uuid` attribute on the ExecutionGroup element is 23bd6f02-f500-0000-0080-8818935dcec8, and this forms part of the value of the ExecutionGroupRef element's `href` attribute for the broker fred_tp.

The name of each message flow that is assigned to each execution group

For each message flow that is assigned to an execution group, there is an AssignedMessageFlow element within the ExecutionGroup element. The name of the message flow is the value of the AssignedMessageFlow element's `xmi.label` attribute. In the example, only one message flow is assigned to the default execution group for the broker fred_tp. The name of the message flow is mqsistrmrmswiftflow.

The properties of each message flow

For each property of an assigned message flow, there is a corresponding element within the AssignedMessageFlow element. The following table lists each property and its corresponding tag:

Property of an assigned message flow	Corresponding tag
Additional instances	<AssignedMessageFlow_additionalInstances>
Commit count	<AssignedMessageFlow_commitCount>
Commit interval	<AssignedMessageFlow_commitInterval>
Coordinated transaction	<AssignedMessageFlow_coordinatedTransaction>

The value of a property is the value of the corresponding element's `value` attribute.

For example, the following table displays the properties of the message flow mqsistrmrmswiftflow that is assigned to the default execution group within broker fred_tp:

Property	Value
Additional instances	3
Commit count	5
Commit interval	1
Coordinated transaction	no

Migration glossary

This glossary lists differences in terminology between WebSphere Message Broker Version 6.0 and previous versions of the product.

For a full list of terms and abbreviations used by the product documentation, and for definitions that are more complete, see the Glossary of terms and abbreviations.

A

assigned message flow
In Version 2.1, a message flow that is assigned to an execution group. In Version 6.0, when you add a message flow to a broker archive (bar) file and deploy the bar file to an execution group, the message flow is assigned implicitly to the execution group.

assigned message flow properties
In Version 2.1, the properties of an assigned message flow. In Version 6.0, you specify the equivalent properties in the deployment descriptor of a broker archive (bar) file.

assigned message set
In Version 2.1, a message set that is assigned to a broker. In Version 6.0, when you add a message set to a broker archive (bar) file and deploy the bar file to an execution group, the message set is assigned implicitly to the execution group.

Assignments view
In Version 2.1, the user interface in the Control Center that is used to assign message flows to an execution group and message sets to a broker. In Version 6.0, the equivalent function is provided by the Broker Administration perspective of the workbench. When you add message flows and message sets to a broker archive (bar) file and deploy the bar file to an execution group, the message flows and message sets are assigned implicitly to the execution group.

B

base message set
In Version 2.1, a finalized message set upon which another message set is based. The two message set definitions are linked and the definition of the base message set can never be modified. In Version 6.0, when you base a message set on another message set, the workbench makes a copy of the definition of the base message set. Subsequently, the two message set definitions are completely separate from each other and can be modified independently.

base type
In Version 2.1, a compound type as a whole can have a simple value. The type of this simple value is called the base type. In Version 6.0, a base type is the simple type or complex type upon which another type is based.

Broker Administration perspective
In Version 6.0, a broker administration console that communicates with one or more Configuration Managers. Administrators work in this perspective to manage the resources (also referred to as domain objects) in the broker domain that are defined on one or more Configuration Managers. In Version 2.1, the equivalent function is provided by the Assignments view.

built-in node
In Version 6.0, a message processing node that is supplied by the product. In Version 2.1, this is referred to as an IBM Primitive node.

C

check in
In Version 2.1, a Control Center action that stores a new or updated resource in the Configuration Manager. In Version 6.0, the equivalent check-in function might be provided by an external repository that is associated with the workbench.

check out
In Version 2.1, a Control Center action that extracts a resource from the Configuration Manager and locks it for local modification by a user. In Version 6.0, the equivalent check-out function might be provided by an external repository that is associated with the workbench.

C language binding
: See language binding.

COBOL language binding
: See language binding.

complex type
: In Version 6.0, a type that has structure. In Version 2.1, this role is performed by a compound type.

compound type
: In Version 2.1, this has two roles:

- A type that has structure. In Version 6.0, this is referred to as a complex type.
- A collection of elements. In Version 6.0, this is referred to as a group.

Composition
: In Version 6.0, a property that describes how a message tree is structured. In Version 2.1, the equivalent property of a compound type is Type Composition.

Configuration Manager
: In Version 6.0, the Configuration Manager does not store message flow and message set definitions. Instead, these definitions are stored in the local file system or in an external repository that is associated with the workbench. In addition in Version 6.0 the Configuration Manager no longer uses an external database to store the domain configuration, but uses an internal repository instead.

connection
: In Version 6.0, this represents the flow of control and data between two message processing nodes. In Version 2.1, this is referred to as a connector.

Connection pane
: In Version 2.1, a pane that is used to specify certain properties of an element or compound type. In Version 6.0, these properties are specified as the logical properties of an element reference or group reference.

connector
: In Version 2.1, this represents the flow of control and data between two message processing nodes. In Version 6.0, this is referred to as a connection.

Content Validation
: In Version 6.0, a property of a group. In Version 2.1, the equivalent property applies to a compound type and is called Type Content.

context tag
: See element qualifier.

Control Center
: In Version 2.1, the graphical user interface that provides the function to define, configure, deploy, and monitor resources. In Version 6.0, this function is provided, with enhancements, by the Message Brokers Toolkit, which is also known as the workbench.

customizer
: In Version 2.1, a means of specifying complex properties of a message processing node. In Version 6.0, you use an editor to perform this task.

D

debugger
: In Version 2.1, a tool that is accessed from the Message Flows view in the Control Center and enables message flows to be debugged. In Version 6.0, similar but enhanced function is provided by the flow debugger.

Description pane
: In Version 2.1, a pane that is associated with every object and that is used to specify the Short Description and Long Description properties of the object. In Version 6.0, some objects still have short and long descriptions, but message model objects have a documentation property instead.

documentation property
: In Version 6.0, the documentation property of an object allows you to add information to enhance the understanding of that object's function. In Version 2.1, the equivalent information is provided using the Description pane.

E

element member
: In Version 2.1, an object that represents the membership of an element within a compound type. In Version 6.0, the object that represents the membership of a global element within a complex type or group is called an element reference.

element qualifier
: In Version 2.1, an object that is associated with an element in a message to enable that element to be treated differently in different contexts. It is also known as a context tag. In Version 6.0, element qualifiers are no longer used.

element reference
: In Version 6.0, an object that represents the membership of a global element within a complex type or group. In Version 2.1, the object that represents the membership of an element within a compound type is called an element member.

element value
: In Version 2.1, a named value that you can use to specify a value constraint on an element. In Version 6.0, element values are no longer used. Instead, you associate value constraints with a simple type.

extended decimal
: In Version 2.1, a Custom Wire Format (CWF) physical representation of a number. In Version 6.0, this is referred to as external decimal.

external decimal
: In Version 6.0, a Custom Wire Format (CWF) physical representation of a number. In Version 2.1, this is referred to as extended decimal.

F

finalize
: In Version 2.1, the action to put a message set into a finalized state. When a message set is in this state, you can never modify its definition again. In Version 6.0, a message set cannot be finalized.

flow debugger
: In Version 6.0, a facility to debug message flows that is provided in the Debug perspective in the workbench. In Version 2.1, the equivalent function is provided by the debugger .

freeze In Version 2.1, the action to put a message set into a frozen state to prevent further modification to its definition. See also unfreeze. In Version 6.0, a message set cannot be frozen.

G

glossary
In Version 2.1, a form of documentation in HTML format that you can generate from a message set definition. In Version 6.0, this has been replaced by message set documentation, which is also in HTML format.

group In Version 6.0, a structure that can be included in a complex type or another group. In Version 2.1, this role is performed by a compound type.

H

History pane
In Version 2.1, a pane that is used to specify additional documentation for certain MRM objects such as an element, a message, and a compound type. In Version 6.0, the Documentation property of an object is used to specify this information.

I

IBM Primitive node
In Version 2.1, a message processing node that is supplied by the product. In Version 6.0, this is referred to as a built-in node.

identifier
In Version 2.1, an identifier that identifies each object uniquely within a message set. You use the identifier to reference the object in an ESQL statement. In Version 6.0, an object is identified by its name only. When you migrate a message set, the name of an object is formed from its Version 2.1 identifier.

Input Compression Technique
In Version 2.1, a Tagged Delimited String (TDS) property of a message set. In Version 6.0, this property does not exist.

Input Terminal node
In Version 2.1, an IBM Primitive node through which a message is received by a subflow. In Version 6.0, this function is provided by the Input node.

L

language binding
In Version 2.1, a set of properties associated with a message set that enable you to generate a C header or COBOL copy book from the message set definition. In Version 6.0, this set of properties does not exist, and it is not possible to generate a C header or COBOL copy book from a message set definition.

Level In Version 2.1, a property of a message set that specifies the version of the message set. In Version 6.0, this property does not exist.

Long Description
See Description pane.

M

message book
In Version 2.1, a form of documentation in HTML format that you can generate from a message set definition. In Version 6.0, this has been replaced by message set documentation, which is also in HTML format.

Message Brokers Toolkit
In Version 6.0, the development and operational management environment for a broker domain. It is also referred to as the workbench. In Version 2.1, the Control Center provides this function.

message category
In Version 6.0, a way of grouping messages, perhaps for documentation or convenience purposes, or for assisting in the generation of Web Services Description Language (WSDL) files. This replaces the Version 2.1 transaction and transaction category.

message repository
In Version 2.1, the set of database tables used by the Configuration Manager to store message set definitions. In Version 6.0, there is no message repository, and message set definitions are stored in the local file system or in an external repository that is associated with the workbench.

Message Repository Manager
In Version 2.1, the component of the Configuration Manager that manages message set definitions in the message repository. In Version 6.0, there is no Message Repository Manager.

message set documentation
In Version 6.0, documentation in HTML format that you can generate from a message set definition. In Version 2.1, the equivalent documentation is provided by a glossary and a message book.

MRM In Version 2.1, the name of a message domain and the abbreviation for Message Repository Manager. In Version 6.0, it is the name of only a message domain.

N

name In Version 2.1, a descriptive name of an object in a message set. This name is not used to identify the object in an ESQL statement; the identifier of the object is used instead. In Version 6.0, the name is used to identify an object and must be a valid XML Schema identifier. When you migrate a message set, the name of an object is formed from its Version 2.1 identifier.

O

Operations view
In Version 2.1, the Control Center view that is used to manage and monitor the brokers in a broker domain. In Version 6.0, the equivalent function is provided by the Domains and Alerts views in the Broker Administration perspective of the workbench.

Output Compression Technique
In Version 2.1, a Tagged Delimited String (TDS) property of a message set. In Version 6.0, this property does not exist.

Output Terminal node
In Version 2.1, an IBM Primitive node through which a message is propagated by a subflow. In Version 6.0, this function is provided by the Output node.

P

plug-in
In Version 2.1, a user-written extension to the broker that provides a message processing node or message parser in addition to those supplied with the product. In Version 6.0, this is referred to as a user-defined extension.

prefixed identifier
In Version 2.1, an identifier that contains a prefix to ensure that the identifier is unique within a message set. Use prefixed identifiers in situations where different elements have the same name so that the element is defined globally even though it is considered to be local to a compound type of which it is a member. In Version 6.0, prefixed identifiers are no longer required because an element can be defined locally. When you migrate a message set, an element with a prefixed identifier becomes a local element, and the prefix is discarded when the identifier is used to form the Version 6.0 name.

R

Run Time pane
In Version 2.1, the pane that is used to specify the runtime parser for a message set. In Version 6.0, this information is specified as a general message set property.

S

shared data
In Version 2.1, configuration repository and message repository data that is shared by Control Center users. In Version 6.0, the equivalent functionality might be provided by an external repository that is associated with the workbench.

Short Description
See Description pane.

simple type
In Version 2.1, one of seven types defined by the MRM to describe an item of data. In Version 6.0, a simple type is one of the 44 XML Schema simple types or a user-defined simple type.

Simple Unordered Set
In Version 2.1, one of the permissible values of the Type Composition property of a compound type. It is intended for the modelling XML DTD attribute list declarations. In Version 6.0, this value is not required because XML Schema attribute groups are modelled as objects in their own right. If the Type Composition property of a compound type has this value when you migrate a message set, the Composition property of the group that is formed from the compound type has the value All, if the Type Content property of the compound type has the value Closed, or the value Unordered Set otherwise.

suspended from use
In Version 2.1, certain MRM objects in a message set that is based on another message set can be suspended from use. When suspended from use, an object cannot be used. In Version 6.0, an object cannot be suspended from use.

T

traffic light status icon
In Version 2.1, an icon next to a message flow, execution group, or broker in the Domain Topology pane of the Operations view in the Control Center. A green traffic light indicates that the resource is running, and a red traffic light indicates that it is not running. In Version 6.0, an alert appears in the Alerts view in the Broker Administration perspective of the workbench only when the resource is not running.

transaction
In Version 2.1, a group of messages that are considered to be part of a single business transaction. In Version 6.0, transactions are no longer used. When you migrate a message set, a transaction is replaced by a message category that contains the same messages as the original transaction.

transaction category
In Version 2.1, a group of transactions within a message set. In Version 6.0, transaction categories are no longer used. When you migrate a message set, a transaction category is replaced by a message category. This message category contains all the messages in all the transactions that were members of the original transaction category.

Type Composition
In Version 2.1, a property of a compound type. In Version 6.0, the equivalent property applies to a group and is called Composition.

Type Content
In Version 2.1, a property of a compound type. In Version 6.0, the equivalent property applies to a group and is called Content Validation.

U

unfreeze
In Version 2.1, the action to remove a message set from a frozen state to allow further modification to its definition. In Version 6.0, a message set cannot be frozen.

unlock
In Version 2.1, the action to end the checked-out state of a resource in such a way that any changes made to the resource since it was checked out are not stored in the Configuration Manager. In Version 6.0, the equivalent function might be provided by an external repository that is associated with the workbench.

user-defined extension
In Version 6.0, a user-written extension to the broker that provides a message processing node or message parser in addition to those supplied with the product. In Version 2.1, this is referred to as a plug-in.

V

value constraint
In Version 2.1, the use of an element value to specify a constraint on the value of an element. In Version 6.0, a value constraint is represented by an XML schema facet that is applied to a simple type.

view
In Version 2.1, a user interface in the Control Center that enables a user to perform a specified task. There are eight views: Message Sets, Message Flows, Topology, Assignments, Topics, Operations, Subscriptions, and Log. In Version 6.0, a view is a component of a perspective in the workbench.

To perform a specified task, the user selects the appropriate perspective and then uses a combination of one or more views and an editor.

W

workbench
See Message Brokers Toolkit.

Part 3. Appendixes

Appendix. Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this information in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this information. The furnishing of this information does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032,
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 information. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this information at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

(C) (*your company name*) (*year*). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _*enter the year or years*_. All rights reserved.

Trademarks

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

AIX	CICS	Cloudscape
DB2	DB2 Connect	DB2 Universal Database
developerWorks	Domino	
Everyplace	FFST	First Failure Support Technology
IBM	IBMLink	IMS
IMS/ESA	iSeries	Language Environment
Lotus	MQSeries	MVS
NetView	OS/400	OS/390
POWER	pSeries	RACF
Rational	Redbooks	RETAIN
RS/6000	SupportPac	S/390
Tivoli	VisualAge	WebSphere
xSeries	z/OS	zSeries

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

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

Intel and Pentium are trademarks or registered trademark of Intel Corporation or its subsidiaries in the United States and other countries.

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

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

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

Index

Printed in USA