MQSeries® Integrator

Using the Control Center

Version 2.0.2

SC34-5602-03

MQSeries® Integrator

Using the Control Center

Version 2.0.2

SC34-5602-03

Note!

Before using this information and the product it supports, be sure to read the general information under “Appendix. Notices” on page 163.

Fourth Edition (May 2001)

This edition applies to IBM® MQSeries Integrator Version 2.0.2 and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

Figures

Tables

About this book

This book describes how to use theMQSeries Integrator Version 2.0.2 Control Center.

This edition of the book has been restructured, and readers of earlier editions will see that the retail scenario and ESQL appendices are no longer included. The retail scenario is now included in the SupportPac™ IC03, and ESQL is described in a separate book, *MQSeries Integrator ESQL Reference*.

MQSeries Integrator Using the Control Center has been structured to make the information in it easy to access. The details of the tasks that you need to perform, and their execution are shown in the first part of this book, whilst the second part describes the concepts that lie behind the tasks. This latter section is provided for reference purposes: when you start using this book you will probably already have met most of the concepts involved if you have read the books described in "What you need to know to understand this book".

A glossary and bibliography are provided at the end of this book.

Who this book is for

This book is intended for anyone who needs to use the Control Center to perform the tasks described above.

The first part of this book describes how to perform your tasks using this tool, while the second part explains both the concepts behind the tool, and useful reference information.

What you need to know to understand this book

You need to have read and understood the general introduction to all aspects of MQSeries Integrator in the *MQSeries Integrator Introduction and Planning* book.

Before you can start to use the Control Center, you must have successfully completed the following tasks:

- MQSeries Integrator must have been installed.
- Users and groups must have been added to MQSeries Integrator security groups.
- All databases required by MQSeries Integrator must have been created, and users and groups authorized to use them.
- A Configuration Manager must have been created. A broker must also have been created if you will be deploying data.
- If you are using ACLs on publish/subscribe topics, a User Name Server must have been created and started.
- The MQSeries resources required to connect the queue managers hosting MQSeries Integrator components must have been defined.
- Listeners for the queue managers must have been started.
- The Configuration Manager must have been started.

For more information about these tasks, see the MQSeries Integrator installation guide for your operating system.

For information about messages, their structure and definition, please see the *MQSeries Integrator ESQL Reference* book.

Terms used in this book

All references in this book to MQSeries Integrator are to MQSeries Integrator Version 2.0 unless otherwise stated.

All references in this book to Windows NT® are also applicable to Windows® 2000 unless otherwise stated. MQSeries Integrator components that are installed and operated on Windows NT can also be installed and operated on Windows 2000.

Migration

If you are migrating from an earlier version of MQSeries Integrator then please see the migration information in the *MQSeries Integrator Administration Guide*.

Summary of changes

This section describes changes in this edition of *MQSeries Integrator using the Control Center* . Changes since the previous edition of the book are marked by vertical lines to the left of the changes.

Changes for this edition (SC34-5602-03)

- This book has been reformatted so that all the tasks you can perform using the Control Center are described in the first part of the book. The second half of the book provides reference and concept information.
- The chapter on ESQL, and much of the background information on message structure has been moved to the *MQSeries Integrator ESQL Reference*
- The section “Chapter 17. Concepts of NEONRules and NEONFormatter Support for MQSeries Integrator” on page 145 is new.
- The description of Control Center tour has been moved to the *MQSeries Integrator Introduction and Planning* book.
- The new debugger facility is described in “Chapter 11. Debugging message flows” on page 119 and “Chapter 15. Concepts of debugging” on page 135.
- The cut and duplicate functions have been removed, and copy and paste now work more intuitively.
- The function of the message flow import and export has been enhanced.
- You are now able to change the orientation of terminals on nodes in a **Message Flows** pane.
- Log view has been redesigned.
- The lock menu has been removed, however the check-in, check-out, and unlock menus remain so there is no loss of function.
- You can now double-click on a resource in a list to add it to a workspace.
- You can double-click on items in the Views to show the property dialog.
- It is now possible to create bend points when connecting message nodes.
- When using the Message Flow and Topology panes the settings for zoom, Manhattan, and snap to grid can be saved between sessions.
- It is now possible to add a description of the connection between message flow nodes.
- A progress indicator is now shown when checking-in, saving a workspace, importing and exporting.
- The addition of new NEON nodes:
  - NEONTransform
  - NEONMap
  - NEONRulesEvaluation

  the NEONFormatter and NEONRules nodes become obsolete in this release, and are only included for backwards compatibility.
- The MQeInput, MQeOutput, SCADAInput and SCADAOutput nodes are all new for this release.

The second edition (SC34-5602-02) of this book included new a section ″Copying messages between parsers″ which can now be found in the *MQSeries Integrator ESQL Reference* book .

Changes

Changes for the first edition (SC34-5602-01) of this book include:

- Additional information to cover the following product changes:
  - New products MQSeries Integrator for AIX® Version 2.0.1 and MQSeries Integrator for Sun Solaris Version 2.0.1.
  - New IBM primitive nodes (FlowOrder, Label, and RouteToLabel)
- Extended scenario showing additional function
- Extended examples of ESQL usage
- Extended information about supported messages
- Minor technical and editorial improvements throughout the book

Part 1. Using the Control Center

Chapter 1. Tasks

Depending on the tasks and the level of experience, more or less information will be required when identifying and implementing tasks. To help in this process we have separated the different levels of information so you can find what you need more easily.

If you are unfamiliar with MQSeries Integrator then you will need to understand the sorts of tasks that you can perform using the Control Center. The primary source for this information is this book, and a summary of all the Control Center tasks are described in "Tasks described in this book".

Note: If you want to create your own plug-in nodes and carry out the tasks for installation, this is described in the *MQSeries Integrator Programming Guide*.

Once you have identified the tasks that you need to perform, you can use the online help to

- Populate the views of the Control Center
- Develop, design and connect resources you create

When developing message flows you might decide to use complex nodes such as the Compute node. Complex nodes need to be customized to suit your environment. This manipulation is done using ESQL which is described in the *MQSeries Integrator ESQL Reference*.

SupportPacs

- SupportPac IC03 describes business scenarios, for example a retail scenario, which includes sample code and worked examples in addition to documentation.
- SupportPac IC04 provides suggested procedures for version management and change control of MQSeries Integrator Version 2.0 objects.
- SupportPac IH03 contains a GUI based utility that is useful for the development and testing of MQSeries Integrator Version 2.0 applications. Test messages are stored as files, which are then read by the application and written to an MQSeries queue.
- SupportPac ID03 MQSeries Integrator - Working with MQSeries Everyplace and SCADA, contains information about using MQSeries Everyplace and SCADA.

See "MQSeries information available on the Internet" on page 173 for more details.

Tasks described in this book

This part of this book describes the tasks and sub-tasks that you might wish to perform when using the Control Center.

The main tasks, which are summarized in this chapter, are:

SupportPacs

- "Chapter 8. Deploying configuration data" on page 93
- "Chapter 9. Setting up publish/subscribe access control lists" on page 101
- "Chapter 10. Running the broker domain" on page 107
- "Chapter 11. Debugging message flows" on page 119

Note: Authority to perform tasks
It is important to have the appropriate level of authorization to perform each of these tasks. For more information see "Managing permissions to Control Center tasks" on page 11.

Apart from starting and exiting from the Control Center, "**Getting started with the Control Center**", describes:

- The naming conventions for Control Center resources
- How to authorize Users to different roles, and manage access to tasks according to role responsibilities

"**Using the Workspace**" describes the tasks you might wish to perform on a workspace including:

- "Create a new workspace" on page 15
- "Open your existing workspace" on page 15
- "Saving your workspace locally" on page 16
- "Update your workspace" on page 16
- "Import resources" on page 19
- "Export resources" on page 20

The concepts of using a workspace are described in "The workspace" on page 127.

The "**Defining messages**" chapter shows you how to create and manage message sets. It describes both the Message Repository Manager (MRM), (bottom-up approach, and SmartGuide message creation. The latter method:

- Creates message type automatically
- Can create elements and lengths
- Can reorder elements
- Means that you will need to set Connection and Custom Wire attributes after the message is created

The chapter also describes:

- "Importing message definitions" on page 39
- "Generating MRM message set definitions as XML DTDs" on page 40
- "Generating language bindings" on page 40
- "Generating documentation" on page 42

"**Defining message flows**" describes how to create and update a message flow, how to tailor a view of a flow to meet your requirements, and describes the message flow nodes available. An example of how to use each node is also given. For information about parameters (for example) used by these nodes, see the online help.

For more information about message flow concepts, see "Chapter 13. Concepts of message flows" on page 129.

"**Defining the broker topology**" describes how to set up your broker environment so that you will be ready for "**Assigning resources to a broker**".

"**Deploying configuration data**" describes how to implement your new configuration, by transmitting it to the relevant brokers.

For more information about the concepts of deployment, see "Chapter 14. Concepts of deployment" on page 131.

"**Setting up publish/subscribe access control**" describes how to:

- Create a new publish/subscribe topic
- Manage access control lists
- Publish messages
- Request persistent delivery of messages
- Subscribe to topics

"**Running the broker domain**" describes how to monitor the status of brokers and message flows while they are running. This chapter also describes how to use user traces, and covers problem determination.

"**Debugging message flows**" describes how to use the debugging tool provided with MQSeries Integrator Version 2.0.2.

Chapter 2. Getting started with the Control Center

This chapter describes:

.

Starting the Control Center

To start the Control Center, you can:

- Double click the **Control Center** icon in the MQSeries Integrator product folder on your desktop.

or

- From the **Start** menu, its **Programs** —>**MQSeries Integrator 2.0.** —> **Control Center**

When you start the Control Center, the **Configuration Manager Connection** dialog is displayed. To complete the dialog:

- In the Host Name field, type the network host name of the system on which the Configuration Manager has been created.
- In the Port field, type the port number on which the queue manager hosting the Configuration Manager is listening. The default port number is 1414. (You can find out the port number to enter here from **MQSeries Services**. Right click the listener associated with the queue manager, select **Properties** and click the **Parameters** tab to display the port number.) No other queue manager must be listening on this port.
- In the Queue Manager Name field, type the name of the queue manager hosting the configuration manager.
- Click **OK**.

After you have started the Control Center and connected to the Configuration Manager, you can update these connection details. To do this, click **Connection** from the Control Center **File** menu. The **Configuration Manager Connection** dialog is displayed. You can alter the values displayed in this dialog, and click **OK** to apply the new values. If you do this from an unsaved workspace, you are given the opportunity to save the workspace before changing the connection information.

If you change your mind about the values you have typed in the dialog and have not clicked **OK** to apply them, click **Reset** to restore the values with which you connected to the Configuration Manager.

Note: You must not alternate between alias names for the Host Name value. If you connect using a different alias for the same host, you get a different local configuration that is unique to the alias name. You will no longer be able to access resources you created in your original local configuration, nor will you be able to check in any resources checked out to the original local configuration.

Starting the Control Center

When you start the Control Center subsequently, the fields in the **Configuration Manager Connection** dialog display the values you supplied when you last connected to the Configuration Manager.

The Control Center interface looks something like this:

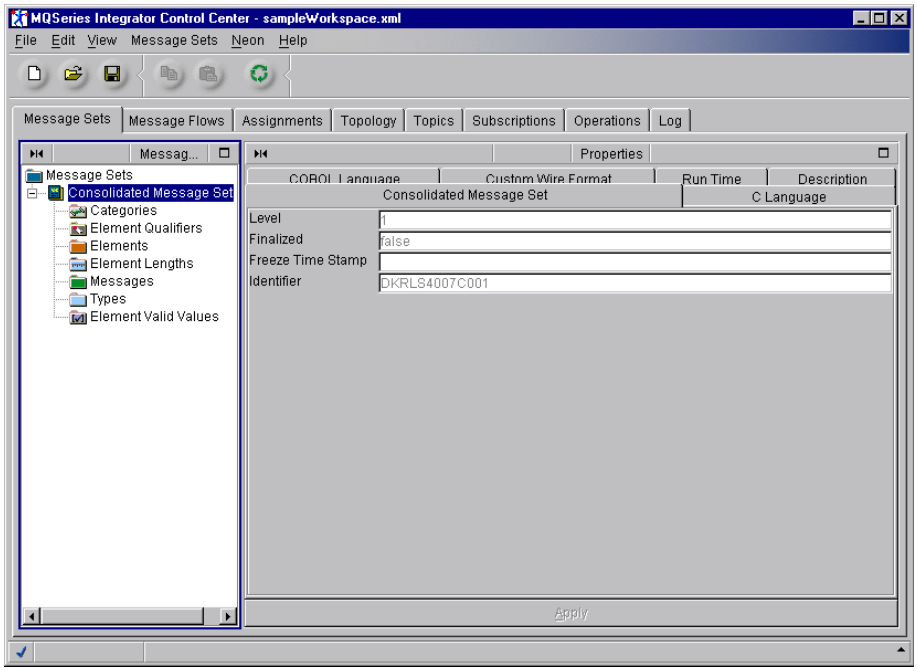


Figure 1. The Control Center

You see all the tabs shown here if your user role is **All roles**. User roles are described in “Managing permissions to Control Center tasks”.

Naming Control Center resources

There are some rules you must follow when providing names for the resources you create using the Control Center:

- You can use the characters:
  - Uppercase A — Z
  - Lowercase a — z
  - Numerics 0 — 9
  - The special characters $ % ‘ ’ - _ @ ~ ! ( ) { } [ ] ^ # & + , ; =
- You can also use the space character, and any Unicode character, except control codes and reserved characters, with an ASCII value greater than 127 (X'7F').

More specific guidelines exist on some operating systems. For example, names on UNIX® systems (AIX, HP-UX, and Sun Solaris) are case sensitive, but on Windows NT they are not. Therefore you must ensure that names specified in the Control Center, for example broker names, exactly match the names used on the system on which they are created. For more information about these guidelines, see the *MQSeries Integrator Administration Guide*.

Managing permissions to Control Center tasks

The Control Center supports many different tasks that you perform when working with configuration data or monitoring operational brokers. These tasks are grouped by *user role*, as follows:

Managing permissions to Control Center tasks

Message flow and message set developer
Can create message flows and message sets.

Message flow and message set assigner
Can create execution groups within brokers, assign message flows to execution groups, and assign message sets to brokers. Can also deploy this data.

Operational domain controller
Can create brokers and collectives, and define the relationships between them (the topology). Can also deploy all types of data, and monitor the operational broker domain.

Topic security administrator
Can create topics and associated ACLs. Can also deploy this data.

How to select the role you want to adopt is described in “Setting user roles” on page 13.

According to the role you select, the Control Center displays only those views or tabs that are relevant to that role, as follows:

- The **Message flow and message set developer** sees the *Message Sets* view and the *Message Flows* view.
- The **Message flow and message set assigner** sees the *Assignments* view only.
- The **Operational domain controller** sees the *Topology* view, the *Assignments* view, the *Topics* view, the *Operations* view, and the *Subscriptions* view.
- The **Topic security administrator** sees the *Topics* view and the *Topology* view.

If you want to perform all tasks, you should select All roles, which allows you to see all available views.

The role you select for yourself only configures what you *see* on the Control Center. It does not control the type of objects you can view or modify. For security purposes, this aspect is controlled by the MQSeries Integrator security groups of which you are a member.

The MQSeries Integrator security groups, and the Control Center tasks that membership of those groups allows, are:

mqbrdevt
Members of this group can design message sets and message flows.

mqbrasgn
Members of this group can manage execution groups within brokers; view message sets and message flows; assign message flows to execution groups; and assign message sets to brokers.

mqbrops
Members of this group can create and delete brokers; deploy, start, and stop message flows; start and stop trace activity on message flows; manage and deploy the broker domain topology, including collectives; view the whole deployed system, including message sets, message flows, and subscriptions; deploy topics; and view logs that report on the deployment activity.

mqbrtpic
Members of this group can manage topics, and the access control lists for the topic tree; deploy topics; view the logs that report on that deployment activity.

The Configuration Manager performs a security check based on the above whenever a Control Center user attempts to view or modify an object in the configuration and message repositories.

Adding users and groups to the MQSeries Integrator groups

The method for adding users and groups to MQSeries Integrator is described here for both Windows NT and Windows 2000.

Using Windows NT

You must use the Windows NT User Manager to add users and groups to the MQSeries Integrator security groups, as follows:

1. Invoke the Windows NT User Manager by selecting **Start —> Programs —> Administrative Tools (Common) —> User Manager**.
2. Double click the MQSeries Integrator group you want to update.
3. Either double-click on the user ID or
   - Select **Add**. From the list of available user IDs, select the user ID to be added to the group
   - Click **Add**. Click **OK** to close the **Add Users and Groups**
   - Click **OK** to close the **Local Group Properties** dialog.
4. Close the User Manager.

Using Windows 2000

To add users and groups to the MQSeries Integrator Groups:

1. Invoke Start -> Settings -> Control Panel -> Users and Passwords
2. Go to the Advanced tab
3. Select **Advanced User Management**
4. Select **groups** from the Local Users and Groups window
5. Double click on each of the MQSeries Integrator groups and select to add users

The authorization is effective after a delay of approximately five minutes, as the Configuration Manager caches this information.

For more detailed information about the MQSeries Integrator groups, and about security in general, see the *MQSeries Integrator Introduction and Planning*.

Setting user roles

At any time during a Control Center session, you can change your user role. To set your Control Center user role:

1. From the **File** menu in the Control Center menu bar, click **Preferences** and then **Control Center**.

   The **Control Center Preferences** dialog is displayed.
2. In the page, which is collapsed by default, click on **User roles**

Setting Control Center preferences

You can select only one role at a time.

Setting user roles

User preferences that govern the general appearance of the Control Center can also be set by selecting **File —> Preferences...**. For more information, see the Control Center online help.

You can also control the presence of the Log view from the Control Center Preferences dialog. The default setting is that the Log view is always visible, regardless of the user role currently in force. You can change the **Show Log** setting in the **Control Center** page if you want to suppress the Log view.

The setting of this option is remembered from session to session.

Managing permissions for MQSeries brokers

Please see *MQSeries Integrator Administration Guide* for information about managing permissions for MQSeries brokers.

Managing permissions for databases

Please see *MQSeries Integrator Administration Guide* for information about managing permissions for databases.

Access to Neon tools

There is a **Neon** menu item provided in the Control Center Message Sets pane. This gives you access to the NEONFormatter, NEONRules, and Visual Tester.

See the NEON books for more information as NEON tools are not described in the MQSeries Integrator documentation.

Exiting the Control Center

To exit the Control Center, click **Exit** from the **File** menu in the menu bar. You are prompted to save any unsaved work before exiting.

Chapter 3. Using the workspace

If you want to create new configuration resources, you use an operation called *create*. This creates a new object within the Control Center and adds a reference to it to your workspace.

At this point, your new object does not exist in the shared configuration. To make your object visible in the shared configuration, you use an operation called *check in*. Once an object has been checked in it becomes visible to all other Control Center users.

If you want to modify an object in the shared configuration, you use an operation called *check out*. This locks the object in the shared configuration, preventing other Control Center users from modifying it, and makes a copy in your Control Center.

When you have made your modifications, you save them back to the shared configuration using *check in*, which also unlocks the object so that others can modify it.

If you want to destroy an object, you use an operation called *delete*. The object is deleted from wherever it exists, which could be the Control Center (if the object is newly created), or the shared configuration. If the object that you delete is in the shared configuration, and another user has a copy of this object in a local workspace, that user is not notified that you have deleted the object unless a refresh is requested.

Create a new workspace

To create a new workspace, click **File —> New Workspace**.

A new workspace is created. This workspace is untitled, and displays the default contents. You specify a title for the workspace when you save it.

You are prompted to save any changes made in a previous workspace: if you do not do so, they are discarded.

Open your existing workspace

When you open an existing workspace, your workspace is populated with the resources from the chosen file, and these replace the contents of any previous workspace.

You are prompted to save any changes made in a previous workspace: if you do not do so, they are discarded.

To open an existing workspace:

1. Click **File —> Open Workspace**.

   The **Open** dialog is displayed.
2. Select the file (which must be a file of any name and extension that contains valid XML: typically the file will have an extension of `.xml`) from the list presented, or specify the name of the file in the File name field. Click **Open** to open the selected workspace.

Alternatively, if you have recently used the workspace, click the **File** menu in the Control Center menu bar, which displays the names of the most recently used workspaces. There can be up to four names in this list. If the workspace you want is listed in the **File** menu, click its name to open it.

Saving your workspace locally

When you save a workspace, both the workspace and any resources created or modified in that workspace are saved to the local configuration.

To save a workspace, click **File —> Save Workspace**. The workspace contents are saved to an XML file. If the workspace is untitled, you are prompted for a name. The name you give it is displayed on the Control Center title bar whenever that workspace is your current workspace.

Note: If you want to save all the resources ″behind the workspace″ then you must export them as described in “Export resources” on page 20.

To save a named workspace under a different name, click **File —> Save Workspace As**. The workspace contents are saved to an XML file of the specified name. This takes a copy of the workspace contents. While the save is in progress, the status bar at the bottom of the window, displays the percentage of work done, and text describing the work in progress.

You can have as many workspaces as you like, but you can have only one local configuration per shared configuration.

It is possible to switch between shared configurations on different Configuration Managers (for example, between a test and production system) using the **File —> Connection** dialog. For each such shared configuration, there is one local configuration.

Update your workspace

Before you can make significant updates to any resource in your workspace, you must check it out of the repository in which it is maintained. Message sets and all their components are maintained in the message repository, all other resources are maintained in the configuration repository. Resources that are currently checked out have the **Key** icon against their entries wherever they are displayed. They are locked to your user ID, thus preventing other users making changes to them.

You can also work with new resources, that you create within your workspace. New resources that have never been checked in have the **New** icon against their entries wherever they are displayed:

The two icons are shown in Figure 2.

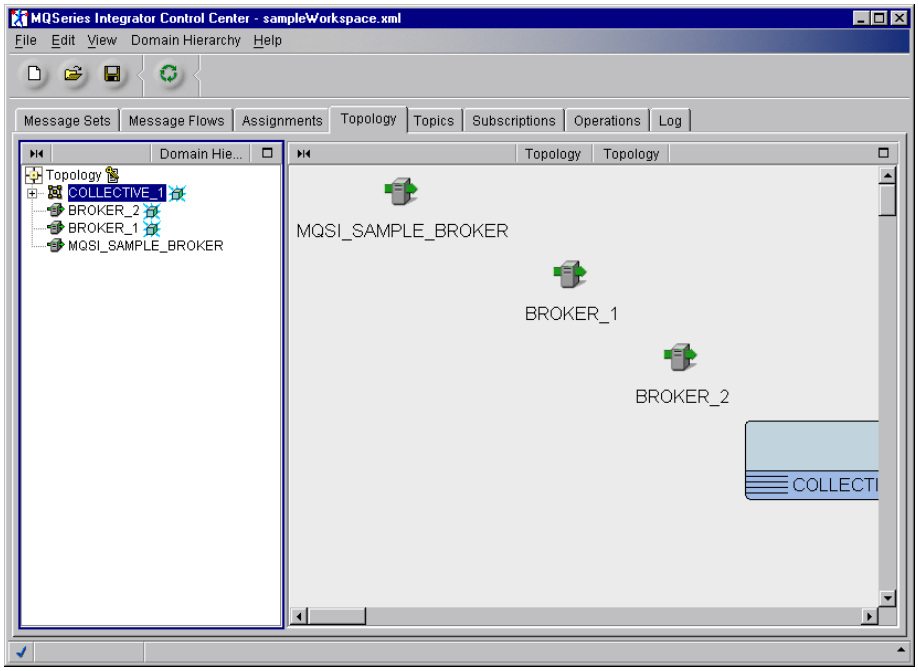


Figure 2. The Key icon and the New icon. The Topology is checked out, and brokers BROKER_1 and BROKER_2 and the collective have never been checked in. Broker MQSI_SAMPLE_BROKER is already checked in (and has no icon against it).

If you see an instance of an item flagged with an **X**, it is a ″stale reference″. A stale reference can occur when a message flow type is created and dropped on to a message flow. This message flow type is subsequently deleted by someone else on the server, and the name of an instance of an item on the pop-up menu is flagged with an **X**.

You can also take the following actions for the workspace itself:

- “Revert your workspace to the shared repository”
- “Save your workspace to the shared repository” on page 18

Revert your workspace to the shared repository

When you revert your workspace to the shared repository, any changes you made to it since opening the local version are lost, and any resources you had checked out are unlocked. The latest versions of the workspace objects in the shared repository are opened.

To revert to the shared version of the workspace, you can use either of the following two methods:

- **File —> Revert to Shared**

  This action unlocks all resources you have checked out, in all your workspaces. Changes you have made are lost. All resources revert back to the state they were in when you checked them out.

  For example, if you have checked out a message flow and have changed a property value for a node within it, and have then selected **Revert to Shared**, the message flow is unlocked and its content reloaded from the Configuration Manager. The change you have made to the node is lost.

This action does not restore deleted resources. Any resource that you have deleted remains deleted.

- **View —> Refresh from Shared**

  This action does not affect the state of the resources you have checked out. The content of all other resources is updated to be consistent with the information held by the Configuration Manager. This allows you to update to and work with the latest changes that other users have made, without losing any changes you are making.

Save your workspace to the shared repository

When you save the workspace to the shared repository, configuration changes you have made are saved. New objects and ones that were checked out are checked in.

You can check in any individual resource at any time by selecting it, right-clicking to bring up the resource menu, and selecting **Check In**.

When you click **Check In**, other resources that are dependent on this resource are also checked in.

You can find out which configuration resources are checked out to you in your current workspace by selecting **File —> Check In —> List** to display the **Check In List** dialog.

You can check in a resource from the **Check In List** dialog by highlighting the resource and clicking **Check In**. When you select one or more items from this list, other resources that are dependent on this resource are displayed in a dialog, and you can choose to continue with the check in action, or cancel it.

You can check in all the objects that are new or checked out in your current workspace. Your current workspace is identified in the title bar of the Control Center. Select **File —> Check In —> All in Workspace** to select this option.

The Control Center locates the new and locked objects that you have in this current workspace, and it checks for other new or locked objects that are referenced directly (in this same workspace) or indirectly (in another workspace) by these objects.

Therefore this action can check in more than you have in your current workspace. This extra action is required to ensure that the data in the message and configuration repositories retains its integrity and consistency.

A dialog box lists all the resources that will be checked in by this action, and you can continue or cancel the action.

You can also check in all the objects that are new or checked out in all your local workspaces. Select **File —> Check In —> All (Save to Shared)**. Information from all your workspaces (your current open workspace and all others on your local system) are reviewed by the Control Center to identify new or locked objects. All items are checked in together.

Note: Whenever items are being checked in, the status bar at the bottom of the window will display the percentage of work done together with text describing the work in progress.

You can find further details on checking resources in and out in the online help for the Control Center.

Import resources

Importing and exporting message sets

You cannot import message sets using this method. You must use the import and export command (**mqsiimpexpmsgset**) to import message sets. See the *MQSeries Integrator Administration Guide* for details.

You can import resources from an XML export file into the local repository. Multiple files can be imported in one import operation, by browsing and selecting the files you require. To import resources:

1. Set up your preferences by selecting:

   File —> **Preferences** —> **Control Center** -> **Import and Export**.

   In this dialog you can choose whether you show or hide import results.
2. If you are importing topology data or Topics remember to:
   - Check out the Topology root in your current workspace
   - Check out the parent topic (which might be the Topic root)
3. Click **File —> Import to workspace**.

   The **Import** dialog is displayed.
4. Select the type of resources you wish to import:
   - Topology
   - Topics
   - Message flows:
     – When a message flow is imported, a reference is added to the existing workspace if it does not already exist.
     – If a Message Flow exists on the Configuration Manager and is not locked , the user can import it, and then update the Configuration Manager with the imported version by selecting **Replace resources in Configuration Manager which are not locked**.
5. Click on the **Browse** button in the **Import Resources** dialog and select the files you wish to import.
6. Click on **Import**. The relevant contents of the file are imported. The file contents do not replace the current workspace contents. While the import is in progress, the status bar at the bottom of the window will display the percentage of work done and text describing the work in progress.

You can import individual message flows provided that they were exported as individual message flows. The import action cannot pick out individual resources, for example a particular message flow, topic or broker definition, from an export file which contains multiple message flows, topics or broker definitions.

When the import action has completed, the results of the import operation are shown in a dialog containing a list of imported resources.

For the import action to succeed:

- For Message Flows with **Replace resources in configuration manager if not locked** selected:

  - If the object you want to import already exists in a shared configuration and it is checked out by you

    then the normal rules for import apply (described in ″Normal import rules″).
  - If the object you want to import already exists in a shared configuration and it **is** checked out to another user

    Nothing about the object will be changed.
  - If the object you want to import already exists in a shared configuration and it is **NOT** checked out to another user:
    - The object will be imported
    - If it is not already in your workspace, it will be added to the workspace
    - It will appear in the appropriate view
    - It will not be marked by the Key icon (as it is not checked out)
- For Message Flows with **Replace resources in configuration manager if not locked** not selected

  The normal rules for import apply (described in ″Normal import rules″).
- **Normal import rules**: For all objects (except for aspects of message flows described earlier) the following rules apply:
  - If the objects you want to import already exist in a shared configuration you **must** check them out
  - If the objects you want to import already exist in a shared configuration, are not checked out by you, and appear in the import file, they are not imported. If the objects
    - Were not in your workspace before import, they are not added to the workspace
    - Were in your workspace before import, they are unchanged
- Resources that do not already exist are imported into your workspace and are marked with the New icon. You must check these in if you want to save them in the shared configuration. The imported workspace might refer to resources that cannot be found in the Configuration Manager you are currently connected to. You can safely remove these resources if you do not want them created in this Configuration Manager.
- You must check out the Topology root if you are importing Topology information.
- You must check out the Parent topic (which might be the Topic root), if you are importing Topic information.

If the current workspace contents were unsaved, you are prompted to save them before the new resources are imported.

See the Control Center online help for more information about the effects of the import operation.

Export resources

Importing and exporting message sets

You cannot export message sets using this method. You must use the import and export command (**mqsiimpexpmsgset**) to export message sets. See the *MQSeries Integrator Administration Guide* for details.

It is possible to select either an item or a group of items to export. To do this:

1. Set up your preferences by selecting:

   File -> **Preferences** —> **Control Center** —> **Import and Export**.

   In this dialog you can choose from the following settings for import and export operations:
   - Whether you want to show or hide export results
   - Specify that files will be exported to a predefined directory with generated file names

     Note: By default, export files go to

     ```
     <<mqsi_root\Tool\export
     ```
   - Choose not to be warned if an export file already exists
   - When exporting a multiple items, decide whether they are to be exported to single or multiple files
2. **To export a message flow**: From the Message Flow pane, select the items you want to export. If you have selected multiple items grouped together, also known as collection, you can choose either to export all collection members from the Configuration Manager, or just those in the workspace. While export is in progress, the status bar at the bottom of the window will display the percentage of work done and text describing the work in progress.
3. **To export everything in a workspace**: From the Workspace select **File —>Export all in workspace**. While export is in progress, the status bar at the bottom of the window will display the percentage of work done and text describing the work in progress.

Note that information being exported might contain sensitive information pertaining to the users and groups who are defined on the User Name Server. If you are a member of MQSeries Integrator group **mqbrtpic** or **mqbrops**, the topic hierarchy and associated ACL are also exported. If you want to avoid this, you should sign on as a user who is not a member of either group before you run the export.

Chapter 4. Defining messages

This chapter describes how to define a message or a message set. For more information about messages, for example :

- The message tree
- Message domains
- Parsers
- and other details related to messages

Please see the *MQSeries Integrator ESQL Reference* book.

The Message Sets view

The **Message Sets** view is the Control Center interface to the MRM. To display the **Message Sets** view, click the **Message Sets** tab in the Control Center. Figure 3 shows an example of the **Message Sets** view.

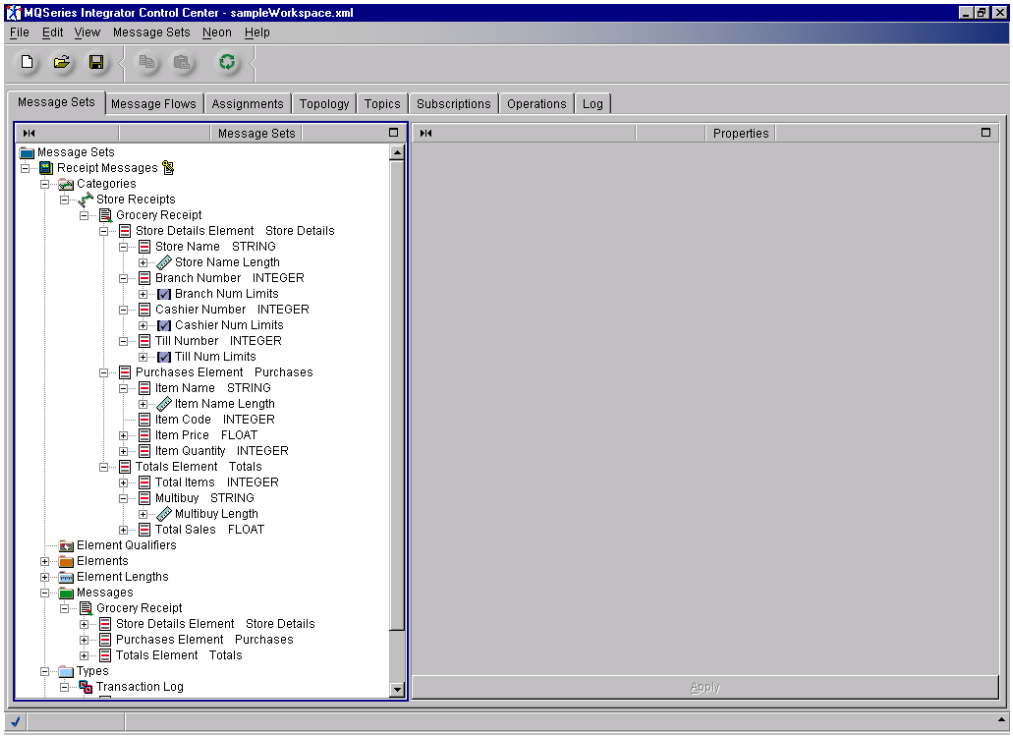


Figure 3. The Message Sets view. The left-hand pane, the Message Sets pane, shows a tree view of the message sets in your workspace. The right-hand pane, the Properties pane, displays the properties of the currently selected entry in the Message Sets pane.

When you click on the plus sign (+) to the left of a message set folder, the contents of the folder are displayed. Each message set folder contains an entry for each of the seven message components. When you create new components within the message set, they appear under the relevant component entry. For example, if you create an element, it appears under the Elements folder within the message set. New components have the **New** blue box icon against them.

Creating message sets

To create a new message set:

1. Either in the Message Sets pane, right click the
   - **Message Sets** root and click **Create —> Message Set**

   or alternatively in the menu bar, you can click the
   - **Message Sets** menu and click **Create —> Message Set**
2. Complete the fields on the initial panel:
   - In the **Name** field, type a name for this new message set. This must follow the naming rules described in “Naming Control Center resources” on page 11. Enter the name ’Consolidated Message Set’ or another name of your choice.
   - Specify a level number if appropriate. For information about setting the level of a message set, see the *MQSeries Integrator ESQL Reference* book.
   - This is a new message set, so the Finalized and Freeze Time Stamp fields can be ignored.
   - If the message set is to be based on another, finalized message set, select that message set from the Base Message Set drop-down list. This list shows all the message sets in the configuration manager, including those for the standard MQSeries headers, which are provided by MQSeries Integrator.
   - Click **Finish** to proceed.
3. Click the **Run Time** tab.

   Select the message parser for messages belonging to this set from the drop-down list. You can choose from:
   - MRM
   - XML
   - NEONMSG
   - JMSMap
   - JMSStream

   The default is MRM.

   For more information about this property see the *MQSeries Integrator ESQL Reference* book.
4. Click the **C Language** tab and complete the Main Header File Name and Orphan Header File Name fields. The defaults for these are CSTRUCTS.H and ORPHANS.H. You can overtype these defaults if you choose. These properties are mandatory, and are used when you generate C language header files from this message set.
5. Click the **COBOL Language** tab and specify a name for the copy book in the Main Copy Book Name field. The default is MAINBOOK.CPY. You can overtype this default if you choose. These properties are mandatory, and are used when you generate COBOL language header files from this message set.
6. If this message set is to contain legacy messages (for example, if message definitions are to be imported into this message set), you need to specify the CWF values.

   Click the **Custom Wire Format** tab. The default Custom Wire Format Identifier is CWF. You are recommended to use this default value. If you choose the change this value, you must set it to a string of 8 characters or less, and the first three characters must be CWF.

7. If you want to provide a description of this message set, click the **Description** tab. Any description text you provide here is included in documentation generated by the MRM.

   Type a short description, or a long description, or both.

8. Click **Finish** to complete the definition of this message set. The message is now created and checked out.

A locked entry appears under **Message Sets** root in the Message Sets pane. When the new message set entry is highlighted in the Message Sets pane, its properties appear in the Properties pane. Notice that an identifier for the new message set has been generated automatically by the MRM.

When you are ready to share a new message set with other Control Center users, you check it into the shared configuration. You can do this before the message set contains any message definitions, if you want. For more information about checking in message sets, see "Checking in and checking out message sets".

Now that you have defined a message set, you are ready to define the messages that will belong to it, as described in "Creating messages" on page 26.

Checking in and checking out message sets

When you have created and populated a message set, you can assign it to a broker (as described in "Assigning message sets to brokers" on page 88). You do not need to have checked the message set into the shared configuration before assigning it. However, you must check it in before the assignment of message set to broker can be deployed in the broker domain.

To check in a message set, in the Message Sets pane right click the folder of the message set you want to check in, and click **Check In**.

The message set is checked into the shared configuration. It still appears in your workspace, but the **Key** icon against its folder has disappeared.

When you check in a message set, any checked out objects in the message set are not checked in by this action. You must check in these objects individually. Alternatively, you can select one of the more comprehensive check in options (available from the File menu) when you check in the message set:

- **File —> Check In —> All in Workspace** checks in all objects that are contained within your current workspace (this is identified on the title bar of the Control Center).
- **File —> Check In —> All (Save to Shared)** checks in all objects in your local repository (that is, within all of your available workspaces).

When you have checked in a message set, it is available to other users from the shared configuration. If you want to make further changes to the message set, you must first check it out of the shared configuration:

1. In the Message Sets pane, right click the folder of the message set you want to edit.
2. Click **Check Out**.

   The message set is checked out of the shared configuration. Its entry in the Messages Pane has a **Key** icon against it to remind you that the definition is checked out.

Creating messages

This section describes how to create a message, using the message shown in Figure 4 to illustrate the process. The Control Center also provides a SmartGuide for message creation, that handles simpler message formats and makes the task of creating them easier. For details of the SmartGuide, see “Defining messages using the SmartGuide” on page 33.

```
Grocery Receipt      (TransactionLog)
   Store Details
      Store Name     (STRING, Length 20, Fixed Length, Left Justified,
                      Padding character Space)
      Branch No.     (INTEGER, Extended Decimal, Length 8,
                      Unsigned 30000000 - 39999999)
      Cashier No.    (INTEGER, Extended Decimal, Length 3,
                      Unsigned 000 - 500)
      Till No.       (INTEGER, Extended Decimal, Length 8,
                      Unsigned 700 - 799)
   Purchase          (Can have up to 15 purchases on one transaction
                      log)
      Item Name      (STRING, Length 40, Fixed Length, Left Justified,
                      Padding character Space)
      Item Code      (STRING, Length 20, Fixed Length, Left Justified,
                      Padding character Space)
      Item Price     (FLOAT, Packed Decimal, Length=4, Signed, VDP=2)
      Item Quantity  (INTEGER, Packed Decimal, Length=2, Signed)
   Totals
      Total Items    (INTEGER, Packed Decimal, Length=5, Signed)
      Multibuy       (STRING, Length 5, Fixed Length, Left Justified,
                      Padding character Space)
      Total Sales    (FLOAT, Packed Decimal, Length=6, Signed, VDP=2)
```

Note: where VDP means virtual decimal point.

Figure 4. Creating a message - sample message

You will work with the message set ’Consolidated Message Set’. Within this message set, you will create a message named ’Grocery Receipt’, of the type ’Transaction Log’. When you have created the message, you will add it to the category ’Store Receipts’. It is assumed you have already created the message set, and have checked this out of the message repository (that is, the message set is displayed with the **Key** icon against it).

Figure 5 on page 27 shows the **Message Sets** view populated with this message and message set. This is the setup you will create if you complete the following message creation instructions.

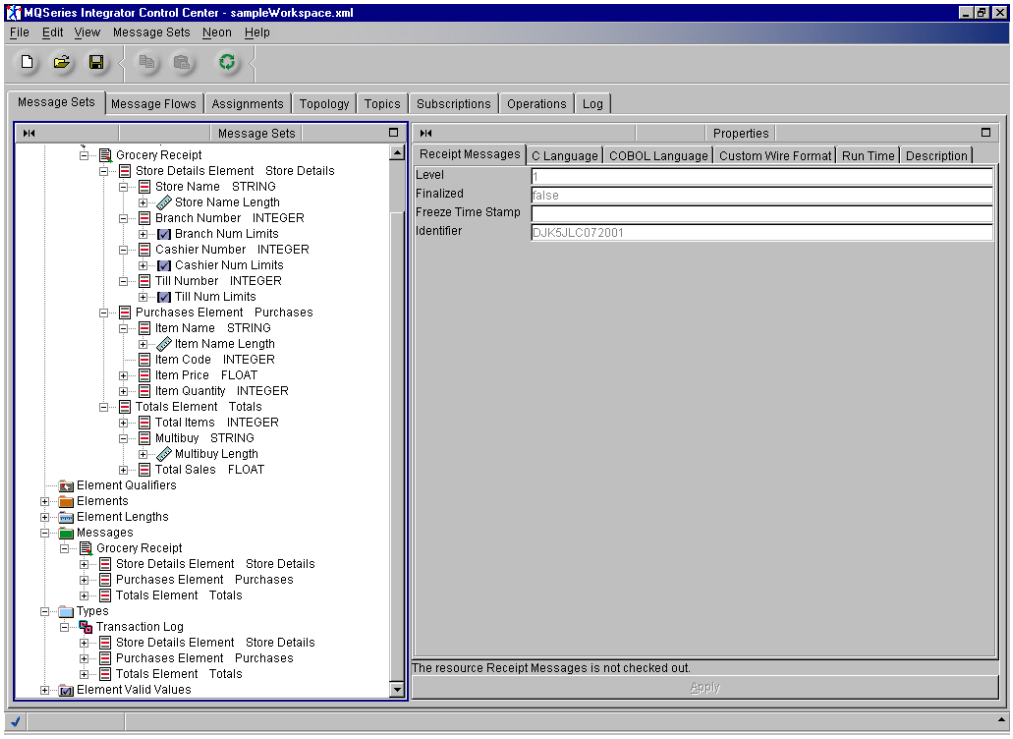


Figure 5. A message defined in the Message Sets view

Defining a message starting from the lowest level elements

These instructions demonstrate how to create the message illustrated in Figure 4 on page 26 from the bottom up (that is, starting with the lowest-level elements and working towards the top of the message hierarchy). All of these tasks are performed in the context of a single message set.

To define this message:

1. Define element length components for all STRING elements.
   a. In the Message Sets pane, right click the entry of the message set Consolidated Message Set and select **Create —> Length**.

      The **Create a new Length** dialog is displayed.
   b. In the **Create a new Length** dialog, type Multibuy Length 5 in the **Name** field; type StrLen5 in the **Identifier** field; and type 5 in the **Maximum Length** field.
   c. Repeat this procedure for the String Length 20 and String Length 40 element length components.
   d. Click **Finish** to complete the definition of this element length component.

      An entry for this new element length component appears in the Element lengths folder in the Consolidated Message Set Receipts.

      Note: If you wanted to provide a description of this component:
      1) Highlight the length
      2) Click on the **Description** tab
      3) Type a short description, or a long description, or both
2. Define element valid value components for the Branch No., Cashier No., and Till No. elements.

a. In the Message Sets pane, right click the entry of the message set and select **Create —> Element Valid Value**.

   The **Create a new Element Valid Value** dialog is displayed.

b. In the **Create a new Element Valid Value** dialog, type Cashier No. Limits in the **Name** field; type Cashier_VV in the **Identifier** field; select type INTEGER from the **Type** drop-down list; type 000 in the **Minimum Value** field; and type 500 in the **Maximum Value** field.

c. If you want to provide a description of this component, click the **Description** tab. Type a short description, or a long description, or both.

d. Click **Finish** to complete the definition of this element valid value component.

   An entry for this new element valid value component appears in the Element valid values folder in the Message Sets pane.

e. If you want to provide a description of this component:

   1) Click on the **Description** tab

   2) Type a short description, or a long description, or both

Repeat this procedure for Branch No. Limits, specifying the identifier Branch_VV, minimum value 30000000, and maximum value 39999999. Repeat this procedure for Till No. Limits, specifying the identifier Till_VV, minimum value 700, and maximum value 799.

3. Create all elements of simple type.

   a. In the Message Sets pane, right click the entry of the message set and select **Create —> Element**.

   b. In the **Create a new Element** dialog, type Store Name in the **Name** field; type StoreName in the **Identifier** field; and select type STRING from the **Type** drop-down list.

   c. Click the **C Language** tab and enter a C name for this element in the C Language Name field. This property is mandatory and is used when you include this element in a C language structure generated from the message repository.

      Note: You can take the defaults for C Language tabs.

   d. Click the **COBOL Language** tab and enter a COBOL name for this element in the COBOL Language Name field. This property is mandatory and is used when you include this element in a COBOL language structure generated from the message repository.

      Note: You can take the defaults for COBOL Language tabs.

   e. Repeat this process for the remaining elements of simple type:

   f. Click **Finish** to complete the definition of this element component.

      An entry for this new element component appears in the Elements folder of the Receipts Consolidated Message Set.

   g. If you want to provide a description of this component:

      1) Click on the **Description** tab

      2) Type a short description, or a long description, or both

Name	Identifier	Type
Branch No.	BranchNo	INTEGER
Cashier No.	CashierNo	INTEGER

Name	Identifier	Type
Till No.	TillNo	INTEGER
Item Name	ItemName	STRING
Item Code	ItemCode	STRING
Item Price	ItemPrice	FLOAT
Item Quantity	ItemQty	INTEGER
Total Items	TotalItems	INTEGER
Multibuy	Multibuy	STRING
Total Sales	TotalSales	FLOAT

4. Add a length reference to elements of type STRING.
   a. In the Message Sets pane, right click the entry for the Store Name element in the Elements folder of the message set. Click **Add —> Length**.

      The **Add an existing Length** dialog is displayed.
   b. From the list of element length components in the **Add an existing Length** dialog, select String Length 20. Click **Finish**.

      An entry for the String Length 20 component appears under the Store Name entry in the Elements folder.

   Repeat this procedure for the elements Item Name (String Length 40), Item Code (String Length 20), and Multibuy (String Length 5).
5. Add a valid value reference to elements Branch No., Cashier No., and Till No.:
   a. In the Message Sets pane, right click the entry for Branch No. in the Elements folder, and click **Add —> Element Valid Value**.

      The **Add an existing Element Valid Value** dialog is displayed.
   b. From the list of element valid value components in the **Add an existing Element Valid Value** dialog, select Branch No. Limits. Click **Finish**.

   Repeat this procedure for Cashier No. (Cashier No. Limits) and Till No. (Till No. Limits).
6. Create the compound types Store Details Type, Purchase Type, and Totals Type.
   a. In the Message Sets pane, right click the entry of the message set and click **Create —> Compound Type**.

      The **Create a new Compound Type** dialog is displayed.
   b. In the **Name** field enter Store Details Type, and in the **Identifier** field enter StoreDetsType.
   c. Click the **C Language** tab and enter a C name for this element in the C Language Name field. This property is mandatory and is used when you include this element in a C language structure generated from the message repository. Type the file name in the File Name field.

      Note: You can take the defaults for C Language tabs.
   d. Click the **COBOL Language** tab and enter a COBOL name for this element in the COBOL Language Name field. This property is mandatory and is used when you include this element in a COBOL language structure generated from the message repository. Type a copy book name in the Structure Copy Book Name field.

Note: You can take the defaults for COBOL Language tabs.

e. Repeat this process for the compound types Purchase Type and Totals Type.

f. Click **Finish** to complete the definition of this compound type component.

The new compound type appears in the Types folder of the Receipts message set.

g. If you want to provide a description of this component:

1) Click on the **Description** tab

2) Type a short description, or a long description, or both

7. Create the elements Store Details, Purchase, and Totals.

a. In the Message Sets pane, right click the entry of the message set and select **Create —> Element**.

The **Create a new Element** dialog is displayed.

b. In the **Create a new Element** dialog, type Store Details in the **Name** field; type StoreDets in the **Identifier** field; and select type Store Details Type from the **Type** drop-down list.

c. Click the **C Language** tab and enter a C name for this element in the C Language Name field. This property is mandatory and is used when you include this element in a C language structure generated from the message repository.

Note: You can take the defaults for C Language tabs.

d. Click the **COBOL Language** tab and enter a COBOL name for this element in the COBOL Language Name field. This property is mandatory and is used when you include this element in a COBOL language structure generated from the message repository.

Note: You can take the defaults for COBOL Language tabs.

e. Repeat this procedure for the elements Purchase and Totals.

f. Click **Finish** to complete the definition of this element component.

An entry for this new element component appears in the Elements folder of the Receipts Consolidated Message Set.

g. If you want to provide a description of this component:

1) Click on the **Description** tab

2) Type a short description, or a long description, or both

8. Add child elements to elements Store Details, Purchase, and Totals.

a. Right click the element Store Details in the Messages Pane. Click **Add —> Element**.

The **Add an existing Element** dialog is displayed, showing all elements in your workspace.

b. Hold down the Ctrl key and select the elements Store Name, Branch No., Cashier No., and Till No. from this list. Click **Finish**.

The selected elements appear under the entry Store Details in the Elements folder.

Repeat this procedure to populate the Purchase and Totals elements.

9. To change the order of the child elements in an element, right click the Type entry in the Messages Pane, and click **Reorder —> Element**. Change the order of the displayed elements, and click **Finish**.

The reordered elements appear in their new order under the entry for the parent element.

10. Add the CWF characteristics to the child elements in each compound type:
    a. Type Purchase must be checked out.
    b. In the Types folder, expand the Purchase entry and click the child Item Name to select it.
    c. Click the **Custom Wire Format** tab in the Properties pane.

       In the Length count field enter 40, and in the Padding Character field type the word Space.

       Click the **Apply** bar at the bottom of the Properties pane.
    d. In the Types folder, expand the Purchase entry and click the child Item Price to select it.
    e. Click the **Custom Wire Format** tab in the Properties pane.

       In the Physical type field, select the type Packed decimal. In the Length Count field, type 4. In the Signed field, type Yes. In the VDP field, type 2.

       Click the **Apply** bar at the bottom of the Properties pane.

    Follow this procedure for the child elements of the compound types Store Details and Totals, and for the remaining child elements in compound type Purchase.

    Note: The CWF characteristics do not belong to an element in isolation. They belong to an element in its context within a type.
11. Create the compound type Transaction Log Type.
    a. In the Message Sets pane, right click the entry of the Receipts Consolidated Message Set and click **Create —> Compound Type**.

       The **Create a new Compound Type** dialog is displayed.
    b. In the **Name** field enter Transaction Log Type, and in the **Identifier** field, enter TransLogType.
    c. Click **Finish**.
    d. Click the **C Language** tab and enter a C name for this type in the C Language Name field. This property is mandatory and is used when you include this type in a C language structure generated from the message repository. Type the file name in the File Name field.

       Note: You can take the defaults for C Language tabs.
    e. Click the **COBOL Language** tab and enter a COBOL name for this type in the COBOL Language Name field. This property is mandatory and is used when you include this type in a COBOL language structure generated from the message repository. Type a copy book name in the Structure Copy Book Name field.
    f. If you want to provide a description of this component, click the **Description** tab. Type a short description, or a long description, or both.

       Note: You can take the defaults for COBOL Language tabs.
    g. Click **Finish** to complete the definition of this compound type component.

    The new compound type appears in the Types folder of the Receipts Consolidated Message Set.
12. Add elements to type Transaction Log.

a. Right click the entry for the Transaction Log type in the Types folder of the message set Consolidated Message Set. Click **Add —> Element**.

   The **Add an existing Element** dialog is displayed, showing all elements in your workspace.

b. Hold down the Ctrl key and select the elements Store Details, Purchases, and Totals from this list. Click **Finish**.

The selected elements appear under the entry Transaction Log in the Types folder of the Consolidated Message Set message set.

13. To change the order of the child elements in a type, right click the Transaction Log type entry in the Types folder of the message set Consolidated Message Set, and click **Reorder —> Element**. Change the order of the displayed elements to that shown in Figure 4 on page 26, and click **Finish**.

    The reordered elements appear in their new order under the entry for the parent element.

14. Add repeat information to child element Purchase in the compound type Transaction Log.

    a. Type Transaction Log must be checked out.

    b. In the Types folder, expand the entry Transaction Log and click the child Purchase to select it.

    c. Click the **Connection** tab in the Properties pane.

       In the Repeat field, type Yes. Click the **Apply** bar at the bottom of the Properties pane.

    d. Click the **Custom Wire Format** tab in the Properties pane.

       In the Repeat Count field, type 15. Click the **Apply** bar at the bottom of the Properties pane.

    e. Check in Transaction Log.

    Note: The repeat information does not belong to an element in isolation. It belongs to an element in its context within a type.

15. Create the message component Grocery Receipt.

    a. In the Message Sets pane, right click the entry of the message set and select **Create —> Message**.

       The **Create a new Message** dialog is displayed.

    b. In the **Name** field, enter Grocery Receipt. In the **Identifier** field, enter GroceryReceipt. From the **Type** field drop-down list, select the value Transaction Log Type. Click **Finish**.

    The new message appears in the Messages folder of the Receipts Consolidated Message Sets.

16. Create a message category.

    a. In the Message Sets pane, right click the entry of the message set and select **Create —> Category** to define the message category.

       The **Create a new Category** dialog is displayed.

    b. In the **Name** field, enter Store Receipts. In the **Identifier** field, enter StoreReceipts. Click **OK**.

    The new category appears in the Categories folder in the Message Sets pane.

17. Add the message Grocery Receipt to the category Store Receipts.

a. Right click the category element in the Message Pane. Click **Add —> Message**.

   The **Add an existing Message** dialog is displayed, showing all messages in your workspace.

b. Select the message Grocery Receipt. Click **Finish**.

The selected message appears under the entry for category Store Receipts in the Receipts Consolidated Message Set.

18. The Multibuy element is optional: it is included only when the customer earns a discount by purchasing a specified multiple of any item. To specify the element:
    a. Check out the Totals compound type
    b. Right-click on the Multibuy element in the Totals compound type of the **Properties** pane
    c. Click the **Connection** tab in the Properties pane. Set the Mandatory field to No.
    d. Click the **Apply** bar at the bottom of the Properties pane to apply the change.

Other types of message could be added to this category within this message set. For example, messages describing receipts from clothing stores or from book stores could be added to the category Store Receipts. The messages themselves could be constructed using many of the message components defined for the message Grocery Receipt.

When you are ready to share a new message set with other Control Center users, you check it into the shared configuration. You must also check in the components you have created within the message set, for example, the messages, elements, and compound types you have created to complete the message set. For more information about checking in message sets, see “Checking in and checking out message sets” on page 25.

Defining messages using the SmartGuide

The MQSeries Integrator Control Center includes a SmartGuide that you can use to create messages from the top down. The SmartGuide also allows you to create compound types, and lets you specify that the compound type created is itself created as a message.

The SmartGuide provides a faster method of defining messages than the process described in “Creating messages” on page 26, not least because it assumes that all the building blocks of the message or compound type are available and do not have to be defined.

The SmartGuide also allows you to reorder elements within the message or compound type you are creating: this makes it easier to view and check the order of elements while you complete the message or compound type structure.

The process for defining a message and defining a compound type are almost identical: this process is described below.

- To create a compound type using the SmartGuide:
  1. Ensure that any lengths you need are defined.

2. In the Message Sets pane of the **Message Sets** view, right click the folder of the message set you want to add definitions to and click **Create with SmartGuide —> Compound Type**.

   The **Create a new Compound Type** dialog is displayed.

3. Complete the dialog:
   a. In the **Name** field, enter the name for this new compound type.
   b. Select the **Create Element** tab to create a new element for this compound type.
      – Enter the new element name
      – Select the Type and Length for the element

        You cannot create a new Type in this dialog.
      – If you have selected the STRING type, choose its length from the dialog box
   c. If you want to add an existing element to the compound type, select the **Add Element** tab, and add one or more elements by selecting from the list of available elements
   d. Click **Add** and this element will be added to a compound type
   e. Repeat this selection process for additional elements that you want to add to this compound type. To remove any existing element, or one you have created, select the element and click **Remove**
   f. You can reorder the elements you have defined in the compound type by highlighting the element, and using the up and down buttons to move the element to where you want it in the structure.
   g. Click **Finish**. The compound type you have created is now included in the Types folder of the message set. It has been given an identifier of `t_` followed by the name you gave the type. For example, if the name of the type you created is `ctype1`, the identifier assigned for you is `t_ctype1`.

      Similarly, any elements you have created are given an identifier of `e_` followed by the name of the element (with spaces removed). For example, `e_element1`.

      Note that any elements you create within the new compound type are not automatically added to the elements folder in the message set, but you can add them if you want to.

   The new compound type is added to the Types folder of the message set.

- To create a message using the SmartGuide, use the same procedure as that described above for a Compound Type. Click **Create with SmartGuide —> Message** from the message set actions list, and complete the **Create a new Message** dialog. (The only difference between the two dialogs is that the Create as message check box is not included in the **Create a new Message** dialog.)

Working with message sets

Message sets can be reordered, and edited. Additionally, most actions carried out against a message set can be undone.

Reordering elements in compound types

The reorder action is only supported for compound types. Right click the compound type you want to reorder, and select **Reorder->Element**. The *Reorder Elements* SmartGuide is displayed listing the elements defined within the compound type. You

can move these elements up or down to change the order as you choose. You cannot add new elements using this option. You cannot reorder external elements (those with the global icon).

Press **Finish** to confirm you new order: the new order is reflected in the tree in the left hand pane.

You can use this action if you want to reorder elements to change the rules that apply when the CWF attributes are set.

Undo action for message sets

The undo action for message sets is consistent with undo across the Control Center, but you are unable to undo the following actions:

- Deleting a message set
- Deleting a compound type
- Deleting an element

Editing message sets and components

You can edit the properties of message sets and components. You can also edit the relationships between components (for example, you can remove an element from a compound element), and you can delete components or remove them from the workspace.

All properties you can edit are displayed in the Properties pane of the **Message Sets** view. For example, if you highlight an element in the Message Sets pane, its properties, including those you can edit, are displayed in the Properties pane. When you change the value of a property, you click the **Apply** bar at the bottom of the Properties pane to make the change take effect.

An individual message component can be removed from the workspace or deleted from the shared configuration. For example, to remove an element from the workspace, right click the element in the Messages Pane and click **Remove**. Note, however, that whether a component is checked out dictates whether you can edit its properties, remove it from the workspace, or delete it, as does the check-out status of any *related* component. Table 1 summarizes the available edit actions and shows for each action:

- Which component needs to be checked out
- What happens when you make the change

Table 1. Editing relationships and properties: check-out requirements

If you want to:	You must check out:	Then:
Edit the basic properties of a component	The component you want to edit	You can edit the component and check it back in
Edit the connection tab of a child element	The compound type that is the parent of the element	You can edit the connection tab then check the parent back in.
Edit the CWF of a child element	The compound type that is the parent of the element	You can edit the CWF tab then check the parent back in.
Edit the C language tab, COBOL language tab, or Description tab of a component	The component you want to edit	You can edit all three tabs. The name is C-validated or COBOL validated by the Control Center; you cannot click **Apply** if they are invalid. If they are valid, you can check the component back in.

Table 1. Editing relationships and properties: check-out requirements (continued)

If you want to:	You must check out:	Then:
Edit an element qualifier assignment	The associated message	You can edit the message then check it back in.
Delete an element length from the Element Lengths folder	Nothing	If nobody has the element length checked out, and if no string element depends on the element length, the element length is deleted from the shared repository. Otherwise, you get an error message and are not allowed to delete the element length.
Remove an element length from the Element Lengths folder.	Check-out status is not significant	The element length is removed from the workspace and there is no change in the shared repository. The element length can be added to the workspace again.
Delete a compound type from the Types folder	Nothing	If any user has an element or a message of this type checked out, or if any user has this type checked out. you get an error message and are not allowed to delete. Otherwise, the type is deleted and all elements of this type are also deleted throughout the message set.
Remove a compound type from the Types folder	Check-out status is not significant	The type and its children are removed from the workspace under the Types folder. Nothing else is affected. The compound type can be added to the workspace again.
Remove a simple type from the Types folder	Nothing	The type is removed from the workspace under the Types folder. Nothing else is affected. The simple type can be added to the workspace again.
Remove a child simple element from a type in the Types folder	The type from which you will remove the element	The child is deleted from the type. When you check the type in, it is updated in the shared repository, but the child element continues to exist. Other types that contain the element as a child are not affected.
Delete a child simple element from a type in the Types folder	Nothing	If nobody has the child simple element checked out and if nobody has any type or element qualifier that is a parent of the simple element checked out, the element is deleted from the shared repository and all the types that previously used it as a child are updated. Otherwise, an error message is issued and you are not allowed to perform the delete.
Delete a child compound element from a type in the Types folder	Nothing	If nobody has the child compound element checked out and if nobody has any type or element qualifier that is a parent of the compound element checked out, the element is deleted from the shared repository and all the types that previously used it as a child are updated. Otherwise, an error message is issued and you are not allowed to perform the delete.
Remove a child compound element from a type in the Types folder	The type from which you will remove the element	The child is deleted from the type and, on check in, the type is updated in the shared repository but the child element continues to exist. Other types that contain the element as a child are unaffected.
Delete a simple element from the Elements folder	Nothing	If nobody has the element checked out, and if nobody has any type or element qualifier that is a parent of the element checked out, the element is deleted from the shared repository and all the types that previously used it as a child are updated. Otherwise, an error message is issued and you are not allowed to perform the delete.
Remove a simple element from the Elements folder	Check-out status is not significant	The element is removed from the workspace under the Elements folder. Nothing else is affected. The element can be added to the workspace again.

Table 1. Editing relationships and properties: check-out requirements (continued)

If you want to:	You must check out:	Then:
Delete a top-level compound element in the Elements folder	Nothing	If nobody has the element checked out, and if nobody has any type or element qualifier that is a parent of the element checked out, the element is deleted from the shared repository, and all the types that used the element as a child are updated. Otherwise, an error message is issued and you are not allowed to perform the delete.
Remove a top-level compound element in the Elements folder	Check-out status is not significant.	The element and its children are removed from the workspace under the Elements folder. Nothing else is affected. The compound element can be added to the workspace again.
In the Elements folder, alter a compound element by deleting a child simple element	Nothing	If nobody has the child element checked out; and if nobody has any type or element qualifier that is a parent of the element checked out; and if nobody has the type of the compound element checked out; the element is deleted from the shared repository, and all the types that previously used the element as a child are updated. Otherwise, an error message is issued and you are not allowed to perform the delete.
In the Elements folder, alter a compound element by removing a child simple element	The type associated with the compound element	The child is deleted from the type, and when you check the type back in, it is updated in the shared repository but the child element continues to exist. Other types that contain the element as a child are not affected.
In the Elements folder, alter a compound element by deleting a child compound element	Nothing	If nobody has the child element checked out; and if nobody has any type or element qualifier, of which the compound element is a child, checked out; and if nobody has the type of the parent compound element checked out; then the element is deleted from the shared repository and all the types that used the element as a child are updated. Otherwise, an error message is issued and you are not allowed to perform the delete.
In the Elements folder, alter a compound element by removing a child compound element	The type associated with the parent compound element	The child compound element is removed from the workspace. Nothing else is affected. The compound element can be added to the workspace again.
Delete a message in the Messages folder	Nothing	If nobody has the message checked out, and if nobody has any category of which the message is a child checked out, the message is deleted from the shared repository, Otherwise, an error message is issued and you are not allowed to perform the delete.
Remove a message from the Messages folder	Check-out status is not significant	The message and its children are removed from the workspace under the Messages folder. Nothing else is affected. The message can be added to the workspace again.
Alter a message by deleting a simple child element in the Messages folder	Nothing	If nobody has the child element checked out, and if nobody has the type of the message checked out, and if nobody has any type, of which the element is a child, checked out, then the element is deleted from the shared repository and all the types that used the element as a child are updated. Otherwise, an error message is issued and you are not allowed to perform the delete.
Alter a message by removing a child simple element in the Messages folder	The type associated with the message that contains the element	The child is deleted from the type, and on check in the type is updated in the shared repository, but the child element continues to exist and other types that contain the element as a child are not affected.

Table 1. Editing relationships and properties: check-out requirements (continued)

If you want to:	You must check out:	Then:
Alter a message by deleting a child compound element in the Messages folder.	Nothing	If nobody has the child element checked out; and if nobody has any type or element qualifier that is a parent of the child compound element checked out; and if nobody has the type of the message checked out; then the element is deleted from the shared repository and all the types that used the element as a child are updated. Otherwise, an error message is issued and you are not allowed to perform the delete.
Alter a message by removing a child compound element in the Messages folder	The type associated with the message that contains the element	The child compound element is removed from the workspace. Nothing else is affected. The child compound element can be added to the workspace again.

Changing the state of a message set

When a message set state is created, its state is normal. During their development, message sets can be frozen, unfrozen, and finalized.

To change the state of a message set:

1. In the Message Sets pane, right click the message set whose state you want to change.
2. Click the state you want. For example, to freeze a message set, click **Freeze**.

Note the following:

- When you freeze a message set, the freeze timestamp is added to the properties of the message set.
- If you unfreeze a message set, the freeze timestamp in the properties of the message set is reset to blank.

  Note: To unfreeze a message set, it must be checked out to you.
- When you finalize a message set, the Finalized field in the properties of the message set is set to True and the freeze timestamp is set. Finalize cannot be reversed.
- You cannot freeze or finalize a message set if any of the elements it contains is checked out.

Adding message sets and message components to the workspace

You can add any message set or component that is defined in the message repository to your workspace. The message repository includes all objects defined by current users of the Control Center, and all objects imported into the message repository through the use of the **mqsiimpexpmsgset** command. See *MQSeries Integrator Administration Guide* for more details. To add an existing message set to the workspace:

1. In the Message Sets pane, right click the **Message Sets** root.
2. Click **Add to Workspace—> Message Set**.

   The **Add an existing Message Set** dialog is displayed, showing all message sets that you can add to your workspace (and that aren't already in the workspace).
3. Select message sets from this list as follows:
   - To select a single message set, click the message set name.

- To select multiple message sets that appear sequentially in the list, click on the first message set you want, press and hold the Shift key, then click on the last message set you want. This action selects the two message sets you highlighted, plus any that appear between these two in the list.
- To select multiple message sets that do not appear in a sequence in the list, hold down Ctrl and click each message set you want.

4. When you have selected the message sets you want, click **Finish**.

You now see the selected message sets in the Message Sets view. All of the components of the message set (messages, elements, and so on) are now available to your workspace, but are not automatically added. This is because message sets can be very complex, and it is likely that you do not need to view or access many of the subcomponents. If you add large numbers of components to the workspace, this can cause slow response times and out-of-memory problems.

You can add just those components that you want to work with, or view, by selecting the appropriate folder and adding the components when and as you need them.

For example, to add an element to your workspace:

1. Right click the Element folder of the message set to which you want to make the element available, and click **Add to Workspace —> Element**.

   The **Add an existing Element** dialog is displayed, showing all elements that you can add, that is, all the elements that are defined in the message set.
2. Select one or more elements from the list:
   - To select a single element, click the element name.
   - To select multiple elements that appear sequentially in the list, click on the first element you want, press and hold the Shift key, then click on the last element you want. This action selects the two message sets you highlighted, plus any that appear between these two in the list.
   - To select multiple elements that do not appear in a sequence in the list, hold down Ctrl and click each element set you want.

   The selected elements are added to the Elements folder of the appropriate message set in the Message Sets pane.

You can add categories, element qualifiers, element lengths, messages, types, and element valid values to your workspace in the same way.

Importing message definitions

Legacy definitions can be imported into the message repository. For example, to save you having to create message definitions in the Control Center for your existing C and COBOL messages, you can import these messages into the Control Center and use MQSeries Integrator to create equivalent message repository definitions for you.

To import a message definition:

1. In the Message Sets pane of the **Message Sets** view, right click the message set into which you want to import the definition and click **Import to Message Set —> C** or **Import to Message Set —> COBOL**. The action is also available if you select **Message Sets** from the task bar and click **Import to Message Set —> C** or **Import to Message Set —> COBOL**.

Importing message definitions

The **C Language Importer** dialog or **COBOL Language Importer** dialog is displayed.

2. Type the fully qualified name of the source file you are importing in the Import Source File field, or use the **Browse** button to search for and select the file you want to import. If you want only to generate a report at this time, select the Report only check box. Click **OK**.

This process imports the specified structures and creates definitions as a new message set in the message repository. To complete the process, you must create a message component for each of the compound types that define a complete message, as described in step 15 on page 32. You do not need to create any other message component.

Note: You cannot import message sets created by another Control Center user in another configuration manager. To do this use the message set import and export command (**mqsiimpexpmsgset**), which is described in *MQSeries Integrator Administration Guide*.

Generating MRM message set definitions as XML DTDs

If you have defined messages with an XML message format in the message repository, you can request a Document Type Definition (DTD) to be generated by the MRM.

To generate a DTD:

1. In the Message Sets pane of the **Message Sets** view, right click the folder of the message set for which you want to generate the DTD. Click **Generate —> DTD**.

   The **Generate DTD** dialog is displayed.

2. In the **Generate DTD** dialog, enter the name of the DTD file in the DTD Filename field. Click **Start**.

The DTD for this message set is generated as requested and written to the specified location.

Generating language bindings

You can generate C or COBOL language bindings from message definitions you have created using the Control Center:

To generate C language bindings:

1. In the Message Sets pane of the **Message Sets** view, right click the folder of the message set for which you want to generate language bindings. Click **Generate —> Language Bindings —> C**.

   The **C Language Extractor** dialog is displayed.

2. In the **C Language Extractor** dialog, enter the fully qualified name of the directory of the generated file in the Generated File Location field. If you want to freeze the message set at this time, select the Freeze Message Set check box. (If the message set is already finalized, you cannot select this check box.)

   The categories defined in this message set are listed in the Categories field. You can select a subset of these for inclusion in the language bindings. Alternatively, to include them all, select the Select All check box.

   You must select at least one category for successful generation of language bindings. If no category is listed in this dialog, you must create one.

3. Click **Start**.

   The requested language bindings are generated and written to the specified location.

The process for generating COBOL bindings is identical.

Note: The **Run NLS** check box is not operational at this time.

Generating documentation

You can generate the following documentation in HTML format from the message repository:

- A message book, which contains an entry for each message in a message set or specified category, showing its hierarchical structure.
- A glossary, which contains descriptions of all elements in a message set or specified category, ordered alphabetically by name.

To generate a message book:

1. In the Message Sets pane of the **Message Sets** view, right click the folder of the message set or message category for which you want to generate documentation.
2. Click **Generate —> Documentation —> Message Book**.

   The **Message Definition Book** dialog is displayed.
3. In the **Message Definition Book** dialog, type the fully qualified name of the generated documentation file in the Generated File Location field. This file must be created on the system on which the Configuration Manager is running, not on the local system.

   If you want to freeze the message set at this time, select the Freeze Message Set check box. (If the message set is already finalized, you cannot select this check box.)

   You can generate documentation based on categories or messages. The categories or messages (depending on which you select) defined in this message set are listed in the Categories or Messages field. You can select a subset of these for inclusion in the language bindings. Alternatively, to include them all, select the Select All check box.

   You must select at least one category or one message for successful generation of documentation. If there is no category listed, you must create one.
4. Click **Start**.

The Message Book is generated and written to the specified location: the file MRM-MAIN.HTML is created, along with a subdirectory named `Private`. This subdirectory contains a large number of files, for example, image files and indexes.

The description of each element within each compound type included in the Message Book indicates whether the element is mandatory or optional. There are four possible settings:

1. The element can be `Optional`.

   This value is set if the element is optional within its type.
2. The element can be `Mandatory if parent present`.

   This value is set if the element is mandatory within its type.
3. The element can be `Always Mandatory`.

   This is set if the element is associated with a context tag and that context tag is mandatory within the message for which the documentation has been created. This setting overrides the values `Optional` and `Mandatory if parent present`.

   If the element is associated with a context tag and that context tag is optional within the message, the current setting of `Optional` or `Mandatory if parent present` is not overridden.
4. The element can be `Implied Mandatory`.

This value is set if one of its descendants has been given the value `Always Mandatory`. This value will override the value set for the element in isolation unless the element itself has the `Always Mandatory` value.

Repeating elements are indicated by the characters *** after the element name.

To generate a glossary:

1. In the Message Sets pane of the **Message Sets** view, right click the folder of the message set or message category for which you want to generate documentation.
2. Click **Generate —> Documentation —> Glossary** from the action list of the message set.

   The **Glossary** dialog is displayed.

   This dialog is identical to the **Message Definition Book** dialog, except that only categories are available.
3. Complete the dialog and click **Start**.

The Glossary is generated and written to the specified location: the file MAINGLOS.HTML is created, along with a subdirectory named `Private`. This subdirectory contains a large number of files, for example, HTML files.

Generating documentation

Chapter 5. Working with message flows

This chapter describes the tasks you need to perform to create message flows.

The Message Flows view

To display the **Message Flows** view, click the **Message Flows** tab in the Control Center. Figure 6 shows an example of the **Message Flows** view.

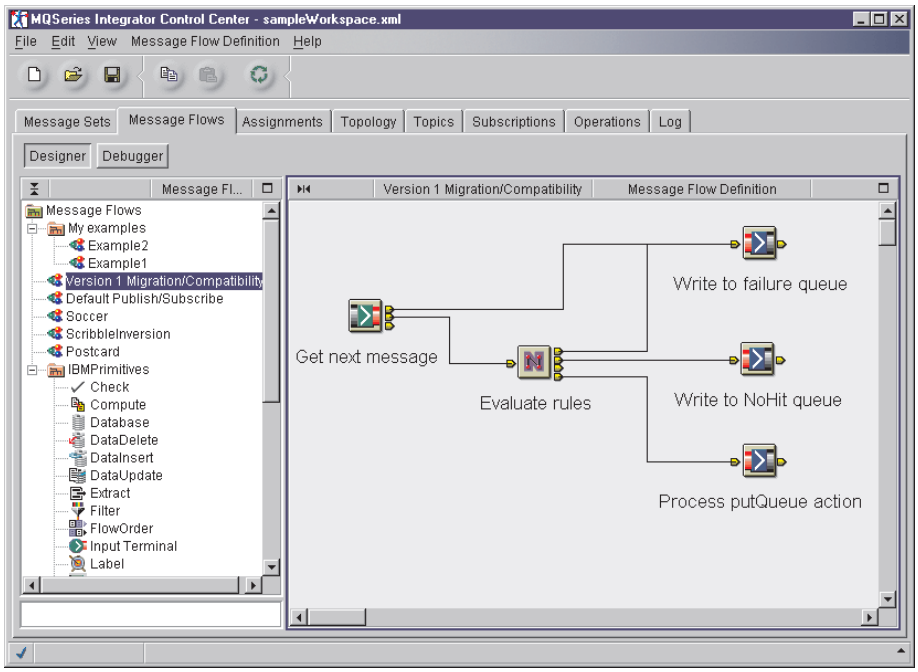


Figure 6. The Message Flows view. The left-hand pane, the Message Flow Types pane, shows a tree view of the message flows in your workspace. The right-hand pane, the Message Flow Definition pane, contains an arrangement of graphical symbols that represent the message flow nodes in a selected message flow.

Double-click on any item in the Message Flow Definition pane to see its properties.

Controlling the appearance of the Message Flow Definition pane

When you add an instance of a message flow node into a message flow by dragging it into the Message Flow Definition pane, the graphical symbol representing the node is created. You can control the appearance and arrangement of these symbols by right-clicking in the Message Flow Definition pane and selecting from the following actions:

Layout graph
Arranges the nodes in the Message Flow Definition pane from left to right, right to left, top to bottom, or bottom to top.

Zoom
Alters the size of all node symbols in the Message Flow Definition pane.

Your zoom settings are saved on exit, and are used when restarting applications.

Manhattan style
Shows connections between nodes as lines at right angles (as shown in Figure 6 on page 45).

Your style settings are saved on exit, and are used when restarting applications.

Snap to grid
Aligns the symbols in the Message Flow Definition pane on an invisible grid.

The settings for aligning your symbols are saved on exit, and used when restarting applications.

Node orientation

You can change the orientation of message flow nodes in the Message Flows pane. The default orientation for nodes is left-to-right; however, you can change the orientation of your nodes to any of the following:

- Right-to-left
- Top-to-bottom
- Bottom-to-top

Additionally, by right clicking on the node, you can:

- Flip the node
- Rotate the node either clockwise or counter clockwise

These settings are saved when you save your workspace.

Creating bend points

When connecting nodes you might want to introduce bend points. Bend points, which are displayed as dots, will only appear if a connection is selected. To create bend points for a new connection:

1. Start the connection by either
   - Left clicking on the node terminal and dragging the mouse to the required position on the canvas

     or
   - By using the pop-up menu (right button) and adding the bend points by clicking on the canvas at the positions you want the bend to occur
2. Complete the connection by selecting the target node

   To cancel the connection before it is completed, either hit the **ESC** key, or click the mouse on a component outside of the canvas.

To bend an existing connection:

1. Click on an existing connection
2. Drag the connection to its new position
3. Release the mouse button, and the bend point will appear

Creating a message flow

1. In the Message Flow Types pane of the **Message Flows** view, right click the Message Flows root, and click **Create —> Message Flow**.

   The **Create a new Message Flow** dialog is displayed.
2. In the **Name** field, type the unique name of your new message flow. This must follow the naming rules described in “Naming Control Center resources” on page 11. Click **Finish**.

Confirmation that the message flow has been created appears in two places in the **Message Flows** view:

- The name of the new message flow appears in the title bar of the Message Flow Definition pane.
- An entry for the new message flow appears in the Message Flow Types pane with a **New** icon against it.

You are now ready to assemble the message flow from the available message flow nodes.

3. From the Message Flow Types pane, drag each of the message flow nodes you want to use into the Message Flow Definition pane. (This step fails if you have not defined a message flow into which you can drag the message flow nodes.)

   A graphical symbol representing each of the nodes you select is shown in the Message Flow Definition pane. The first node of each type that you select has the number "1" appended to its name. For example, if you construct a simple message flow using the MQInput, DataUpdate, and MQOutput message nodes, each appears in the Message Flow Definition pane with a ″1″ appended to its name (that is, MQInput1, DataUpdate1, and MQOutput1).

   If you use more than one instance of any of these nodes within a single message flow, and do not rename each node immediately, the number appended is incremented each time (the second instance has "2", the third has "3", and so on).

4. If you want these nodes to have different names from those assigned, you can rename them by following this procedure:

   a. Check out the message flow

   b. In the Message Flow Definition pane, right click on one of the message flow node symbols, and click **Rename**.

      The **Rename Message Flow Node** dialog is displayed.

   c. In the New Name field, type the new name of this instance of the message node. Click **Finish**.

   The new name of the message flow node appears beneath its symbol in the Message Flow Definition pane. Repeat this process for other message nodes you want to rename.

   Now you are ready to connect the message nodes in your message flow in a way that will provide the processing logic you require. For the remainder of this section, let's assume that you are connecting the MQInput message flow node to the DataUpdate message flow node.

5. To connect the *out* terminal of MQInput to the *in* terminal of DataUpdate, right click the MQInput symbol in the Message Flow Definition pane, and click **Connect —> Out**. (All terminals available to this node appear in this list.) The cursor becomes a cross-hair attached by a red line to the *out* terminal.

6. Move the cross-hair to the *in* terminal shown on the symbol of the DataUpdate node, and click. A line now connects the *out* terminal of the MQInput node to the *in* terminal of the DataUpdate node.

   Note: An alternative way of connecting terminals is to move the cursor slowly over the terminal icons of the node until the label of the terminal you want to connect is displayed, then press the left mouse button. This action converts the cursor to a cross-hair attached by a red line to the

node, which you can move to the appropriate terminal of the next node. Release the mouse button to connect the line. This method requires a certain dexterity.

Follow this process for all terminals within the message flow between which you want to establish connections.

To view or change the description of a connection, select the Properties menu, **Properties**, and view or modify the description as required.

7. You must configure the nodes in your message flow to match your processing requirements.

   Note: When you have assembled the message nodes you want to use in the Message Flow Definition pane, the order in which you rename, connect, and configure them is unimportant.

8. If you are ready to make this message flow generally available within the broker domain, check it into the shared configuration as described in "Checking in message flows" on page 54.

You must be aware of the following information regarding message flows that access external databases:

1. Fully globally coordinated message flows that involve a DB2® resource manager are supported on DB2 Universal Database® V6.1 and V7.1.
2. The message flow thread connects to the specified data source, unless it is already connected. When a thread has acquired a connection to an ODBC data source, the connection is not relinquished.

   You are recommended to determine the number of database connections required by a broker for capacity and resource planning purposes. The default action taken by DB2 is to limit the number of concurrent connections to a database to the value of the *maxappls* configuration parameter. The default for *maxappls* is 40. Check the appropriate documentation for connections to databases from other suppliers.

   The connection requirements for a single message flow are:

   - One required per message flow thread that contains a publication node.
   - One required per database access node to separate ODBC data source names per message flow thread (that is, if the same DSN is used by a different node, the same connection is used).

Note: These database connections are in addition to the run-time connections required by the broker (to the DB2 or SQL Server database that is defined to hold its internal information). For details of these connections, see the MQSeries web site.

Creating a message flow category

When you have a large number of message flows in your workspace, the Message Flows tree in the Message Flow Types pane can become difficult to navigate. To introduce some structure into the list, you can define message flow categories, under which you can organize related message flows. (The IBM Primitives, for example, belong to the IBM Primitives message flow category.)

To create a message flow category:

1. In the Message Flow Types pane, right click the root of the Message Flows tree, and click **Create —> Message Flow Category**.

The **Create a new Message Flow Category** dialog is displayed.

2. In the **Name** field, type the name of your message flow category. Click **Finish**.

An entry for the new message flow category appears in the Message Flow Types pane.

You can create new message flows within this new message flow category, as follows:

1. Right click on the message flow category folder in the Message Flow Types pane, and click **Create —> Message Flow**.
2. Follow the instructions for creating a message flow from step 2 on page 46.

You can also add existing message flows, to a message flow category, as described in “Adding a message flow to your workspace”.

Adding a message flow to your workspace

If you want to incorporate message flows:

- Created by other Control Center users
- Created in another Control Center session
- Removed from the Control Center in the current Control Center session

into your own workspace, you need to begin by adding them into your workspace. When you add definitions to your workspace, a reference to each definition is created in your workspace.

To add a message flow to your workspace:

1. Right click the Message Flows root in the Message Flow Types pane, and click **Add to Workspace —> Message Flow**.

   The **Add an existing Message Flow** dialog is displayed.

   - To select a single entry from this list, click the message flow name.
   - To select multiple entries that appear sequentially in the list, click on the first message flow you want, press and hold the Shift key, then click on the last one you want. This action selects the two message flows you highlighted, plus any that appear between the two in the list.
   - To select multiple message flows that do not appear in a sequence in the list, hold down Ctrl and click each entry you want.
2. When you have selected the message flows you want, click **Finish**.

   The items you selected are added to the Message Flow Types pane, from where you can include them in new message flows.

If you perform this task by right clicking on a message flow category in the Message Flow Types pane and clicking **Add —> Message Flow**, the items you select are added to the folder of the message flow category in the Message Flow Types pane.

Checking a message flow

When you have created a message flow, you can use the message flow SmartGuide to check the following:

- All ESQL syntax is correct.
- All references to message fields are resolved.

Checking a message flow

- All message flow properties (including promoted properties) are valid.
- All message field names are recognized.

This SmartGuide allows you to check for errors before you deploy the message flow, thus saving time and inconvenience.

To use the message flow check SmartGuide:

1. Right click the message flow you want to check, and select **Check message flow...**.
2. Select the message set and messages that are used in the message flow from the list presented in the dialog.
3. Click **Next**. The SmartGuide performs the checks identified above and presents a results window.

   You can use the information displayed in this window to identify the node that is incorrect and the nature of the error. The upper part of the screen lists the errors found: the lower part of the screen provides the details of the error selected in the upper part. You can move through the list of errors using the arrow keys on the right hand side of the window.

   You can leave this window active as a reference while you make the changes and corrections.

Including one message flow in another

You can create a message flow that includes a mixture of message flow nodes and existing message flows. You might want to do this, for example, if you have created a standard message flow to process errors or to perform a particular calculation. You can define this standard message flow once and include it in other message flows wherever it is required, which is easier than redefining the same sequence of nodes in each message flow that uses them.

A set of nodes created as a partial message flow for use in this way is also known as a subflow or embedded flow.

Note: Any message flow that you intend to reuse in this way does not normally use the standard MQInput and MQOutput nodes to start and end the flow. Instead, it uses the Input Terminal and Output Terminal nodes that are included in the IBM Primitives message category.

1. To include an existing, reusable message flow in a new message flow, you must begin by adding that message flow to your workspace, if it isn't already there, as described in "Adding a message flow to your workspace" on page 49.
2. Create the new message flow, following steps 1 and 2 on page 46.
3. In the Message Flow Types pane, with the new message highlighted in the Message Flow Types pane, drag the message flows and nodes that will make up your new flow into the Message Flow Definition pane.

   Embedded message flows have terminal icons that represent the Input Terminal and Output Terminal nodes they contain. For example, if the nested message flow has one Input Terminal node and two Output Terminal nodes, the message flow icon will have one input terminal and two output terminals, which you connect to other nodes in the higher-level flow in the usual way. You can rename these terminals if you want: for example, one of the Output Terminals might be for an error path, and you might rename this 'failure'.

You can work with the subflow by adding it to your workspace and either selecting it in the Message Flow Types pane (as you can with any message flow), or by double-clicking on the node icon for the embedded flow displayed within the main flow in the Message Flow Definition pane.

The right-hand pane changes to display the configuration for that subflow. You can also reach this view by right-clicking the subflow node, and selecting *Open sub-flow*.

You can return to the parent flow by right-clicking the Message Flow Definition pane and selecting the menu item *Return to parent flow*.

Promoting message flow node properties

A message flow contains one or more message flow nodes, each of which is an instance of a message flow type (either an IBM Primitive, or one you have defined). You can promote the properties of these message flow nodes to apply to the message flow to which they belong. If you do this, any user of the message flow can set values for the properties of the nodes in the message flow, by setting them at the message flow level, without being aware of the message flow's internal structure.

For example, you might want to set the name of a data source as a property of the message flow, rather than a property of each individual node in the message flow that references that data source.

You are creating a message flow that accesses a database called SALESDATA. However, while you are testing the message flow, you want to use a test database called TESTDATA. If you set the data source properties of each individual node within the message flow to reference TESTDATA, you will have to update all these references when you put your message flow into production.

If you promote the data source property, you can set the properties for all of the individual nodes to be SALESDATA, and set the value of TESTDATA for the promoted property to override the node data source values while you test the message flow (the promoted property always takes precedence over the settings for the properties within any relevant nodes).

To promote message flow node properties to a message flow:

1. You must check out the message flow for which you want to promote properties. If it is not checked out, right click the entry for the message flow in the Message Flow Types pane, and click **Check Out**.

   The message flow contents are now displayed in the Message Flow Definition pane.
2. Right click the symbol of the message flow node whose properties you want to promote, and click **Promote Property**.

   The **Promote Property** dialog is displayed.
3. In the **Promote Property** dialog, the names of the properties of the message flow node are displayed in the left-hand pane. This pane is always fully expanded to show all properties that are available for promotion. If you have already promoted properties from this node, they do not appear in the left-hand pane, but in the right-hand pane.

   The names of the properties of the message flow itself, of which the message flow node is a part, are displayed in the right-hand pane. These are properties

that have already been promoted up to the message flow. The original name of the property and the name of the message flow node from which it came, are shown beneath the property entry. This allows you to determine the specific node that is the origin of each promoted property, regardless of the name of the promoted property. See “Renaming promoted properties” on page 53 for information about renaming properties.

4. To promote a property from the message flow node to the message flow, drag its entry from the left-hand pane of the **Promote Property** dialog to the right-hand pane and drop it in an empty part of the pane. It then appears at the top of the pane.

   Figure 7 shows an example of the **Promote Property** dialog.

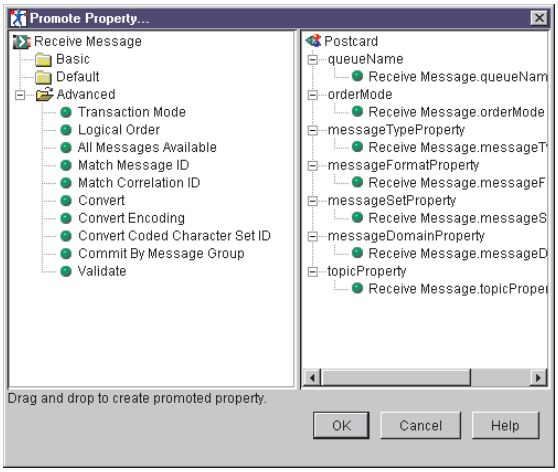


Figure 7. The Promote Property dialog. Some of the properties of the message flow node have been dragged across to the message flow and thus promoted.

5. When you have selected the properties you want to promote to the message flow, click **OK**.

The message flow node properties have been promoted to the message flow. To confirm this, in the Message Flow Types pane, right click the entry for this message flow and click **Properties**.

The **Properties** dialog of the message flow is displayed, showing the message flow node properties you promoted. If you now set a value for one of these properties, that value appears as the default value for the property whenever the message flow is itself included in other message flows.

When you select an embedded message flow within another message flow and view its properties, you see the promoted property values. If you look inside the embedded flow (that is, if you select **Open sub-flow**), you see the original values for the properties. The value of a promoted property does not replace the original property, but it takes precedence at deploy time.

Promoting properties through a hierarchy of message flows

The process of promoting message flow node properties can be repeated as you construct a hierarchy of message flows. You can promote properties from any level in the hierarchy to the next level above, and so on through the hierarchy. The value of a property is propagated from the highest point in the hierarchy at which it is set

down to the original message flow node when the message flow is deployed to a broker. The value of that property on the original message flow node is overridden.

Converging multiple properties

It is possible for a promoted property to provide a value for several message flow node properties at once. For example, if a message flow contains two Database nodes that each refer to the same physical database, you can define the physical database only once on the message flow. To do this, you promote several message flow properties to a single promoted property. Drag the property entry from the left-hand pane to the right-hand pane, and drop it onto an existing promoted property (instead of into the empty pane). You can now see the new property added under the existing promoted property.

Note: If the type of the property you are promoting does not match the type of the existing promoted property, a new promoted property is created at the top of the pane when you drop the property onto the existing property.

Renaming promoted properties

To rename a promoted property:

1. In the **Promote Property** dialog, right click the promoted property, and click **Rename**.
2. In the **Rename** dialog, type the new name for the property. Click **OK**.

The new name of the property appears in the right-hand pane of the **Promote Property** dialog.

Deleting a promoted property from a message flow

To delete a promoted property from a message flow, in the **Promote Property** dialog, right click the promoted property, and click **Delete**.

Note: Any higher level message flow that has used this message flow, and that has set a value for the deleted property, is not automatically updated to reflect the deletion. However, when you deploy that message flow in the broker domain, the deleted property is ignored.

Promoting mandatory properties

If you promote a property that is mandatory (that is, the name appears in bold type in the properties dialog of the message flow node), the mandatory characteristic of the property is not preserved. You are recommended always to provide a default value for the property using the properties dialog of the message flow node from which the property originated.

Example of how to promote message flow node properties

This example demonstrates how to promote message flow node properties.

1. Create a new message flow called Base.
2. Drag an MQInput node and an MQOutput node from the Message Flow Types pane into the Message Flow Definition pane.
3. In the Message Flow Definition pane, right click the symbol of the MQInput node, and click **Promote Property**.

   The **Promote Property** dialog is displayed.
4. Drag the properties you want to promote from the left-hand pane into the right-hand pane. Click **OK**.

How to promote message flow node properties

5. Repeat steps 3 on page 53 through 4 on page 53 for the MQOutput node.
6. Create a new message flow called Middle.
7. Click on the entry for the message flow Middle in the Message Flow Types pane, then drag the message flow Base into the Message Flow Definition pane.

   A graphical symbol of the message flow labeled Base1 appears in the Message Flow Definition pane.
8. In the Message Flow Definition pane, right click the symbol of the Base1 message flow, and click **Properties**.

   The properties you promoted from the MQInput and MQOutput nodes appear as properties of the message flow Base1.
9. Click **Cancel**.
10. In the Message Flow Definition pane, right click the symbol of the Base1 message flow again, and click **Promote Property**.

    The properties that appear in the left-hand pane of the **Promote Property** dialog are those you promoted from the message flow nodes in the Base1 message flow. You can promote these properties to the message flow Middle, displayed in the right-hand pane. If you do this, note that Base1 is listed as the originating message flow.
11. Repeat this procedure to add further levels to the hierarchy of message flows and to promote properties throughout the hierarchy.

Checking in message flows

When you have created a message flow, you can assign it to an execution group as described “Assigning message flows to execution groups” on page 86. You do not need to have checked the message flow into the shared configuration before assigning it. However, you must check it in before you can deploy it to one or more brokers in the message domain.

To check in a message flow, right click the folder of the message flow you want to check in within the Message Flow Types pane, or right click on the background of the Message Flow Definition pane, and click **Check In**.

The message flow is checked into the shared configuration. It still appears in your workspace (as evidenced by the inclusion of its folder in the Message Flow Types pane), but the **New** icon or the **Key** icon against its folder has disappeared.

Checking out message flows

Once you have checked in a message flow, it is available to other users from the shared configuration. If you want to make further changes to the message flow, you must first check it out of the shared configuration:

1. In the Message Flow Types pane, right click the folder of the message flow you want to edit.
2. Click **Check Out**.

   The message flow is checked out of the shared configuration. Its entry in the Message Flow Types pane has a **Key** icon against it to remind you that the definition is checked out.

Creating your own message nodes

You can create your own nodes, called plug-in nodes, to extend the function of MQSeries Integrator. This is described in the *MQSeries Integrator Programming Guide*.

These nodes have to be represented in the Control Center so that users can include them in message flows. The Message Flows (designer view) contains a SmartGuide to help you define the interface for plug-in nodes. See the online help for how to use this SmartGuide and the *MQSeries Integrator Programming Guide* for details on how to build the runtime library, and where to store the library to enable the node's use. This book also provides information about how to write customizers and property editors that allow you to extend the function of your node beyond the defaults supplied by IBM.

Configuring Message flow nodes

The message flow nodes supplied with MQSeries Integrator Version 2.0.2 are also known as the IBM Primitives. For reference information about the message flow nodes, for example, their properties, see the online help.

With the exception of the Compute node, NEONMap and NEONTransform the input message received by a node, and the output message sent on by the node, are identical.

A number of these nodes allow manipulation of the message using ESQL. For more information about ESQL please refer to the *MQSeries Integrator ESQL Reference* book, or SupportPac IC03 (see "MQSeries information available on the Internet" on page 173 for more details).

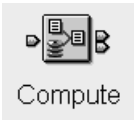
Using the check node

The Check node compares the format of a message arriving on its input terminal with a message-type specification that you supply when you configure the Check node. The message-type specification comprises any combination of the message domain, message set, and message type. The Check node checks only the message-type specification; it does not check the message body.

The following scenario illustrates one possible use for this node:

You might receive electronic messages from your staff at your head office. These messages are used for multiple purposes, for example to request technical support, stationery, or advising you about new customer leads. These messages can be processed automatically because your staff fill in a standard form. If you want these messages to be processed separately from other messages received, you use the Check node to ensure that only staff messages which have a specific message type are processed by this message flow.

Using the Compute node



The Compute node constructs a new message, or modifies elements (headers, header fields, and body data) within an existing message, or its associated destination or exception list, or both. These components of the message can be defined using an ESQL expression, and can be based on elements of both the input message and data from an external database. The expression can make use of arithmetic operators, text operators (for example, concatenation), logical operators, and other built-in functions.

The Compute node can be used

- To build a new message using a set of assignment statements
- To copy messages between parsers
- For data conversion
- For message transformation

For more detail about how to construct the ESQL to do each of these tasks, see the *MQSeries Integrator ESQL Reference* book.

The following scenario illustrates one possible use for this node:

You want to give each order you receive a unique identifier for audit purposes. The Compute node does not modify its input message, it creates a new, modified copy of the message as an output message. You can use the Compute node to insert a unique identifier for your order into the output message.

Using the Database node

The Database node allows a database operation in the form of an ESQL statement to be applied to the specified ODBC data source. Data from the input message can be substituted into the ESQL expression, and transforms can be applied to the data as part of that assignment. A node property controls whether the update to the database is committed immediately, or deferred until the completion of processing of the message flow at which time the update is committed or rolled back according to the overall completion status of the message flow.

The syntax of the statements that are accepted by the Database node is a subset of the statements that are accepted by a Compute node.

Like the Compute node, the Database node is configured using a series of statements. All of the normal compute statements such as SET, WHILE, DECLARE, and IF can be used to control the flow of the series of statements.

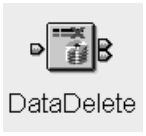
Unlike the Compute node, however, the Database node propagates the message that it receives at its input terminal to its output terminal unchanged. This means that, like the Filter node, there is only one message to be referred to in a Database node.

Because you can't modify any part of any message, the assignment statement (the SET statement, not the SET clause of the INSERT statement) can only assign values to temporary variables. Therefore the scope of actions you can take with an assignment statement is limited.

The following scenario illustrates one possible use for this node:

If you receive an order for 20 monitors, and have sufficient numbers of monitors in your warehouse, you want to decrement the stock level on your Stock database. You can use the Database node to check you have sufficient monitors available, and decrement the value of the quantity field in your database.

Using a DataDelete node



The DataDelete node is a specialized form of the Database node that allows deletion of one or more rows from a table in the specified ODBC data source. Data from the input message can be substituted into the ESQL expression, and transforms can be applied to the data as part of that assignment. A property controls whether the update to the database is committed immediately, or deferred until the completion of processing of the message flow at which time the update is committed or rolled back according to the overall completion status of the message flow.

This node offers an alternative way of deleting data from a database. You could also use the Database node.

The following scenario illustrates one possible use for this node:

You were running a limited promotion. The goods are only available for the period of the promotion, and each customer can only have one item. When stocks of the sale goods run out, you want to remove their details from the stock database, when a message containing an order for the last item comes in, the DataDelete node is triggered to remove all the details for that item from the database.

Using a DataInsert node

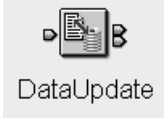
The DataInsert node is a specialized form of the Database node that allows insertion of one or more rows into a table in the specified ODBC data source. Data from the input message can be substituted into the ESQL expression, and transforms can be applied to the data as part of that assignment. A property controls whether the update to the database is committed immediately, or deferred until the completion of processing of the message flow at which time the update is committed or rolled back according to the overall completion status of the message flow.

This node offers an alternative way of inserting data into a database. You could also use the Database node.

The following scenario illustrates one possible use for this node:

You have developed a new product. The details about the product have been sent from your engineering department, and you need to extract details from the message and add them as a new row in your stock database.

Using a DataUpdate node



The DataUpdate node is a specialized form of the Database node that allows the modification of one or more rows in a table in specified ODBC data source. Data from the input message can be substituted into the ESQL expression, and transforms can be applied to the data as part of that assignment. A property controls whether the update to the database is committed immediately, or deferred until the completion of processing of the message flow at which time the update is committed or rolled back according to the overall completion status of the message flow.

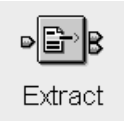
DataUpdate node

This node offers an alternative way of updating data in a database. You could also use the Database node.

The following scenario illustrates one possible use for this node:

You have already added details of a new product, a keyboard, to your stock database. Now you have received a message from the Goods In department which indicates that 500 keyboards have been delivered to your premises. You can use the DataUpdate node to change the quantity of keyboards in your database from zero to 500.

Using an Extract node



The Extract node derives an output message from an input message. The output message comprises only those elements of the input message that you specify for inclusion when configuring the Extract node.

The following scenario illustrates one possible use for this node:

When you receive orders from new clients, you might want to collect their names and addresses for future promotions. To do this you would extract their names and addresses from the orders, and send them as a new message to head office. These messages would be processed at head office so that the customer details can be included in the next marketing campaign.

Using a Filter node

The Filter node routes a message according to message content using a filter expression specified in ESQL. The filter expression can include elements of the input message or message properties. It can also use data held in an external database. The output terminal to which the message is routed depends on whether the expression is evaluated to true, false, or unknown.

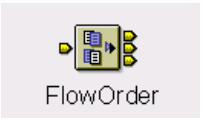
The filter node can route a message depending on a:

- Value in the message body
- Calculation, resulting in a value
- True or false being returned from an expression

The following scenario illustrates one possible use for this node:

There is an online test with 10 multiple choice questions. Each message coming in has a candidate name and address followed by a series of answers. Each answer is checked, and if it is correct the field ″SCORE″ is incremented by 1. Once all the answers have been checked, the field ″SCORE″ is tested to see if it is greater than 5. If it is, the filter node propagates the message to the flow which handles successful candidate input, otherwise the message is filtered into the rejection process, and a rejection message is created.

Using a FlowOrder node



The FlowOrder node enables you to control the order in which a message is processed by a message flow. You can use this node to specify the order in which each message is propagated to each (of two) output terminals. The message is only propagated to the second output terminal if propagation to the first output terminal is successful.

You can include this node in a message flow at any point where the order of execution of subsequent nodes is important. The FlowOrder node by definition propagates the input message through the first output terminal and its target node or nodes before propagating the input message to the second output terminal and its target node or nodes. The default behavior of all other nodes is for the order of propagation to any output terminal to be random and unpredictable.

If you connect multiple target nodes to the first terminal, or the second terminal or both, the order in which the multiple connections on each terminal are processed is random and unpredictable. However, the message is propagated to all target nodes connected to the first terminal, and all must complete successfully before it is propagated to any connection on the second output terminal.

The following scenario illustrates one possible use for this node:

You might want to use a FlowOrder node to control the sequence of branching when, for example, receiving orders from the internet. When the order is received, it is processed by nodes connected to the first terminal. The tasks performed here could be to debit the stock level in your database and raise an invoice. Finally a check is done to see if the customer has said his details can be sent to other suppliers. If the customer has indicated that he does not wish this information to be divulged, then this check fails and no further processing occurs. If however, the customer is happy for you to share his details with other companies (that is, the test is successful) then the input message is propagated to the second terminal so the customers details can be added to the mailing list.

Using the Input Terminal

The Input Terminal provides an in terminal for an embedded message flow.

From an Input Terminal, you can make a connection to any in terminal on any message flow node. The Input Terminal is the first node of a subflow that you can embed in another flow. An MQInput node is not required in a subflow. For more information about embedded flows, see “Including one message flow in another” on page 50.

The following scenario illustrates one possible use for this node:

A subflow might be used for common tasks, such as setting up a counter for a loop. The InputTerminal is the entry point into the subflow.

Using a Label node

The Label node is a named destination for a message processed by a RouteToLabel node. The Label node is identified by an entry in a destination list of the message when it is processed by a RouteToLabel node.

The combination of a RouteToLabel node with Label nodes provides a level of dynamic routing within a message flow, with the destination of the message following the RouteToLabel node being determined by the contents of the destination list within the message itself.

Typically, a label node connects to a subflow that processes each message in a specific way, and either ends in an output node or in another RouteToLabel node.

Figure 8 on page 67 illustrates a message flow that is made up of subflows that are associated with the main flow using RouteToLabel and Label nodes.

Using the MQeInput node

The MQeInput node reads messages from a specified ″bridge″ queue on the broker's MQSeries Everyplace queue manager (which must be created and configured before a message flow containing this node is deployed), and establishes the processing environment for the message.

If you plan to deploy message flows containing MQeInput nodes to a broker, you can only use one execution group, no matter how many message flows that is. The MQSeries Everyplace nodes in the flows must all specify the same MQSeries Everyplace queue manager name. You'll get an error on deploy otherwise.

MQeInput routes messages to the out terminal. If this fails, the message is retried. If the retry out expires, the message is routed to the failure terminal. If this is not connected, the message is written to the backout queue.

If the message is caught by this node after an exception has been thrown further on in the message flow, the message is routed to the catch terminal.

You must define a backout requeue queue or a dead letter queue (DLQ) within MQSeries Everyplace to prevent the message looping continuously through the node.

See *MQSeries Integrator Programming Guide* for information about writing MQSeries Everyplace applications which communicate with MQSeries Integrator.

The following scenario illustrates one possible use for this node:

A farmer in Australia checks his fields to see how well they are irrigated. He is carrying a Palm Pilot with MQSeries Everyplace installed. He sees an area of field requiring water, so using his Palm Pilot and a Global Satellite Navigation link, he sends a message to an MQeInput node. The data is manipulated using a Compute node, and a message is published so that a remote SCADA device can pick up the message and trigger the irrigation sprinklers. The farmer can see the water delivered to the field minutes after sending his message.

Using the MQeOutput node

The MQeOutput node forwards messages to MQSeries Everyplace queue managers. This node has three **Destination mode** options:

- **Queue Name**
- **Destination List**
- **Reply to Queue**

If a non-local destination queue manager is specified, ensure there is either a route to the queue manager, or store-and-forward queue servicing for the queue manager if it exists.

Using an MQInput node

The MQInput node uses MQGET to read a message from an MQSeries message queue defined on the broker's queue manager, and establishes the processing environment for the message.

MQInput routes messages to the out terminal. If this fails, the message is retried. If the retry out expires (as defined by the BackoutThreshold attribute of the input queue), the message is routed to the failure terminal. If this is not connected, the message is written to the backout queue.

If the message is caught by this node after an exception has been thrown further on in the message flow, the message is routed to the catch terminal.

You must define a backout requeue queue or a dead letter queue (DLQ) to prevent the message looping continuously through the node.

You must use one of the supplied MQInput nodes (MQInput, MQeInput, SCADAInput): you cannot replace it with a user-written equivalent.

The following scenario illustrates one possible use for this node:

Set the ″Convert″ property to ″yes″, so data conversion will be performed on the message received (in conformance with the CodedCharSetId and Encoding values set in the MQMD).

Using an MQOutput node

The MQOutput node uses MQPUT to write messages to an MQSeries message queue defined on any queue manager accessible by the broker's queue manager, or to the destinations identified in the destination list associated with the message.

You do not have to use this output node, you could use either of the MQSeries Integrator output nodes: MQOutput or MQeOutput.

The following scenario illustrates one possible use for this node:

You can set the node properties to specify that MQSeries can, if appropriate, break the message into segments in the queue manager.

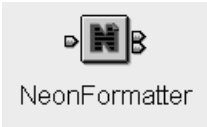
Using an MQReply node

The MQReply node is a specialized form of the MQOutput node that sends a response to the originator of the message by putting a message to the MQSeries queue identified by the ReplyToQ field of the message header.

The following scenario illustrates one possible use for this node:

This node is useful when receiving an order from a customer. When the order message is processed, a response is sent to the customer acknowledging receipt of his order and providing a possible date for delivery.

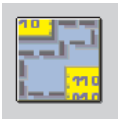
Using the NEONFormatter node



The NEONFormatter node is used transform a message from a known input format to a specified output format. The message definition and transformations are defined using the NEON Formatter graphical utility, not the MQSeries Integrator Control Center. This node is deprecated and has been superseded by the NEONMap and NEONTransform nodes. You are not recommended to use this node except to maintain backwards compatability with existing message flows.

See “Chapter 17. Concepts of NEONRules and NEONFormatter Support for MQSeries Integrator” on page 145 for more information.

Using the NEONMap node



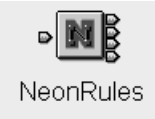
The NEONMap node performs exactly the same function as the NEONTransform node except that any output operations associated with the specified Target Format are not applied to the output message. See “Chapter 17. Concepts of NEONRules and NEONFormatter Support for MQSeries Integrator” on page 145 for more information concerning the use of the NEONMap node.

See “The NEONMap node” on page 149 for more information.

The following scenario illustrates one possible use for this node:

You wish to reformat a input format message to an output format for the purposes of processing the message. However the output format has output operations associated with it that cause extra formatting information, such as delimiters or tags, to be added to the output message data. The NEONMap node may be employed to reformat the message without applying these operations, thus allowing it to be used as a generic content model format without the need to alter its definition in the NEONFormatter User Interface.

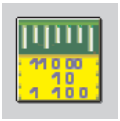
Using the NEONRules node



The NEONRules node provides an encapsulation of the NEONRules engine within MQSeries Integrator 2.0. The firing of the Propagate action propagates the output message to the propagate terminal. The firing of the PutQueue action attaches a queue name to the destination list associated with the message and routes the message to the putqueue terminal. This node is deprecated and has been superseded by the NEONRulesEvaluation node. You are not recommended to use this node except to maintain backwards compatability with existing message flows.

See “Chapter 17. Concepts of NEONRules and NEONFormatter Support for MQSeries Integrator” on page 145 for more information.

Using the NEONRulesEvaluation node



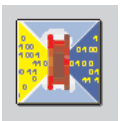
The NEONRulesEvaluation node implements the processing and execution of the Rules defined in the NEONRules User Interface. These rules transform and redirect incoming messages based on the format, application group and content of the messages. See the NEONRules and NEONFormatterSupport documentation for more information regarding Rule definition. Three of the actions which a Rule may invoke include Put Queue, Route and Propagate. These result in the output message being sent to the putqueue, route and propagate terminals of the NEONRulesEvaluation node respectively. The Put Queue and Route actions also cause the Destination List of the output message to be appropriately configured for it to be processed by an MQOutput or RouteToLabel node. Messages are sent to the nohit terminal when they fail to meet the criteria for any of the defined Rules.

See “The NEONRulesEvaluation node” on page 150 for more information.

The following scenario illustrates one possible use for this node:

You wish to place incoming messages to various MQSeries queues depending on the value of a field within the message. The NEONRulesEvaluation node may be employed to do this. You must first define an appropriate input format for the incoming messages in the NEONFormatter User Interface, then define a Rule in the NEONRules User Interface which executes various Put Queue actions depending on the contents of the required field. The putqueue terminal of the NEONRulesEvaluation node must be connected either directly or indirectly to an MQOutput node configured with a Destination Mode of ″Destination List″ in order for the Put Queue action to have any effect.

Using the NEONTransform node



The NEONTransform node is used to transform a message from a known input format to a specified output format. The message format definitions are defined using the NEONFormatter graphical utility, not the MQSeries Integrator Control Center.

See “The NEONTransform and NEONMap nodes” on page 147 for more information.

NEONTransform node

The following scenario illustrates one possible use for this node:

You wish to translate a message with a wire format suitable for one system into a wire format suitable for a second, different, system. This may be accomplished by defining both wire formats within the NEONFormatter User Interface and setting the Target Format attribute of the NEONTransform node to the name of the required output format. For more information concerning the definition of input and output formats, and the mappings between them, see the NEONRules and NEONFormatter Support documentation.

Using the Output Terminal

The Output Terminal provides an out terminal for an embedded message flow.

An Output Terminal can only receive connections from a message flow node. The Output Terminal is the last node of a subflow that you can embed in another flow. A subflow can be set up with more than one Output Terminal. When the subflow is included in an embedding flow, the number of out terminals displayed represents the number of Output Terminals in the embedded subflow. Each out connector is labeled with the name you assigned to the Output Terminal node. For more information about embedded flows, see “Including one message flow in another” on page 50.

The following scenario illustrates one possible use for this node:

A subflow might be used for common tasks, such as setting up a counter for a loop. The OutputTerminal is the exit point from the subflow.

Using the Publication node

The Publication node filters and transmits the output from a message flow to subscribers who have registered an interest in a particular set of topics.

The Publication node must always be an output node of a message flow and has no output terminals of its own.

The default publish/subscribe message flow provides a simple publish/subscribe service. It emulates exactly the basic publish/subscribe function supported by the Publish/Subscribe SDK, and is appropriate for all publish/subscribe services in which no additional processing of the message content is required. For more information see “Using the IBM supplied message flows” on page 129.

The operation of the two nodes within this node is:

1. The **Get next message** node (type MQInput) gets the next available message from the input queue and passes it to the Publication node for matching against the table of subscription requests. The input queue is initially defined to be SYSTEM.BROKER.DEFAULT.STREAM, but you can change this according to your requirements.

   Failures in this node are not handled explicitly: the failing message is put to a backout queue or dead letter queue (if these queues have been defined). You can change this behavior by connecting other nodes to the failure terminal of this node, if you want to.

2. The **Route to matching subscribers** node (type Publication) matches the inbound publication against its internal subscription table (created and maintained in response to client subscription requests).

   For each matching subscription, the message is delivered to the subscriber by putting it to the queue on the queue manager specified in the subscription.

If you want to deploy this default message flow, you are recommended to make a copy of it. This preserves the default message flow in your configuration repository for future reuse.

Using the ResetContentDescriptor node

The ResetContentDescriptor node takes the bit stream of the input message and reparses it using a different message template and either the same or a different parser. The node can reset any combination of message domain, set, type, and format.

The following scenario illustrates one possible use for this node:

You can use a ResetContentDescriptor node to swap between two message parsers in the middle of a message flow. For example, you can swap between the BLOB and the MRM parser.

The format of an incoming message might be unknown when it enters a message flow, so the BLOB parser is invoked. Later on in the message flow, you might decide that the message is defined in the MRM format and you can use the ResetContentDescriptor node to reparse the message using the MRM parser.

The following table shows typical ResetContentDescriptor node attributes:

Table 2. Typical ResetContentDescriptor node attributes

messageDomain	MRM
resetMessageDomain	check
messageSet	DH53CU406U001
resetMessageSet	check
messageType	m_MESSAGE1
resetMessageType	check
messageFormat	CWF
resetMessageFormat	check

The messageDomain is set to MRM, which causes the MRM parser to be invoked. The message set, type, and format are the message template values that define the message format in MRM, and all the reset boxes are checked because all the properties need to change.

The ResetContentDescriptor node in the previous example causes the incoming BLOB parser to construct the physical bitstream of the message (not the logical tree representation of it) and pass this bitstream to the MRM parser. The MRM parser then parses the bitstream using the message template given above (messageSet, messageType, and messageFormat).

It is important to note that the ResetContentDescriptor node does NOT:

- Change the physical bit stream, only the way in which the bitstream is parsed in MQSeries Integrator.
- Convert the message from one format to another. For example, if the incoming message has a messageFormat of XML and the outgoing messageFormat is CWF, the ResetContentDescriptor node does not do any reformatting. It attempts to parse the bitstream of the incoming XML message as a CWF message. This results in all the XML tags remaining in the message and the reparse will fail.

Using a RouteToLabel node

The RouteToLabel node provides a dynamic routing facility based on the contents of the destination list associated with the message. The destination list contains the identity of one or more target Label nodes, identified by their **Label Name** property (not the node name).

The destination, defined by the **Label Name** of a Label node, is resolved by the broker itself during message flow processing: you do not connect a terminal on the RouteToLabel node to the destination nodes.

A RouteToLabel node uses a destination list within a message to route the message to a target node of type Label that matches the label within the destination list item. Therefore the message must include a destination list to be acted on by the RouteToLabel node.

You must create the destination list, and include it in the message, in a Compute node. You must select the *Advanced* tab on the Compute node properties dialog, and select an option, for the Compute node, that includes *Destination* from the drop-down list.

The destinations are set up as a list of label names. The label names can be any string value, and can be explicitly specified in the Compute node or taken or cast from any field in the message or from a database. A label name in the destination list must, however, match the Label Name property of a corresponding Label Node.

Figure 8 on page 67 illustrates a message flow that uses these techniques to achieve dynamic routing based on message content. The flow is made up of subflows that are associated with the main flow using RouteToLabel and Label nodes.

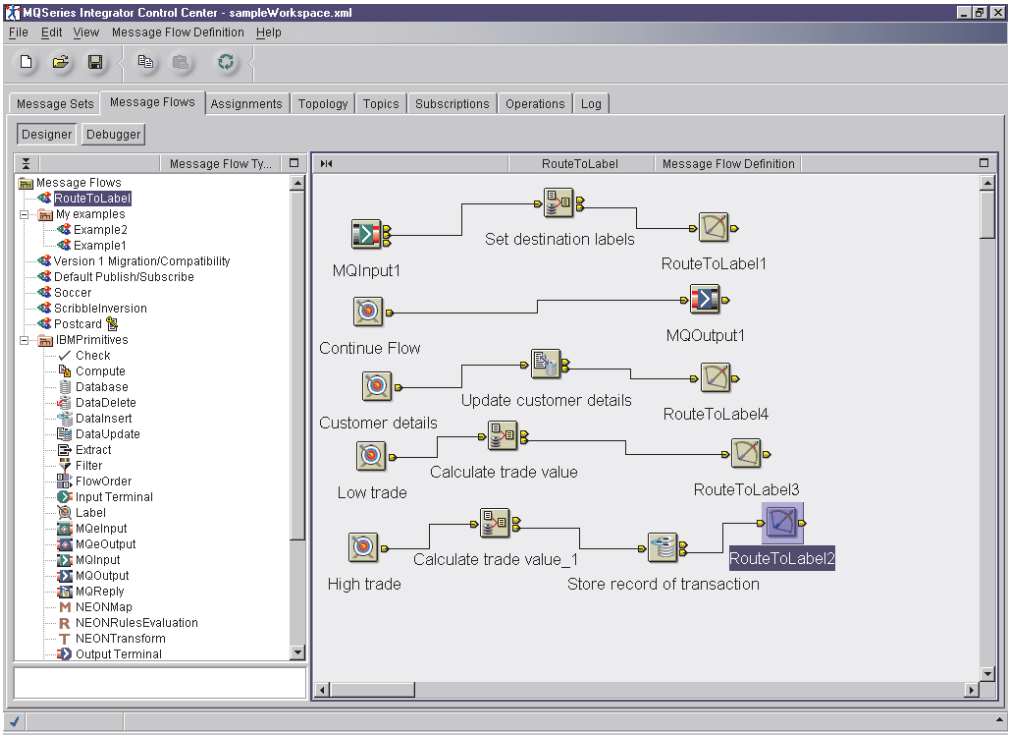


Figure 8. A message flow with RouteToLabel and Label nodes. This shows the nodes connected 'Manhattan Style' (you can select this style of connecting the nodes by right clicking on the pane background).

The message flow shows how you might handle brokerage transactions if you want to process high-value trading requests, low-value trading requests and requests to updates customer details differently.

The use of RouteToLabel and Label nodes makes a simpler message flow than the message flow you would need if you used a sequence of Filter nodes that identify and route the message for different processing, or a sequence of nodes that each performs an action on a subset of the total number of messages processed by the message flow.

Each message in this brokerage example has a request field that indicates whether the message contains ″hightrade″, ″lowtrade″, or ″custdetails″ information. Each type is routed to a different sequence of nodes before being completed by a common flow.

You configure the compute node ″Set destination labels″ to create a destination list in the message by entering the following ESQL:

```
SET OutputDestinationList.Destination.RouterList.DestinationData[1].labelname
 = 'continue';
SET OutputDestinationList.Destination.RouterList.DestinationData[2].labelname
 = "InputBody.MRM.trademsg.request";
```

If you set Route To Last on the RouteToLabel node, a message is routed to the last label in the destination list. In this example, that is the label that matches the value of the ″request″ field in the message. Therefore a message with a value of ″hightrade″ in the request field is routed to the Label node with a Label Name property of ″hightrade″.

RouteToLabel node

If the message fragment performing the dynamically routed work itself ends in a RouteToLabel node, the message is passed to the next destination in the list. In the example above, the message is passed to the Label node with a Label Name property of ″continue″, and continues along the common part of the message flow.

There are four message flow fragments, beginning with a Label node:

1. Continue (Label Name property = continue). This fragment does not end in a RouteToLabel node. It might end in an MQOutput node to complete the message flow execution or it might continue with work that is not required to be dynamically routed. Typically, the Label Name of this fragment is the one at index [1] of the destination list. This means that, when all the dynamically routed work is complete, the flow either finishes or continues with common processing.
2. High Trade (Label Name property = hightrade). This fragment performs processing specific to high-value trading requests, for example storing records of the trade in a database and performing credit authorizations. This fragment ends in a RouteToLabel node to send the message on to the next destination in the list.
3. Low Trade (label name property = lowtrade). This fragment performs processing specific to low-value trading requests. It ends in a RouteToLabel node to send the message on to the next destination in the list.
4. Customer Details (Label Name property = custdetails). This performs processing specific to requests to update customer details, for example, updating a customer details database. This fragment ends in a RouteToLabel node to send the message on to the next destination in the list.

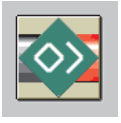
The flexibility of this dynamic routing facility enables an infinite number of variations on the above scenario.

Message flows can be self-contained, or can include subflows within them. Where the message flow contains no ″lower Level″ flows, the Label nodes and their subflows which follow must be defined in the Message Flow Definition pane for that message flow. Where a message flow contains other message flows, the Label nodes and their subflows which follow must be defined in either, the Message Flow Definition pane of the ″lower level″ message flow, or the Message Flow Definition pane of the ″high level″ message flow.

This ensures that all the Label nodes are included when you deploy the message flow. The Label node is not connected to a prior node: if you create a subflow that starts with any other type of node, the subflow defined in the Message Flow Definition pane is ignored when the message flow is deployed. Subflows that start with a Label node are not ignored.

If you intend to derive destination values from the message itself, or from a database, you might also need to cast values from one type to another. Casts and destination lists are described in more detail in *MQSeries Integrator ESQL Reference*.

Using the SCADAInput node



The SCADAInput node receives a message from a client connecting using the MQIsdp protocol into the format recognized by MQSeries Integrator and establishes the processing environment for the message.

If you plan to deploy message flows containing SCADA nodes to a broker, you can only use one execution group, no matter how many message flows that is.

A SCADA listener can be started and stopped using publish and subscribe messages with a specific topic. This can be done for all ports, or, for a single port identified in the message.

The following scenario illustrates one possible use for this node:

The SCADAInput node receives messages each minute from a remote machine. The messages contain details of the machine's switch settings. The data received is fed into a ResetContentDescriptor node to cast the data from binary to MRM message format. The information about the machine is stored in a database using the Database node, and then ″enriched″ using a Compute node to create an XML message which is Published using a Publication node.

Since XML messages are expensive to send (because satellite transmission has a high cost per byte), it is advantageous to use this method as data is enriched by the broker.

Note: If you want to process the data in an incoming SCADA message, then it will be necessary to use a node like the ResetContentDescriptor.

Using the SCADAOutput node

The SCADAOutput node sends a message to a client connecting using the MQIsdp protocol. Only occasionally, in advanced applications, might you need to use the SCADAOutput node directly; for example, if you want to write your own Publication node.

If you plan to deploy message flows containing SCADA nodes to a broker, you can only use one execution group, no matter how many message flows that is.

Using a Throw node

The Throw node provides a mechanism for throwing an exception within a message flow. The exception might be caught and processed by a preceding TryCatch node within the message flow, or handled by the MQInput node.

The following scenario illustrates one possible use for this node:

You might want to use the Throw and TryCatch nodes where you are using the Compute node to calculate a total. You might want to create a message that is sent to your system administrator when the total calculated exceeds the maximum value for the Total field.

Using a Trace node

The Trace node generates trace records that can incorporate text, message content, and date and time information, to help you to monitor the behavior of the message flow.

The operation of the Trace node is independent of the setting of user tracing for the message flow in which it is included: output from the trace node is written even if user tracing is set off. In particular, if user tracing is set to `None`, and a Trace node in that message flow has its Destination property set to `User Trace`, the entries written by the Trace node are recorded in the user trace log.

An example of the trace output is below:

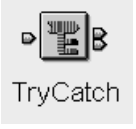
```
(0x1000010)XML        = (
  (0x5000018)XML        = (
    (0x6000011) = '1.0'
  )
  (0x1000000)MESSAGETAG = (
    (0x3000000)highlevel   = '1'
    (0x3000000)higherlevel = '21'
    (0x1000000)m_m1        = (
      (0x3000000)level2     = '2'
      (0x1000000)e_string01 = (
        (0x3000000)level3 = '3'
        (0x2000000)       = 'IAPMDI27 ddddddd'
```

This can be interpreted as follows:

1. The numbers in brackets at the left hand end of each line are element types. These numbers, which do not display leading zeros, are defined as follows:
   - Bits 0 to 7 identify the specific element type. See the parser descriptions in*MQSeries Integrator ESQL Reference* for details about element types.
   - Bits 8 to 23. These bits are not used.
   - Bits 24 to 31 identify the generic element type:
     – Bit 24 indicates that the element has a name.
     – Bit 25 indicates that the element has a value.
     – Bit 26 indicates that the element is meaningful only to its parser type (therefore it is not copied to others).
2. The names after the brackets are element names.
3. The values after the equal sign are the element values. These values are displayed as ESQL literals. Where a node contains other nodes this is indicated by the nesting.

Note: Floating point numbers, which are stored internally as IEEE double precision are rounded to 15 decimal places when output to a trace.

Using theTryCatch node



The TryCatch node provides a special handler for exception processing. The input message is initially routed on the try terminal of this node. If an exception is subsequently thrown by a downstream node, it is caught by this node, which then routes the original message to its catch terminal. If the TryCatch node catch terminal is not connected, the message is discarded.

The following scenario illustrates one possible use for this node:

You might want to use the Throw and TryCatch nodes where you are using the Compute node to calculate a total. You might want to create a message that is sent to your system administrator when the total calculated exceeds the maximum value for the Total field.

Warehouse node

The Warehouse node is a specialized form of the Database node that stores the entire message, or parts of the message, or both, to the specified ODBC data source.

You can use a message warehouse:

- To maintain an audit trail of messages
- For offline or batch processing of messages (a process sometimes referred to as *data mining*)
- To enable subsequent reprocessing of selected messages

Once stored in the message warehouse, messages can be retrieved using standard database query and data mining techniques. No explicit support for these functions is supplied by MQSeries Integrator.

You can choose to store in the message warehouse:

- The entire message
- Selected parts of the message

Storing the entire message

When you store the entire message in a message warehouse, it is stored as a binary object. You can choose to store a timestamp for the message, though this is optional. Any timestamp is stored in a separate column from the message itself.

The advantages of storing the entire message are:

- You do not have to have decided how you will use the data before you store it.
- You do not have to have defined a database schema for every type of message that could pass through the broker.

However, you could consider preceding each Warehouse node with a Compute node that would convert each message to a common schema before the Warehouse node stores it.

Storing parts of the message

If you store selected parts of a message, with a timestamp if required, you must define a database schema for that message type. The message is mapped to true type: for example, a character string in a message is stored as a character string.

Using the Warehouse node to store the entire message

To configure the Warehouse node to store the entire message:

1. In the Message Flow Definition pane, right click the symbol of the Warehouse node you want to configure and click **Properties**.
2. In the **Warehouse** dialog, click **Add** to define the input message.

   The **Add** dialog is displayed.

3. In the **Add** dialog, **Message** is preselected. Select the names of a message set and message from the drop-down lists. Click **OK**.

   The message tree structure appears in the Input pane. A tab is added to the Input pane showing the name of the message.

   Repeat this step for additional messages. To remove a message from the Input pane, click **Delete** when the relevant tab is to the fore.
4. Click **Add** to define the Output.

   The **Add** dialog is displayed. **Database table** is preselected.
5. In the **Add** dialog, enter Data Source and Table Name values. Click **OK**.

   The database tree structure is shown in the Output pane. You can name only one database in this pane. To delete table and database names, click **Delete**.
6. Now you must identify the columns you want to work with within the database table you identified. To do this:
   a. Right click anywhere in the white space around the database tree structure in the Output pane, and click **Add column**.

      The **Enter database column** dialog is displayed.
   b. Click in the Column field of the dialog, then enter the column identifier.
   c. Click **OK**.

      The column is added to the database tree structure in the Output pane.

   Repeat this process for each column you want to work with. (You need entries for only those columns you will be using, even if additional columns exist in the database.)

   Note that there no validation is done on these values at this stage: the existence of the database, tables, and columns that you specify here cannot be determined until the message flow is deployed and executed within a broker.
7. Select the **Store Message** check box, and select the column in which you want to store the index record and attached binary object.
8. From the **Transaction Mode** drop-down list, select automatic or commit.
9. If you want to store a timestamp, select the **Store Timestamp** check box and select the column in which you want to store it.
10. If you want warnings to be treated as errors, click the **Advanced** tab of the **Warehouse** dialog, and select the **Treat warnings as errors** check box.
11. If you want to provide a description of this instance of the Warehouse node (which is recommended if you want other Control Center users to be able to make use of it), click the **Description** tab of the **Warehouse** dialog. Type a short description, or a long description, or both.
12. Click **OK** to finish configuring this Warehouse node.

Using the Warehouse node to store parts of a message

To configure the Warehouse node to store parts of a message message:

1. In the Message Flow Definition pane, right click the symbol of the Warehouse node you want to configure and click **Properties**.
2. In the **Warehouse** dialog, click **Add** to define the input message.

   The **Add** dialog is displayed.
3. In the **Add** dialog, **Message** is preselected. Select the names of a message set and message template from the drop-down lists. Click **OK**.

The message tree structure appears in the Input pane. A tab is added to the Input pane showing the name of the message.

Repeat this step for additional messages. To remove a message from the Input pane, click **Delete** when the relevant tab is to the fore.

4. Click **Add** to define the Output.

   The **Add** dialog is displayed. **Database table** is preselected.

5. In the **Add** dialog, enter Data Source and Table Name values. Click **OK**.

   The database tree structure is shown in the Output pane. You can name only one database in this pane. To delete table and database names, click **Delete**.

6. Now you must identify the columns you want to work with within the database table you identified. To do this:

   a. Right click anywhere in the white space around the database tree structure in the Output pane, and click **Add column**.

      The **Enter database column** dialog is displayed.

   b. Click in the Column field of the dialog, then enter the column identifier.

   c. Click **OK**.

      The column is added to the database tree structure in the Output pane.

   Repeat this process for each column you want to work with. (You need entries for only those columns you will be using, even if additional columns exist in the database.)

   Note that there no validation is done on these values at this stage: the existence of the database, tables, and columns that you specify here cannot be determined until the message flow is deployed and executed within a broker.

7. Drag components of the input data from the Input pane to the target database column in the Output pane. This process is known as mapping, and represents the ESQL mappings that will be used in the processing of data through the node. The mappings are shown in the Input Message ESQL and Output Message ESQL pane. To delete mappings, right click on the expression to delete and click **Delete**. To delete all the expressions in the pane, click **Delete All**.

8. From the **Transaction Mode** drop-down list, select automatic or commit.

9. If you want to store a timestamp, select the **Store Timestamp** check box and select the column in which you want to store it.

10. If you want warnings to be treated as errors, click the **Advanced** tab of the **Warehouse** dialog, and select the **Treat warnings as errors** check box.

11. If you want to provide a description of this instance of the Warehouse node (which is recommended if you want other Control Center users to be able to make use of it), click the **Description** tab of the **Warehouse** dialog. Type a short description, or a long description, or both.

12. Click **OK** to finish configuring this Warehouse node.

Warehouse node

Chapter 6. Defining the broker Topology

This chapter describes how to:

- Check out and check in the Topology
- Create a broker
- Connect brokers
- Delete the connection between brokers
- Delete a broker from the Topology
- Rename a broker
- Making changes operational

The Topology view

To display the **Topology** view, click the **Topology** tab in the Control Center.

The Topology view has two panes:

- The left-hand pane, the Domain Hierarchy pane, shows a tree view of the Topology of this broker domain.
- The right-hand pane, the Topology pane, contains an arrangement of graphical symbols that represent the current Topology.

Either double-click on any item in the Topology pane to see its properties, or right click and select **Properties** from the pop-up menu on any object in either pane.

Controlling the appearance of the Topology pane

When you populate the broker domain in your workspace, graphical symbols representing collectives and brokers are added to the Topology pane. You can control the appearance and arrangement of these symbols by right-clicking in the Topology pane to display the **Topology** list, and selecting from the following actions:

Layout graph
Arranges the connected brokers and collectives in the Topology pane from left to right, right to left, top to bottom, or bottom to top.

Your layout settings are saved on exit, and used when restarting applications.

Zoom
Alters the size of all broker and collective symbols in the Topology pane.

Your zoom settings are saved on exit, and used when restarting applications.

Manhattan style
Shows connections between brokers as lines at right angles.

Your style settings are saved on exit, and used when restarting applications.

Snap to grid
Aligns the symbols in the Topology pane on an invisible grid.

The settings for aligning your symbols are saved on exit, and used when restarting applications.

Checking out the Topology

The remainder of this chapter describes tasks that alter the Topology of the broker domain. You cannot perform any of these tasks unless you have exclusive access to the Topology document, which you obtain by checking the Topology out of the configuration repository.

To check out the Topology:

1. In the Domain Hierarchy pane of the **Topology** view, right click the root of the Topology tree.

   Note: Alternatively, you can right click anywhere on the background of the Topology pane, or you can highlight the root of the Topology tree and click on the **Domain Hierarchy** menu in the menu bar.
2. Click **Check Out**.

The **Key** icon appears to the right of the root in the Topology tree to confirm that the Topology document is checked out. You can now update the Topology. Other users with access to this broker domain by another instance of the Control Center cannot make changes to the Topology while it remains checked out to you.

The collectives that you define in the Topology are considered part of the Topology itself for check out and check in purposes. Therefore if you want to update a collective, for example to add a broker to it, you must check out the Topology. You cannot check out the collective as a separate resource. See “Adding an existing broker to a collective” on page 78 and “Removing a broker from a collective” on page 80 for further details about updating collectives.

Creating a broker

To create a broker in the configuration repository:

1. Ensure that you have checked out the Topology, as described in “Checking out the Topology”.
2. In the Domain Hierarchy pane of the **Topology** view, right click the root of the Topology tree.

   Note: Alternatively, you can right click anywhere on the background of the Topology pane, or you can highlight the root of the Topology tree and click the Topology menu in the Control Center menu bar.
3. Click **Create —> Broker**.

   The **Create a new Broker** dialog, is displayed.
4. In the **Name** field, type the name of your broker.

   This must be exactly the name specified when the broker was created (that is, the broker name specified on the **mqsicreatebroker** command). You must specify the name using the same case (lower, upper, or mixed). This value is required and must be unique.
5. In the **Queue Manager** field, type the name of the broker’s queue manager.

   This must be exactly the name specified for the broker’s queue manager when the broker was created (that is, the queue-manager name specified on the **mqsicreatebroker** command). You must specify the name using the same case (lower, upper, or mixed). This value is required and must be unique in your MQSeries network.

6. For documentation purposes, you can provide either a short description, or a long description, or both, of your broker, though a description is not required.

   If you want to provide a description, click the **Description** tab in the **Create a new Broker** dialog, and type some text.
7. Click **Finish** in the **Create a new Broker** dialog to complete creation of this broker.

Confirmation that your new broker has been created appears in two places in the **Topology** view:

- An entry representing the broker appears under the root of the Topology tree in the Domain Hierarchy pane. The **New** icon next to the broker entry indicates that this new definition has not yet been checked into the shared configuration.
- A graphical symbol of the broker appears in the Topology pane.

It also appears in the Domain Topology pane of the Assignments view.

If you have no further Topology changes to make, check in the Topology as described in “Checking in the Topology” on page 82.

Collectives

A **collective** is a set of one or more brokers that are directly connected to each other. For more information, please see the *MQSeries Integrator Introduction and Planning* book.

Creating a collective

To create a collective in the configuration repository:

1. Ensure that you have checked out the Topology, as described in “Checking out the Topology” on page 76.
2. In the Domain Hierarchy pane of the **Topology** view, right click the root of the Topology tree. (Alternatively, you can right click anywhere on the background of the Topology pane, or you can highlight the root of the Topology tree and click the Topology menu in the Control Center menu bar.)
3. Click **Create —> Collective**.

   The **Create a new Collective** dialog is displayed.
4. In the **Name** field, type the name of your collective. This must follow the naming rules described in “Naming Control Center resources” on page 11 and must be unique within your broker domain.
5. Click **Finish** in the **Create a new Collective** dialog to complete creation of this collective.

Confirmation that your new collective has been created appears in two places in the **Topology** view:

- A folder representing the collective appears under the root of the Topology tree in the Domain Hierarchy pane. The **New** icon next to the collective entry indicates that this new definition has not yet been checked into the shared configuration.
- A graphical symbol representing the empty collective appears in the Topology pane.

If you have no further topology changes to make, check in the Topology as described in “Checking in the Topology” on page 82.

Note that collectives are checked in as part of the Topology check in, not as separate resources, as they exist only in the Topology document.

Adding an existing broker to a collective

There are several ways of adding an existing broker to a collective using the Control Center. This section describes one of these methods in detail, then mentions others briefly.

When you add brokers to the collective, the collective symbol in the Topology pane can appear crowded. To increase the size of the collective symbol, drag the double-headed arrow at the bottom-right corner of the symbol downward.

To add an existing broker to a collective:

1. Ensure that you have checked out the Topology, as described in “Checking out the Topology” on page 76. (You cannot check out the collective.)
2. Right click the collective folder in the Topology tree.
3. Click **Add —> Broker**.

   The **Add an existing Broker** dialog is displayed.

   - To select a single broker, double-click on a broker name.
   - To select multiple brokers:
     a. If they appear sequentially in the list, click on the first broker you want, press and hold the Shift key, then click on the last broker you want. This action selects the two brokers you highlighted, plus any that appear between these two in the list.
     b. If they do not appear in a sequence in the list, hold down Ctrl and click each broker you want.
     c. When you have selected the brokers you want to add to the collective from this list, click **Finish**.

Confirmation that the selected brokers have been added to the collective appears in two places in the **Topology** view:

- In the Domain Hierarchy pane, the brokers are now shown under the collective folder.
- In the Topology pane, the broker symbols now appear inside the collective symbol.

Alternatively, you can invoke the **Add an existing Broker** dialog from the **Topology** menu in the Control Center menu bar.

You can also add an existing broker to a collective simply by:

- Dragging the broker symbol in the Topology pane into the symbol of the collective in the same pane

  or
- Dragging the broker entry in the Domain Hierarchy pane into the symbol of the collective in the Topology pane.

Note: the collective icon should be big enough for the broker icon to fit within the light blue shading of the collective icon in the Topology pane, otherwise it will not be added. A broker that has a cross-hair symbol behind it has been added successfully.

If you have no further topology changes to make, check in the Topology as described in “Checking in the Topology” on page 82.

Creating a broker to add to a collective

To create a broker to add to a collective:

1. Ensure that you have checked out the Topology, as described in “Checking out the Topology” on page 76.
2. In the Domain Hierarchy pane or Topology pane, right click on the collective folder.

   Note: When the Topology pane has the focus, the **Topology** menu appears in the Control Center menu bar. When the Domain Hierarchy pane has the focus, the **Domain Hierarchy** menu appears in the Control Center menu bar. The menu items of the **Topology** and **Domain Hierarchy** menus are identical.
3. Click **Create —> Broker**.

   The **Create a new Broker** dialog is displayed.

   Note: You can also select the **Create —> Broker** action by highlighting the collective symbol in the Topology pane, then clicking the **Topology** menu in the Control Center menu bar.
4. In the **Name** field, type the name of your broker.

   This must be exactly the name specified when the broker was created (that is, the broker name specified on the **mqsicreatebroker** command). You must specify the name using the same case (lower, upper, or mixed). This value is required and must be unique.
5. In the **Queue Manager** field, type the name of the broker’s queue manager.

   This must be exactly the name specified for the broker’s queue manager when the broker was created (that is, the queue-manager name specified on the **mqsicreatebroker** command). This value is required and must be unique in your MQSeries network.
6. For documentation purposes, you can provide either a short description, or a long description, or both, of your broker, though a description is not required.

   If you want to provide a description, click the **Description** tab in the **Create a new Broker** dialog, and type some text.
7. Click **Finish** in the **Create a new Broker** dialog to complete creation of this broker.

Confirmation that your new broker has been created appears in two places in the **Topology** view:

- An entry representing the broker appears under the appropriate collective folder in the Domain Hierarchy pane. The **New** icon next to the broker entry indicates that this new definition has not yet been checked into the shared configuration.
- A graphical symbol of the broker appears inside the symbol of the appropriate collective in the Topology pane.

The broker has been both created and added to the collective.

If you have no further topology changes to make, check in the Topology as described in “Checking in the Topology” on page 82.

Removing a broker from a collective

To remove a broker from a collective:

1. Ensure that you have checked out the Topology, as described in “Checking out the Topology” on page 76. (You cannot check out the collective.)
2. In the Topology pane, right click the symbol of the broker inside the collective symbol.
3. Click **Remove**.

Confirmation that the broker has been removed from the collective appears in two places in the **Topology** view:

- In the Domain Hierarchy pane, the broker is no longer shown under the collective folder.
- In the Topology pane, the broker symbol now appears outside the collective symbol.

Alternatively, you can drag the broker symbol out of the symbol of the collective in the Topology pane. You can also right click the broker entry under the relevant collective in the Domain Hierarchy pane, and click **Remove**.

If the removed broker was connected to a broker outside the collective, you might need to remove the connection also. For more information, see “Deleting the connection between brokers” on page 81.

If you have no further topology changes to make, check in the Topology as described in “Checking in the Topology” on page 82.

Connecting brokers

You need to connect brokers so that the brokers know who their neighbors are in the publish/subscribe topology network. A connection is created only if a cycle of connections would not result. If the addition of a connection would cause a cycle, an error message is issued. Before making any connections you should be aware of the following rules:

- A single broker, outside a collective, can be connected to just one broker inside a collective.
- You can connect a single broker outside a collective to multiple collectives (that is, to one broker per collective).
- You can connect a broker in one collective to a broker in another collective.
- You can connect two brokers outside a collective.
- Brokers within a collective cannot be explicitly connected, because they are already implicitly connected.

To connect one broker to another:

1. Ensure that you have checked out the Topology, as described in “Checking out the Topology” on page 76.
2. In the Topology pane, right click the symbol of one of the two brokers you want to connect.
3. Click **Connect —> port**.

   The cursor becomes a cross-hair attached by a red line to the broker you selected initially.
4. Move the cross-hair to the symbol of the broker you want to connect to, and click.

Note: Broker connections can have bend points just like node connections. See "Creating bend points" on page 46 for more information.

The brokers are now connected. In the Topology pane, a line connects the symbols of the two brokers.

If you have no further Topology changes to make, check in the Topology as described in "Checking in the Topology" on page 82.

Deleting the connection between brokers

To delete the connection between two brokers:

1. Ensure that you have checked out the Topology, as described in "Checking out the Topology" on page 76.
2. In the Topology pane, right click on the line between the two brokers you want to disconnect.
3. Click **Delete**.

The line between the two brokers disappears. The brokers are now disconnected.

If you have no further Topology changes to make, check in the Topology as described in "Checking in the Topology" on page 82.

Deleting a broker from the Topology

This procedure describes how to delete a broker reference from the configuration repository. This procedure does not delete the broker from your system: it simply marks the broker as logically deleted from the configuration repository. For a full description of the process required to delete a broker from your broker domain, see "Deleting a broker from the broker domain" on page 93.

1. Ensure that you have checked out the Topology, as described in "Checking out the Topology" on page 76. If this is not done the configuration repository will not be updated with the change.
2. Ensure that the broker you want to delete is either checked in or new. If it is not (that is, if the **Key** icon is displayed next to its entry in the Domain Hierarchy pane), right click the broker entry and click **Check In**. All execution groups assigned to the broker must also be checked in before you can delete the broker (this can be done from the Assignments view or by using **All (Save to Shared)**).
3. In the Topology pane, right click the broker you want to delete.
4. Click **Delete**.
5. A confirmation message is displayed. If you want to proceed with the deletion, click **Yes**.

Confirmation that the broker has been deleted appears in two places in the **Topology** view:

- The broker entry no longer appears in the Domain Hierarchy pane.
- The broker symbol no longer appears in the Topology pane.

If the broker was connected to another, the connection is also deleted.

If you have no further Topology changes to make, check in the Topology as described in "Checking in the Topology" on page 82.

Renaming a broker

You might need to rename a broker if your original attempt at creating a broker reference contained an error: renaming the broker is simpler than deleting and recreating it.

To rename a broker:

1. Ensure that you have checked out the Topology, as described in “Checking out the Topology” on page 76.
2. Ensure that the broker you want to rename is checked out. If it is not (that is, if neither the **Key** icon nor the **New** icon is displayed next to its entry in the Domain Hierarchy pane), right click the broker entry in the Topology tree and click **Check Out**.
3. In the Topology pane, right click the broker you want to rename.
4. Click **Rename**.

   The **Rename Broker** dialog is displayed.
5. In the New name field, type the new name of the broker. This must be exactly the name specified on the **mqsicreatebroker** command. Click **Finish**.

Confirmation that the broker has been renamed appears in two places in the **Topology** view:

- The broker entry in the Domain Hierarchy pane shows the new name.
- The broker symbol in the Topology pane shows the new name.

If you need also to specify a different queue manager name for the renamed broker:

1. In the Topology pane, right click the broker you want to rename.
2. Click **Properties**.
3. In the broker’s properties panel, type the new queue manager name, and correct the description if necessary. The name you specify must be exactly the name specified for this broker’s queue manager on the **mqsicreatebroker** command. Click **Finish**.

If you have no further Topology changes to make:

1. Check in the broker:
   a. In the Topology pane, right click the broker you want to check in.
   b. Click **Check In**.

   The **Key** icon against the broker entry in the Topology tree disappears.
2. Check in the Topology as described in “Checking in the Topology”.

Checking in the Topology

When you have finished making changes to the Topology, you *must* check it in. Until you check in the Topology, no one else is able to make changes to the Topology of this broker domain, nor can you deploy the changes you have made.

You can check in Topology changes only, or all changes.

Checking in Topology changes

To check in the Topology:

1. Right click the root of the Topology tree.

2. Click **Check In** to store the Topology document in the Configuration Manager database.

To confirm that the Topology has been checked in:

- The **Key** icon disappears from the root of the Topology tree in the Domain Hierarchy pane.
- The **New** icon against any new brokers and collectives in the Topology tree disappears, indicating that they have also been checked into the shared configuration. Newly created resources are checked in automatically to ensure that the configuration remains consistent.

Note that any brokers with the **Key** icon against them must be checked in separately; they are not checked in as part of the general Topology check in.

Checking in multiple changes

The **File —> Check In** menu option allows you to check in multiple changes. You can use this instead of checking in individual objects such as the Topology. The options are:

- **File —> Check In —> List**
- **File —> Check In —> All in Current Workspace**
- **File —> Check In —> All (Save to Shared)**

These options are more efficient when you have many different resources checked out. The List option also allows you to check which resources are checked out in your current workspace before you decide which resources to check in.

For more information about check in options, see “Save your workspace to the shared repository” on page 18.

Making changes operational

In checking in resources that are new or that you have altered, you make them visible in the shared configuration. However, the changes you have made have no operational effect until you *deploy* them in the broker domain. For information about deploying resources, see “Chapter 8. Deploying configuration data” on page 93.

Chapter 7. Assigning resources to a broker

This chapter describes how to:

The Assignments view

To display the **Assignments** view, click the **Assignments** tab in the Control Center. Figure 9 shows an example of the **Assignments** view.

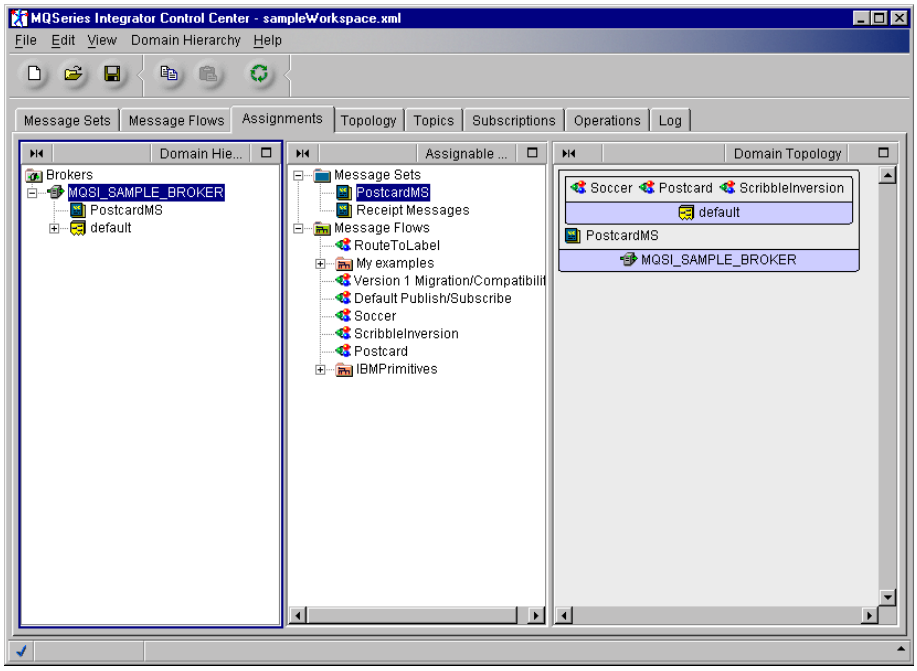


Figure 9. The Assignments view. The left-hand pane, the Domain Hierarchy pane, shows the current hierarchy of brokers, execution groups, message flows, and message sets in your workspace. The center pane, the Assignable Resources pane, shows the message sets and message flows in your workspace. The right-hand pane, the Domain Topology pane, shows in a graphical form the current assignment of execution groups to brokers; of message flows to execution groups; and of message sets to brokers in your workspace.

Double-click on any item in the Domain Topology pane to see its properties.

Creating an execution group

When you create a broker, it has a default execution group. If you want additional execution groups, you must create them explicitly. For more details about why you might want to use multiple execution groups, see *MQSeries Integrator Introduction and Planning*.

To create an execution group:

1. Ensure that the broker to which you want to assign the new execution group is checked out of the shared configuration.

   If the broker entry in the Domain Hierarchy pane of the **Assignments** view has neither the **Key** icon nor the **New** icon against it, right click the broker entry, and click **Check out**.
2. In the Domain Hierarchy pane or Topology pane, right click the entry for the broker.
3. Click **Create —> Execution Group**.

   The **Create a new Execution Group** dialog is displayed.
4. In the **Name** field, type the unique name of the execution group. This must follow the naming rules described in “Naming Control Center resources” on page 11. Click **Finish**.

The new execution group appears:

- Inside the broker symbol in the Domain Topology pane, alongside the symbols for other execution groups assigned to this broker
- Beneath the broker folder in the Domain Hierarchy pane, with a **New** icon against it

Assigning message flows to execution groups

To assign a message flow to an execution group:

1. Ensure that the execution group to which you want to assign the message flow is checked out of the shared configuration.

   If the execution group entry in the Domain Hierarchy pane of the **Assignments** view has neither the **Key** icon nor the **New** icon against it, right click the execution group entry, and click **Check out**.
2. Drag the message flow symbol from the Assignable Resources pane into the symbol of the execution group in the Domain Topology pane. The Assignable Resources pane lists all message flows in your workspace.
3. Check in the execution group.

An alternative approach, and one that is useful when you have a large number of message flows to assign to a single execution group, is as follows:

1. In the Domain Hierarchy pane, right click the entry for the checked-out execution group to which you want to assign a message flow.
2. Click **Add —> Message Flow**.

   The **Add an existing Message Flow** dialog is displayed, showing all message flows in this workspace.
   - To select a single message flow from this list, click the message flow name.
   - To select multiple message flows that appear sequentially in the list, click the first message flow you want, press and hold the Shift key, then click the last message flow you want. This action selects the two message flows you highlighted, plus any that appear between these two in the list.

- To select multiple message flows that do not appear in a sequence in the list, hold down Ctrl and click each message flow you want.

3. When you have selected the message flows you want to assign to the execution group from this list, click **Finish**.

When you assign message flows:

- You cannot add a single message flow more than once to any execution group.
- Subflows are included in the **Add an existing Message Flow** dialog. If you are using a subflow in a higher-level flow, you must assign only the higher-level flow to the execution group: this action includes the subflow.

The message flows you selected appear:

- Inside the execution group symbol in the Domain Topology pane.
- Beneath the execution group entry in the Domain Hierarchy pane.

When you deploy an assigned message flow, each node in the message flow is allocated a label that consists of the name you gave the node qualified by its containing flow. This guarantees the uniqueness of each label within the message flow.

Setting the properties of an assigned message flow

You can change some of the properties of a message flow after you have assigned it to an execution group. To change the properties of a message flow, right click the entry for the message flow under the appropriate execution group in the Domain Hierarchy pane, and click **Properties**. The properties whose values you can change are:

Additional Instances

Specifies the number of threads that the broker should start in order to read messages from the input queue named on the MQInput node of the message flow and process them concurrently. You can have up to 256 threads.

Having additional threads can increase the throughput of a message flow. However, you should consider the impact on message order and set the Order Mode property on the MQInput node (**Advanced** tab) accordingly. You must also ensure that the input queue has been defined with the SHARE attribute to enable multiple threads to read the same queue.

Its default value is 0.

Commit Count

Specifies how many input messages are processed by a message flow before a syncpoint is taken (by issuing an MQCMIT).

This attribute should be used only if the Additional Instances property is set to 0.

The default value of 1 is also the minimum permitted value. Change this attribute if you want to avoid frequent MQCMIT calls when messages are being processed quickly and the lack of an immediate commit can be tolerated by the receiving application.

Use the Commit Interval to ensure that a commit is performed periodically when not enough messages are received to fulfill the Commit Count.

Commit Interval

Specifies a time interval at which a commit is taken when the Commit Count property is greater than 1 (that is, where the message flow is

batching messages) but the number of messages processed has not reached the value of the Commit Count property. It ensures that a commit is performed periodically when not enough messages are received to fulfill the Commit Count.

The time interval is specified in seconds and must be in the range 0 through 60.

This attribute should be used only if the Additional Instances property is set to 0.

Its default value is 0.

Coordinated Transaction

Controls whether the message flow is processed as a global transaction, coordinated by MQSeries. Such a message flow is said to be fully globally-coordinated. The default value is no.

Use coordinated transactions only where you need the message and any database updates performed by the message flow to be processed in a single unit-of-work, using a two-phase commit protocol. This means that both the message is read and the database updates are performed, or neither is done.

If you change this value, you **must** ensure that the broker's queue manager is configured correctly. If you do not set up the queue manager correctly, a message is generated by the broker when a message is received by the message flow to indicate that although the message flow is to be globally coordinated, the queue manager configuration does not support this.

See the *MQSeries Integrator Administration Guide* for information about which databases are supported as participants in a global transaction, and the *MQSeries System Administration* book for how to configure MQSeries and the database managers.

Assigning message sets to brokers

To assign a message set to a broker:

1. Ensure that the broker to which you want to assign the message set is checked out of the shared configuration.

   If the broker entry in the Domain Hierarchy pane of the **Assignments** view has neither the **Key** icon nor the **New** icon against it, right click the broker entry, and click **Check out**.
2. Drag the message set symbol from the Assignable Resources pane into the symbol of the broker (but not into any execution group contained in the broker) in the Domain Topology pane. The Assignable Resources pane lists all message sets in your workspace.

An alternative approach, and one that is useful when you have a large number of message sets to assign to a single broker, is as follows:

1. In the Domain Hierarchy pane, right click the entry for the checked-out broker to which you want to assign a message set.
2. Click **Add —> Message Set**.

   The **Add an existing Message Set** dialog is displayed, showing all message sets in this workspace.
   - To select a single message set from this list, click the message set name.

- To select multiple message sets that appear sequentially in the list, click the first message set you want, press and hold the Shift key, then click the last message set you want. This action selects the two message sets you highlighted, plus any that appear between these two in the list.
- To select multiple message sets that do not appear in a sequence in the list, hold down Ctrl and click each message set you want.

Note that you cannot assign a single message set more than once to any broker.

3. When you have selected the message sets you want to assign to the broker from this list, click **Finish**.

The message sets you selected appear:
- Inside the broker symbol in the Domain Topology pane.
- Beneath the broker symbol in the Domain Hierarchy pane.

Removing resources from a broker

You can remove message sets from the broker to which they have been assigned, you can remove message flows from the execution groups to which they have been assigned, and you can delete execution groups from their owning broker.

Deleting an execution group from a broker

To delete an execution group from a broker:

1. Ensure that the broker from which you want to delete the execution group is checked out of the shared configuration.

   If the broker entry in the Domain Hierarchy pane of the **Assignments** view has neither the **Key** icon nor the **New** icon against it, right click the broker entry, and click **Check out**.
2. Ensure that the execution group you want to delete is *not* checked out.

   If the execution group entry in the Domain Hierarchy pane of the **Assignments** view has the **Key** icon against it, right click the execution group entry and click **Check in**.
3. Right click the execution group entry under the broker in the Domain Hierarchy pane, or right click the execution group symbol in the Domain Topology pane, and click **Delete**.

The execution group and any message flow assignments it contains are deleted:
- From the broker symbol in the Domain Topology pane
- From the relevant broker entry in the Domain Hierarchy pane

The message flows themselves are not deleted or removed from your workspace, and remain in the Assignable Resources pane to be assigned to other execution groups.

Removing a message set from a broker

To remove a message set from a broker:

1. Ensure that the broker from which you want to remove the message set is checked out of the shared configuration.

   If the broker entry in the Domain Hierarchy pane of the **Assignments** view has neither the **Key** icon nor the **New** icon against it, right click the broker entry, and click **Check out**.

2. Right click the message set symbol in the Domain Topology pane, or right click its entry in the Domain Hierarchy pane, and click **Remove**.

The message set assignment disappears from:
- The broker symbol in the Domain Topology pane
- The broker entry in the Domain Hierarchy pane

The message set is not deleted or removed from your workspace, and is still available in the Assignable Resources pane to be assigned to other brokers.

Removing a message flow from an execution group

To remove a message flow from an execution group:

1. Ensure that the execution group from which you want to remove the message flow is checked out of the shared configuration.

   If the execution group entry in the Domain Hierarchy pane of the **Assignments** view has neither the **Key** icon nor the **New** icon against it, right click the execution group entry, and click **Check out**.
2. Right click the message flow symbol inside the execution group symbol in the Domain Topology pane, or right click the message flow entry in the Domain Hierarchy pane, and click **Remove**.

The message flow assignment disappears from:
- The execution group symbol in the Domain Topology pane
- The execution group entry in the Domain Hierarchy pane

The message flow is not deleted or removed from your workspace, and is still available in the Assignable Resources pane to be assigned to other execution groups.

Checking in the assignments

When you have finished assigning resources to a broker, you *must* check in any brokers and execution groups that are checked out. Until you check in brokers and execution groups, no one else is able to make changes to them, nor can you deploy the assignments you have made.

When a newly created broker or execution group is checked in, all related resources are also checked in automatically. For example, when you check in a new broker, its default execution group and the Topology document are also checked in, to ensure consistency of configuration data. MQSeries Integrator does this to prevent you from accidentally excluding related resources that can not easily be recovered. After a new resource has been checked in for the first time, you can check individual resources out, modify them, and check them in individually.

Checking in a broker

To check in a broker using either the Assignments or Topology page:

1. Right click the broker entry in the Domain Hierarchy pane.
2. Click **Check in** to store the broker in the shared configuration.

To confirm that the broker assignments have been checked in, the **Key** icon disappears from the broker entry in the Domain Hierarchy pane.

To check in an execution group using the Assignments page:

1. Right click the execution group entry in the Domain Hierarchy pane.
2. Click **Check in** to store the execution group in the shared configuration.

To confirm that the execution group has been checked in, the **Key** icon disappears from the execution group in the Domain Hierarchy pane.

Checking in multiple changes

The **File —> Check In** menu option allows you to check in multiple changes. You can use this instead of checking in individual objects such as Assignments data. The options are:

- **File —> Check In —> List**
- **File —> Check In —> All in Current Workspace**
- **File —> Check In —> All (Save to Shared)**

These options are more efficient when you have many different resources checked out. The List option also allows you to check which resources are checked out in your current workspace before you decide which resources to check in.

For more information about check in options, see “Save your workspace to the shared repository” on page 18.

Refreshing the Assignments view

You can update the Assignments view with the most recent contents of the configuration repository at any time. Select **View —> Refresh from Shared**. This shows changes to resources that are not checked out: if you have a resource checked out, the version you have in your current workspace is not overwritten with the version from the shared configuration.

Making changes operational

In checking in resources that are new or that you have altered, you make them visible in the shared configuration. However, the changes you have made have no operational effect until you *deploy* them in the broker domain. For information about deploying resources, see “Chapter 8. Deploying configuration data” on page 93.

Chapter 8. Deploying configuration data

The following types of configuration data need to be *deployed* before they can take effect in the broker domain:

Assignments data
Execution groups to brokers; message flows to execution groups; and message sets to brokers.

Topics data
Topics and associated Access Control Lists (ACLs) for the broker domain

Topology data
Broker and collective data for the broker domain

When you request deployment of any type of configuration data, the Configuration Manager copies the relevant configuration data from the shared configuration and transmits it to the relevant brokers. When the deployment is successful, the brokers are able to act in accordance with the newly deployed data.

For more information about the concepts behind the deployment function, please see “Chapter 14. Concepts of deployment” on page 131.

Deleting a broker from the broker domain

When you delete a broker using the Control Center, the broker symbol is no longer visible in the **Assignments** view or the **Topology** view.

- If the broker has never been deployed to, the broker and all execution groups assigned to it are deleted immediately from the shared configuration repository.
- If the broker has been deployed to, its definition remains in the shared deployed configuration after the topology (with the broker deleted) has been checked in, until the updated configuration is deployed.
  - When configuration data of any type is next deployed after the broker reference in the Control Center is deleted, the Configuration Manager sends a configuration data stream requesting deletion of all data of the type relevant to that deployment request (which can be topology, topics, assignments, or all types) to the deleted broker.

    For example, if you request a delta deployment of topics data after having deleted a broker using the Control Center, the Configuration Manager constructs a configuration data stream to delete all topics data deployed to the deleted broker.
  - When all configuration data of all types has been successfully deleted in this way, which might take several deployment requests if the deploys are of different types, the deleted broker is finally removed from both the shared and the deployed configurations.

  You can, of course, ensure early completion of this stage by requesting a complete or delta deployment of all types of data.

  When the broker has been removed from both the shared *and* the deployed configurations, it is removed from the **Operations** view when you next refresh that view.

  You are strongly recommended to perform the actions for deleting a broker that has been deployed to in the following order:

- Delete the broker reference in the Control Center.
- Deploy this change by selecting **File—>Deploy—> Complete Configuration (all types)—>Normal**. You can deploy just the topology data, however it is not recommended as other deploys will be needed as mentioned earlier.
- Delete the physical broker on its local system using the **mqsideletebroker** command.

This order ensures that the deletion messages described above are processed by the broker on its local system before it is deleted and avoids potential inconsistency of deployed data.

- If you subsequently recreate a broker with the same name as a deleted broker, you **must** complete the necessary steps in the recommended order above, followed by these steps:
  - Recreate the physical broker on its local system using the **mqsicreatebroker** command
  - Create a new broker reference in the Control Center.
  - Make any assignments that are needed
  - Deploy the new configuration. As the previous instance of this broker has been completely deleted, you can select either **File—>Deploy—> Complete Configuration (all types)—>Normal**. or **Deploy—>Complete Topology Configuration** or **Deploy—>Delta Topology Configuration** in the Topology view (described in “Deploying delta topology” on page 97 and “Deploying a complete topology” on page 98).

The act of deployment, recreation, and redeployment in the Control Center is required to reset the configuration repository and update internal identifiers for the broker that are generated by the create command. If you do not delete and recreate the broker, and redeploy all data to that broker, the identifiers will not match and check in or deployment will fail.

Deploying delta data of all types

You can deploy delta data of all types from all views on the Control Center.

1. Ensure that the assignments, topics, and topology data you want to deploy has been checked into the shared configuration, as described in“Chapter 7. Assigning resources to a broker” on page 85.
2. Click **File —> Deploy —> Delta configuration (all types)**.

The Configuration Manager compares data of all types for all brokers in the shared configuration with the currently deployed data for all brokers, and deploys only the differences between the two versions.

For information about checking the progress of this deployment request, see “Monitoring progress of deployment” on page 98.

Deploying complete data of all types

You can deploy complete data of all types from all views on the Control Center.

1. Ensure that the assignments, topics, and topology data you want to deploy has been checked into the shared configuration, as described in “Chapter 7. Assigning resources to a broker” on page 85.
2. Click **File —> Deploy —> Complete configuration (all types) —> Normal**.

Deploying complete data of all types

The Configuration Manager creates a request consisting of instructions to delete *all* deployed data of all types, followed by instructions to create a new set of data, based on the shared configuration, and deploys it to the target brokers.

For information about checking the progress of this deployment request, see "Monitoring progress of deployment" on page 98.

Forcing deployment of all data

You can force deployment of complete data of all types from all views on the Control Center.

1. Ensure that the assignments, topics, and topology data you want to deploy has been checked into the shared configuration, as described in "Chapter 7. Assigning resources to a broker" on page 85.
2. Click **File —> Deploy —> Complete configuration (all types) —> Forced**.

The Configuration Manager creates a request consisting of instructions to delete *all* deployed data of all types, followed by instructions to create a new set of data, based on the shared configuration, and deploys it to the target brokers. Any outstanding deployment request, of any type, is overridden by this forced deployment of configuration data.

If you have deleted and recreated a broker, and have not followed the order of actions indicated in "Deleting a broker from the broker domain" on page 93, you might find it necessary to use a forced deployment to reset your broker domain configuration. See "If the broker is not running" on page 99 for another example of where you might want to use a forced deployment.

For information about checking the progress of this deployment request, see "Monitoring progress of deployment" on page 98.

Deploying delta assignments

You must be in the Assignments view to deploy only assignments data.

1. Ensure that the assignments data you want to deploy has been checked into the shared configuration, as described in "Chapter 7. Assigning resources to a broker" on page 85.
2. Select the objects to which you want to deploy the assignments data.

   If you are deploying to all brokers in the broker domain:

   - In the Domain Hierarchy pane of the **Assignments** view, right click the root of the Broker tree. Select **Deploy—>Delta Assignments Configuration**.

   If you are deploying to a single broker:

   - In the Domain Hierarchy pane of the **Assignments** view, right click the entry of the broker to which you want to deploy assignments data. Select **Deploy—>Delta Assignments Configuration**.

     Alternatively, you can right click the broker symbol in the Domain Topology pane and select **Deploy—>Delta Assignments Configuration**.

   If you are deploying to a single execution group:

   - In the Domain Hierarchy pane of the **Assignments** view, right click the execution group to which you want to deploy assignments data. Select **Deploy—>Delta Assignments Configuration**.

Alternatively, you can right click the execution group symbol in the Domain Topology pane and select **Deploy—>Delta Assignments Configuration**.

3. You can also invoke the **Deploy —> Delta Assignments Configuration** action from the Domain Hierarchy menu on the Control Center menu bar.

The Configuration Manager compares assignments data for the target brokers or execution groups in the shared configuration with the currently deployed assignments data for the same brokers, and deploys only the differences between the two versions.

For information about checking the progress of this deployment request, see "Monitoring progress of deployment" on page 98.

Deploying complete assignments

You must be in the Assignments view to deploy only assignments data.

1. Ensure that the assignments data you want to deploy has been checked into the shared configuration, as described in "Chapter 7. Assigning resources to a broker" on page 85.
2. Select the objects to which you want to deploy assignments data.

   If you are deploying to all brokers in the broker domain:

   - In the Domain Hierarchy pane of the **Assignments** view, right click the root of the Broker tree. Select **Deploy—>Complete Assignments Configuration**.

   If you are deploying to a single broker:

   - In the Domain Hierarchy pane of the **Assignments** view, right click the entry of the broker to which you want to deploy assignments data. Select **Deploy—>Complete Assignments Configuration**.

     Alternatively, you can right click the broker symbol in the Domain Topology pane and select **Deploy—>Complete Assignments Configuration**.

   If you are deploying to a single execution group:

   - In the Domain Hierarchy pane of the **Assignments** view, right click the execution group to which you want to deploy assignments data. Select **Deploy—>Complete Assignments Configuration**.

     Alternatively, you can right click the execution group symbol in the Domain Topology pane and select **Deploy—>Complete Assignments Configuration**.
3. You can also invoke the **Deploy —> Complete Assignments Configuration** action from the Domain Hierarchy menu on the Control Center menu bar.

The Configuration Manager creates a request consisting of instructions to delete *all* deployed assignments data, followed by instructions to create a new set of assignments data, based on the shared configuration, and deploys it to the target brokers.

For information about checking the progress of this deployment request, see "Monitoring progress of deployment" on page 98.

Deploying delta topics

You must be in the Topics view to deploy only topics data.

1. Ensure that the topics data you want to deploy has been checked into the shared configuration, as described in “Checking in topics data” on page 104.
2. In the Topics pane of the **Topics** view, right click Topics.

   Select **Deploy —> Delta Topics Configuration**.
3. You can also invoke the **Deploy —> Delta Topics Configuration** action from the **Topics** menu on the Control Center menu bar.

The Configuration Manager compares topics data for all brokers in the shared configuration with the currently deployed topics data for all brokers, and deploys only the differences between the two versions.

For information about checking the progress of this deployment request, see “Monitoring progress of deployment” on page 98.

Deploying complete topics

You must be in the Topics view to deploy only topics data.

1. Ensure that the topics data you want to deploy has been checked into the shared configuration, as described in “Checking in topics data” on page 104.
2. In the Topics pane of the **Topics** view, right click Topics.

   Select **Deploy —> Complete Topics Configuration**.
3. You can also invoke the **Deploy —> Complete Topics Configuration** action from the **Topics** menu on the Control Center menu bar.

The Configuration Manager creates a request consisting of instructions to delete *all* deployed topics data, followed by instructions to create a new set of topics data, based on the shared configuration, and deploys it to the target brokers.

For information about checking the progress of this deployment request, see “Monitoring progress of deployment” on page 98.

Deploying delta topology

You must be in the Topology view to deploy only topology data.

1. Ensure that the topology data you want to deploy has been checked into the shared configuration, as described in “Checking in the Topology” on page 82.
2. Choose one of the following options:
   - In the Domain Hierarchy pane of the **Topology** view, right click the root of the Topology tree. Select **Deploy —> Delta Topology Configuration**.

     You can also select **Deploy —> Delta Topology Configuration**. from the **Domain Hierarchy** menu on the Control Center menu bar.
   - In the Topology pane of the **Topology** view, right click the background and select **Deploy —> Delta Topology Configuration**.

     You can also select **Deploy —> Delta Topology Configuration** from the **Topology** menu on the Control Center menu bar.

The Configuration Manager compares topology data for all brokers in the shared configuration with the currently deployed topology data for all brokers, and deploys only the differences between the two versions.

For information about checking the progress of this deployment request, see “Monitoring progress of deployment” on page 98.

Deploying a complete topology

You must be in the Topology view to deploy only topology data.

1. Ensure that the topology data you want to deploy has been checked into the shared configuration, as described in “Checking in the Topology” on page 82.
2. Choose one of the following options:
   - In the Domain Hierarchy pane of the **Topology** view, right click the root of the Topology tree. Select **Deploy —> Complete Topology Configuration**.

     You can also select **Deploy —> Complete Topology Configuration**. from the **Domain Hierarchy** menu on the Control Center menu bar.
   - In the Topology pane of the **Topology** view, right click the background and select **Deploy —> Complete Topology Configuration**.

     You can also select **Deploy —> Complete Topology Configuration** from the **Topology** menu on the Control Center menu bar.

The Configuration Manager creates a request consisting of instructions to delete *all* deployed topology data, followed by instructions to create a new set of topology data, based on the shared configuration, and deploys it to the target brokers.

For information about checking the progress of this deployment request, see “Monitoring progress of deployment”.

Monitoring progress of deployment

You can find out whether stage two of a deployment has succeeded by refreshing the **Log** view: click the green refresh button on the menu bar, or select **View —> Refresh**. It might take a while for the response to arrive. The refreshed **Log** view displays a group of messages for each broker to which configuration data has been deployed. Typical messages are:

Message	Meaning
BIP2056	Indicates that a deployment was completely successful for the broker.
BIP2086	Indicates that a deployment was partially successful for the broker.
BIP2087	Indicates that a deployment was completely unsuccessful for the broker.

If a deployment fails completely or partially succeeds, and message BIP4046 also appears in the **Log** view, Topics or Topology data was not processed. In this case, the broker in question is out of step with the rest of the broker domain, and so you *must* correct the problem that caused the failure and deploy again to restore consistency of data throughout the broker domain. This might occur, for example, if you have deleted and recreated a broker. See “Deleting a broker from the broker domain” on page 93 for further details.

Refresh the **Operations** view of the Control Center to display the status of each broker after the deployment.

If deployment is in doubt

It is possible for the deployment of an execution group to time out while it is being processed by the target broker. This effectively leaves the status of the execution group in doubt. This status is shown in the **Operations** view by the appearance of

a yellow question mark over the traffic light status icon. A message in the **Log** view confirms the problem. The in-doubt status of the execution group can be resolved only by a subsequent deployment of *all* assignments data. (Note that a subsequent delta deployment is automatically converted to a complete deployment if any execution group is in the in-doubt state).

If the broker is not running

If a broker is not running when a deployment takes place, or an MQSeries queue manager on the route to the broker is not running, the deployment message is not processed immediately. Note, however, that the deployment message does not expire, so it will be processed eventually. You cannot perform a complete or delta deployment to a broker when a deployment of any type is outstanding to that broker: an attempt to do so returns an error message in a Control Center dialog box. Stage two of the deploy must complete before a further deploy is allowed, unless a forced deployment is requested.

Chapter 9. Setting up publish/subscribe access control lists

This chapter describes how to create a new publish/subscribe topic, and how to update access control lists (ACLs). ACLs allow you to restrict user permission to publish messages, subscribe to topics, and request persistent delivery of messages.

For more information about the Subscriptions view, monitoring and deleting subscriptions please see “Subscriptions view” on page 113.

The Topics view

To display the **Topics** view, click the **Topics** tab in the Control Center.

In the **Topics** view, you can create the topics under which messages can be published. In addition, you can give users or groups permission to publish messages, or to subscribe to messages published under these topics. You can also deny users or groups these access rights. You would do this to ensure that privileged information was not being viewed by unauthorized users or groups, for example.

The information in the **Topics** view can be viewed in two ways:

- The hierarchy of topics is shown in the **Topic/Users** view, where the Access Control List (ACL) for the selected topic is shown.
- The list of users and groups is shown in the **User/Topics** view, and the access to each topic is shown for the selected user or group.

In addition to the **Topics** view, you can use the **Subscriptions** view to see currently registered subscriptions if you are a member ofMQSeries Integrator group **mqbrops**.

Creating topics

Topics is a special topic that cannot be deleted or renamed. It always has the Public Group in its ACL. You create new topics beneath Topics (always displayed in the Topics pane) or beneath any topic already defined. Any topic can have any number of children, and each of these can have different ACL settings.

To create a new topic:

1. Click the **Topic/Users** button in the **Topics** view.
2. Ensure that the topic under which you want to create a new one, which can be Topics or any topic already defined, is checked out. If it is not checked out, right click the topic and click **Check Out**.
3. Right click the parent topic and click **Create —> Topic**.

   The **Create a new Topic** dialog is displayed.
4. In the **Create a new Topic** dialog, type the name of the topic in the Name field.
5. Select the users and groups that are to have explicit access defined for this topic. Note that this list contains users and groups (also known as principals) only if you have a User Name Server installed and running, and the Configuration Manager is configured to communicate with it.

   If you do not select users or groups from the list on this dialog, access for this topic defaults to the ACL setting for Public Group for the topic root. You can update access to this topic for explicit users at a later time, if you do not do so now.

   To specify explicit users or groups access now, expand the Groups and Users folders and select the users or groups:

   - To select a single user or group from the list, click the user or group name.
   - To select multiple users or groups that appear sequentially in the list, click the first user or group you want, press and hold the Shift key, then click the last user or group you want. This action selects the two users or groups you highlighted, plus any that appear between these two in the list.
   - To select multiple users or groups that do not appear in a sequence in the list, hold down Ctrl and click each principal you want.
6. Select the required access setting for this topic. The values that you set apply to all users and groups that you selected in the create topic dialog (step 4).
   - For the Publish field, select one of

     Allow — Publications are allowed.

     Deny — Publications are not permitted.

     Inherit — Permission to publish is inherited.
   - For the Subscribe field, select one of

     Allow — Subscriptions are allowed.

     Deny — Subscriptions are not permitted

     Inherit — Permission to subscribe is inherited.
   - For the Persistent field, select one of

     Yes — Persistent delivery of messages is allowed.

     No — Persistent delivery of messages is not allowed.

     Inherit — Permission to request persistent delivery of messages is inherited.

7. Click **Finish**.

The new topic appears beneath its parent topic.

After you create a topic, you can add more users or groups to the ACL using the Properties dialog as described “Adding a principal to an ACL”.

If you do not select any users or groups when you create the topic, the ACL is empty, and the Topics Access Control List pane is left blank. In this case, each user or group inherits the same access to this topic as it has to the parent topic.

If you have selected users or groups, they appear in the Topic Access Control List pane. Beside the users or groups, you see the permissions they have to publish messages, subscribe to messages, and request persistent delivery of messages. You can change these permissions by selecting them. A drop down list is shown, allowing you to select a different permission.

Renaming, copying, and deleting topics

Topics can be renamed, copied, or deleted by right clicking the appropriate topic and selecting the desired action from the pop-up menu. When you copy a topic, a sibling topic with a unique name is created.

Adding a principal to an ACL

To add a principal [1] to an ACL:

1. In the **Topics** view, click the **Topic/Users** button.
2. Ensure that the topic for which you would like to edit the ACL is checked out. If it is not checked out, right click the topic and click **Check Out**.
3. Right click the topic and click **Properties**.
4. Expand the Groups or Users folders in the Available Principals.

   You can add principals that are not yet listed in the ACL; principals that are already in the ACL are not shown. You can grant permissions to a principal, or revoke permissions for a principal. You can specify that the principal inherit the same level of access to a permission as it has to the parent topic. Setting the access level of a principal in the ACL of the Topics to Inherit is not allowed, since the Topics do not have a parent topic. Each principal can be assigned the following permissions:

   Publish
   Permits or denies the principal permission to publish messages on this topic.

   Subscribe
   Permits or denies the principal permission to subscribe to messages on this topic.

   Persistent
   Permits or denies the principal permission to request persistent delivery of a publication when the principal subscribes to the topic.

If a user subscribes to a topic, and the user requests persistent delivery of the messages, the user must be granted permission both to subscribe to that topic and

1. A principal is a user or a group.

to request persistent delivery of messages for that topic. If the user does not request persistent delivery, only permission to subscribe to that topic is required.

Permission for persistent delivery does not affect the publishing of messages. You need only to be granted publish permissions to be able to publish messages on a topic.

To remove an entry from an ACL, in the **Topic/Users**, right click the entry and click **Remove**.

Resolving permissions

Many factors play a part in determining whether the user has permission to publish messages on a topic, subscribe to messages under a topic, and to request persistent delivery of messages being subscribed to. The user can be explicitly listed in the topic's ACL. Groups to which the user belongs can also be listed, and their permissions may differ from each other and with the user's ACL entry. Users can also inherit permissions from parent topics. Determining whether the user has a permission might not always be straightforward.

For a complete description of how permissions are resolved, see the *MQSeries Integrator Introduction and Planning*.

Checking in topics data

To check in topics data:

1. Right click the topic entry in the **Topics** view.
2. Click **Check in** to store the topics data in the shared configuration.

To confirm that the topics data has been checked in, the **New** icon or the **Key** icon disappears from the topic entry.

When you check in a new topic, its parent is also checked in. When you check in a parent topic, all new child topics are also checked in.

Checking in multiple changes

The **File —> Check In** menu option allows you to check in multiple changes that you have made in this or any other view. You can use this instead of checking in individual objects in the Topics view. The options are:

- **File —> Check In —> List**
- **File —> Check In —> All in Current Workspace**
- **File —> Check In —> All (Save to Shared)**

These options are more efficient when you have many different resources checked out. The List option also allows you to check which resources are checked out in your current workspace before you decide which resources to check in.

For more information about check in options, see “Save your workspace to the shared repository” on page 18.

Making changes operational

When you check in resources that are new, or that you have altered, you make them visible in the shared configuration. However, the changes you have made have no operational effect until you *deploy* them in the broker domain. For information about deploying resources, see “Chapter 8. Deploying configuration data” on page 93.

Chapter 10. Running the broker domain

This chapter describes:

The Operations view

To display the **Operations** view, click the **Operations** tab in the Control Center. Figure 10 shows an example of the **Operations** view.

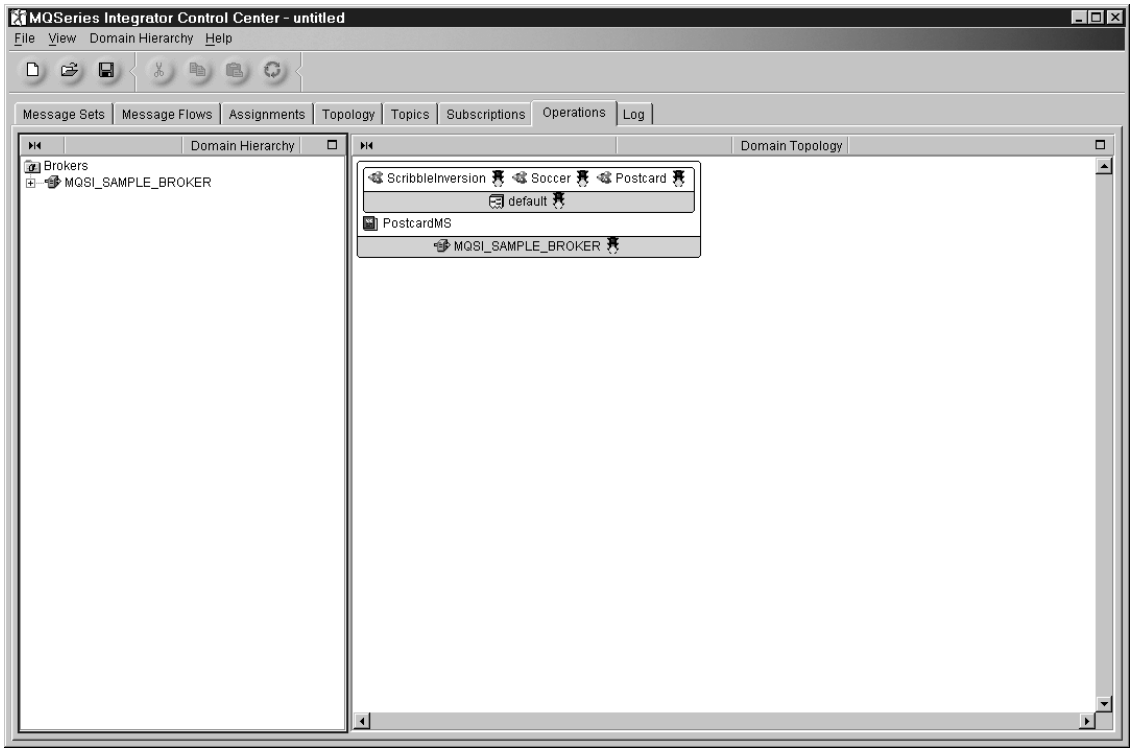


Figure 10. The Operations view. The left-hand pane, the Domain Hierarchy pane, shows a tree view of the brokers in your broker domain. The execution groups and message sets assigned to a broker are displayed when you expand the broker. The message flows assigned to an execution group are displayed when you expand the execution group. The right-hand pane, the Domain Topology pane, contains an arrangement of graphical symbols that represent the current broker domain. Execution groups and message sets appear inside the brokers to which they have been assigned. Message flows appear inside the execution groups to which they have been assigned. The brokers shown in the Operations view are those to which configuration data has been deployed.

Double-click on any item in Domain Topology pane to see its properties.

Monitoring the operational state of the broker domain

When a deploy operation has taken place successfully, the target brokers automatically start to run the message flows, or to provide the publish/subscribe capability, associated with the deployment request. Using the Control Center, you can monitor the status of the brokers and the message flows they are running, and can perform a limited number of actions to control the operation of the brokers. For example, you can start and stop message flows.

To display a snapshot of the current status of the broker domain, you must refresh the view. You can either click the green circular refresh icon below the menu bar, or select **View —> Refresh**. This causes the Configuration Manager to update the information displayed in the Domain Topology pane from its deployed configuration.

Any resource shown in the Domain Topology pane of the **Operations** view can be in one of three states:

Started	Indicated by a green traffic light next to the resource.
Stopped	Indicated by a red traffic light next to the resource.
Unknown	Indicated by a yellow question mark next to the resource.

If you initiate a complete deployment, the operational state of all resources is reset. Therefore, if, for example, you have stopped an individual message flow, or you have started user trace, you will have to reissue this requests.

Starting message flows

You can start:
- All message flows in all execution groups assigned to a specified broker
- All message flows in a specified execution group
- A single message flow

Starting all message flows for a broker

To start all message flows in all execution groups assigned to a specified broker:

1. Right click the broker symbol in the Domain Topology pane or the broker entry in the Domain Hierarchy pane. (Alternatively, you can highlight the broker in either pane, then click the Domain Hierarchy or Domain Topology menu in the menu bar.)
2. Click **Start Message Flows**.

   The Configuration Manager sends a configuration message to the broker requesting that all message flows be started.
3. To monitor the outcome of this request, after a suitable delay:
   a. Refresh the **Operations** view, as described in “Monitoring the operational state of the broker domain” on page 108. If the request was successful, all message flows within the broker have a green status light against them.
   b. Refresh the **Log** view. Any messages returned by the broker in response to this request are displayed here.

Starting all message flows within an execution group

To start all message flows in an execution group:

1. Right click the execution group symbol in the Domain Topology pane or the execution group entry in the Domain Hierarchy pane. (Alternatively, you can highlight the execution group in either pane, then click the Domain Hierarchy or Domain Topology menu in the menu bar.)
2. Click **Start Message Flows**.

   The Configuration Manager sends a configuration message to the broker requesting that all message flows within the specified execution group be started.
3. To monitor the outcome of this request, after a suitable delay:
   a. Refresh the **Operations** view, as described in “Monitoring the operational state of the broker domain” on page 108. If the request was successful, all message flows within the execution group have a green status light against them.
   b. Refresh the **Log** view. Any messages returned by the broker in response to this request are displayed here.

Starting a single message flow

To start a single message flow:

1. Right click the message flow symbol in the Domain Topology pane or the message flow entry in the Domain Hierarchy pane. (Alternatively, you can highlight the message flow in either pane, then click the Domain Hierarchy or Domain Topology menu in the menu bar.)
2. Click **Start**.

   The Configuration Manager sends a configuration message to the broker requesting that the specified message flow be started.

3. To monitor the outcome of this request, after a suitable delay:
   a. Refresh the **Operations** view, as described in “Monitoring the operational state of the broker domain” on page 108. If the request was successful, the message flow has a green status light against it.
   b. Refresh the **Log** view. Any messages returned by the broker in response to this request are displayed here.

Stopping message flows

You can stop:
- All message flows in all execution groups assigned to a specified broker
- All message flows in a specified execution group
- A single message flow

Stopping all message flows for a broker

To stop all message flows in all execution groups assigned to a specified broker:

1. Right click the broker symbol in the Domain Topology pane or the broker entry in the Domain Hierarchy pane. (Alternatively, you can highlight the broker in either pane, then click the Domain Hierarchy or Domain Topology menu in the menu bar.)
2. Click **Stop Message Flows**.

   The Configuration Manager sends a configuration message to the broker requesting that all message flows be stopped.
3. To monitor the outcome of this request, after a suitable delay:
   a. Refresh the **Operations** view, as described in “Monitoring the operational state of the broker domain” on page 108. If the request was successful, all message flows within the broker have a red status light against them.
   b. Refresh the **Log** view. Any messages returned by the broker in response to this request are displayed here.

Stopping all message flows within an execution group

To stop all message flows in an execution group:

1. Right click the execution group symbol in the Domain Topology pane or the execution group entry in the Domain Hierarchy pane. (Alternatively, you can highlight the execution group in either pane, then click the Domain Hierarchy or Domain Topology menu in the menu bar.)
2. Click **Stop Message Flows**.

   The Configuration Manager sends a configuration message to the broker requesting that all message flows within the specified execution group be stopped.
3. To monitor the outcome of this request, after a suitable delay:
   a. Refresh the **Operations** view, as described in “Monitoring the operational state of the broker domain” on page 108. If the request was successful, all message flows within the execution group have a red status light against them.
   b. Refresh the **Log** view. Any messages returned by the broker in response to this request are displayed here.

Stopping a single message flow

To stop a single message flow:

1. Right click the message flow symbol in the Domain Topology pane or the message flow entry in the Domain Hierarchy pane. (Alternatively, you can highlight the message flow in either pane, then click the Domain Hierarchy or Domain Topology menu in the menu bar.)
2. Click **Stop**.

   The Configuration Manager sends a configuration message to the broker requesting that the specified message flow be stopped.
3. To monitor the outcome of this request, after a suitable delay:
   a. Refresh the **Operations** view, as described in "Monitoring the operational state of the broker domain" on page 108. If the request was successful, the message flow has a red status light against it.
   b. Refresh the **Log** view. Any messages returned by the broker in response to this request are displayed here.

Starting user tracing

You can start user tracing from the Operations view:
- For all message flows in a specified execution group
- For a single message flow

The user tracing function of MQSeries Integrator is described in the *MQSeries Integrator Administration Guide*. Refer to this for information about the levels of tracing that can be started (normal and debug), and for information on how to format and read the output.

Starting user tracing for an execution group

To start user tracing of all message flows in an execution group:

1. Right click the execution group symbol in the Domain Topology pane or the execution group entry in the Domain Hierarchy pane. (Alternatively, you can highlight the execution group in either pane, then click the Domain Hierarchy or Domain Topology menu in the menu bar.)
2. Click **User Trace —> Normal** or **User Trace —> Debug**.

   The Configuration Manager sends a configuration message to the broker requesting that user tracing be started for all message flows within the specified execution group.
3. To monitor the outcome of this request, after a suitable delay:
   a. Refresh the **Operations** view, as described in "Monitoring the operational state of the broker domain" on page 108. If the request was successful, the execution group has an icon against it indicating that user tracing is active.
   b. Refresh the **Log** view. Any messages returned by the broker in response to this request are displayed here.

Starting user tracing for a single message flow

To start user tracing for a single message flow:

1. Right click the message flow symbol in the Domain Topology pane or the message flow entry in the Domain Hierarchy pane. (Alternatively, you can highlight the message flow in either pane, then click the Domain Hierarchy or Domain Topology menu in the menu bar.)
2. Click **User Trace —> Normal** or **User Trace —> Debug**.

   The Configuration Manager sends a configuration message to the broker requesting that user tracing be started for the specified message flow.

3. To monitor the outcome of this request, after a suitable delay:
   a. Refresh the **Operations** view, as described in “Monitoring the operational state of the broker domain” on page 108. If the request was successful, the message flow has an icon against it indicating that user tracing is active.
   b. Refresh the **Log** view. Any messages returned by the broker in response to this request are displayed here.

Stopping user tracing

The user tracing function of MQSeries Integrator is described in the *MQSeries Integrator Administration Guide*. You can stop user tracing:

- For all message flows in a specified execution group
- For a single message flow

Stopping user tracing for an execution group

To stop user tracing of all message flows in an execution group:

1. Right click the execution group symbol in the Domain Topology pane or the execution group entry in the Domain Hierarchy pane. (Alternatively, you can highlight the execution group in either pane, then click the Domain Hierarchy or Domain Topology menu in the menu bar.)
2. Click **User Trace —> None**.

   The Configuration Manager sends a configuration message to the broker requesting that user tracing be stopped for all message flows within the specified execution group.
3. To monitor the outcome of this request, after a suitable delay:
   a. Refresh the **Operations** view, as described in “Monitoring the operational state of the broker domain” on page 108. If the request was successful, any user tracing icon against the execution group has disappeared.
   b. Refresh the **Log** view. Any messages returned by the broker in response to this request are displayed here.

Stopping user tracing for a single message flow

To stop user tracing for a single message flow:

1. Right click the message flow symbol in the Domain Topology pane or the message flow entry in the Domain Hierarchy pane. (Alternatively, you can highlight the message flow in either pane, then click the Domain Hierarchy or Domain Topology menu in the menu bar.)
2. Click **User Trace —> None**.

   The Configuration Manager sends a configuration message to the broker requesting that user tracing be stopped for the specified message flow.
3. To monitor the outcome of this request, after a suitable delay:
   a. Refresh the **Operations** view, as described in “Monitoring the operational state of the broker domain” on page 108. If the request was successful, any user tracing icon against the message flow has disappeared.
   b. Refresh the **Log** view. Any messages returned by the broker in response to this request are displayed here.

Subscriptions view

You use the **Subscriptions** view to monitor subscriptions to topics taken out by the applications running in your broker domain. Figure 11 shows an example of the **Subscriptions** view.

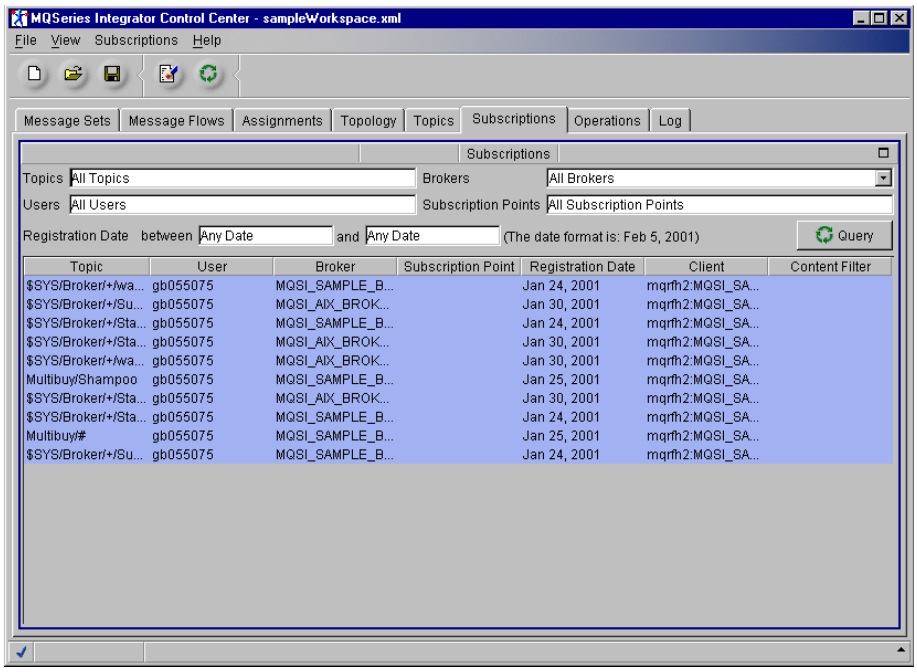


Figure 11. The Subscriptions view. Subscriptions owned by the brokers in this broker domain are shown in this view in a tabular form. Each subscription occupies one row in the table. For each subscription, the Topic, User, Broker, Subscription Point, Registration Date, Client, and Content Filter are displayed. Fields at the top of the view support filtering of information. The entries in the list of subscriptions are not timestamped, and are not ordered.

Filtering information in the Subscriptions view

Within any broker domain there can be many hundreds of active subscriptions. You are unlikely to want to view information relevant to all of these subscriptions at any one time. Therefore, the **Subscriptions** view allows you to select the information you are interested in by specifying a filter. You can filter the information displayed in the **Subscriptions** view by specifying any combination of:

- Brokers
- Topics
- Users
- Registration date
- Subscription points

For example, you can restrict the information displayed to particular topics within a single broker.

To filter the information by broker:

1. Click the **Brokers** drop-down list and click the broker name.
2. Refresh the **Subscriptions** view by clicking **Query**, clicking the green refresh icon below the menu bar, or selecting **View —> Refresh**.

The **Subscriptions** view is refreshed to display information for the selected broker.

To filter information by any other value, simply enter data in the appropriate field in the view. For example, to filter by Topic, enter the topic name in the **Topics** field, and refresh the **Subscriptions** view as described above. The wildcard characters (% which represents any number of characters, and _ (underscore) which represents one character), can be used to represent characters in the topic, user, and subscription point values.

To clear all data from the table, click the clear table icon next to the refresh icon on the menu bar. This action does not delete subscriptions; it simply clears the data from the **Subscriptions** view.

Refreshing the Subscriptions view

The **Subscriptions** view displays a snapshot of all current subscriptions in the broker domain, filtered by the current filter. The Configuration Manager updates its record of the deployed configuration whenever a subscription is created, changed, deleted, or expires. However, the **Subscriptions** view is not updated automatically to reflect these changes. You have to request that the **Subscriptions** view is refreshed by clicking **Query**, or by clicking the green refresh icon on the menu bar, or by selecting **View —> Refresh**.

Deleting subscriptions

To delete (deregister) a subscription from the deployed configuration:

1. In the **Subscriptions** view, select the subscriptions that you want to delete:
   a. To select a single subscription, click the row pertaining to that subscription.
   b. To select multiple rows that appear in a sequence in the table, click the first row you want to delete, press and hold the Shift key, then click the last row you want. This action selects the two rows you highlighted, plus any that appear between these two in the table.
   c. To select multiple rows that do not appear in a sequence in the table, hold down Ctrl and click each row you want.
2. From the **Subscriptions** menu in the menu bar, click **Delete**.
3. To monitor the outcome of this request, after a suitable delay:
   a. Refresh the **Subscriptions** view, as described in “Refreshing the Subscriptions view”. If the subscription has been successfully deleted, its entry is no longer included in the **Subscriptions** view.
   b. Refresh the **Log** view. Any messages returned by the broker in response to this deletion request are displayed here.

   Note that some subscriptions (specifically those used internally, start with `$SYS` and `$ISYS` and are used by the broker and the Configuration Manager) cannot be deleted. Any request to delete such a subscription fails.

Log view

To display the contents of the **Log** view, click the **Log** tab in the Control Center. Figure 12 on page 115 shows an example of the **Log** view.

The Log view is accessible unless you suppress it using the options on the Control Center Preferences dialog (see “Setting user roles” on page 13). The Log view shows messages associated with your user ID, and those that have no associated user ID.

When you change to the Log view, you must refresh the view to see the latest information available. You can request a refresh using the refresh button on the menu bar, the **View** menu, or the pop-up menu within the message pane.

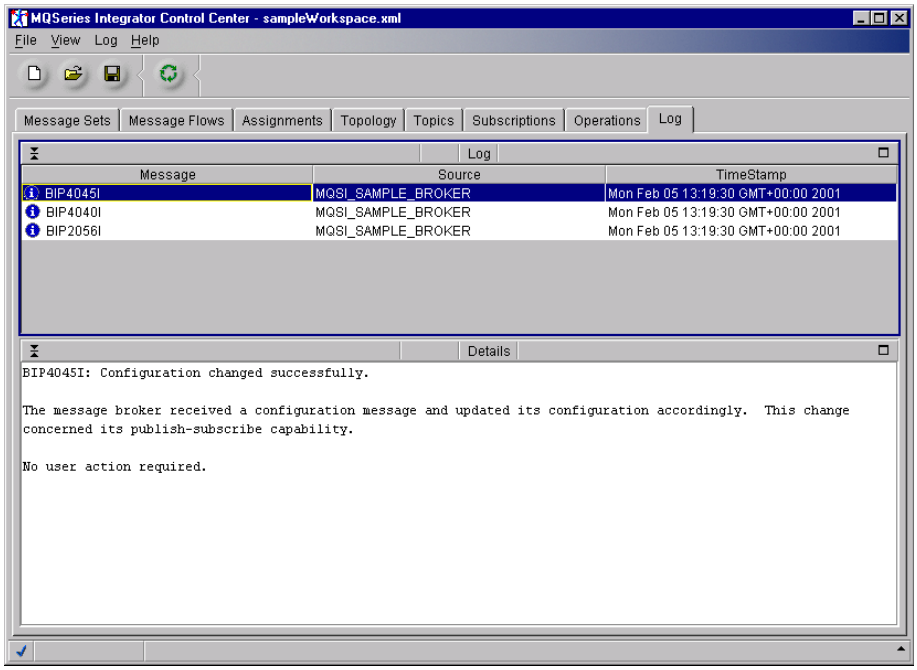


Figure 12. The Log view. The Log view displays messages returned to you by the Configuration Manager in response to requests that update the broker domain configuration. It also displays messages relating to deployment requests and to requests to delete subscriptions.

You can perform the following tasks from the **Log** view or from **Log** on the toolbar, by right-clicking on messages within the **Log** pane to display the pop-up menu:

- **Save Log As**, which saves the Log view in a file

  When you have saved the log to a filename and location of your choice, the saved records remain in the view. They are not removed unless you select the **Clear Log** action. You can view the saved log file, and print its contents if you want.

  The default location for this file is in the `<mqsi_root>Tool\working` directory.

  You can use a text editor to open and work with a saved log file: you cannot open it in the Control Center Log view.

- **Clear Log**, which removes messages

  This menu option implements two actions:

  1. All messages are removed from the Log view.
  2. All messages specific to your user ID are removed from the Configuration Manager's database.

  If you select this option, you cannot retrieve any of the messages cleared from the Log. You are recommended to save the log if you want to preserve some or all of its content.

  If you do not clear messages from the Configuration Manager's database using the Clear function, they are automatically cleared after 72 hours to ensure that the database is not filled.

- Use the **Save Log As** action if you want to refer to an old log in the future. The **Save Log As** action prompts you for the name of a file into which the current Log view contents are saved. The default location for this file is the `Tool` directory on your workstation. Note that the log is saved separately to the workspace.
- Use the **Filter** log to limit the number of messages displayed on your Log View. You can filter on:
  - Message severity. For example, Error messages
  - Message source. For example, a deployed broker

  When working with the **Filter** on you can:

  - **Refresh**, which adds any new messages to the Log view

    The **Refresh** action does not remove or overwrite existing messages in the Log view. Therefore if you frequently refresh without clearing or saving the Log, the messages will build up in the local view and within the Configuration Manager's database.

    When refresh is run, the contents of the view become unfiltered and if required, the **Filter** must be applied again.
  - **Clear** removes all log entries that were displayed by the most recent **Refresh** action including all those not currently in the filtered view.

Problem determination

If an error occurs while you are performing a Control Center operation, the Control Center displays a dialog box containing an MQSeries Integrator V2.0.2 message. The message can originate from either the Control Center itself or from the Configuration Manager. The message should explain any corrective action you can take.

Any errors that occur:

- During the second phase of a deploy operation

or

- From starting or stopping message flows

or

- From starting or stopping user tracing

or

- From deleting subscriptions

are displayed as MQSeries Integrator messages in the **Log** view. These messages originate from the broker.

You might also find it helpful to refer to additional information provided in SupportPac MHI1. This SupportPac provides latest problem determination information in a useful question-and-answer format. You can find this SupportPac at:

`http://www.ibm.com/software/mqseries`

Controlling service traces

The Control Center can be traced by invoking it with a special command, **mqsilcc**, which is described in the *MQSeries Integrator Administration Guide*. You are recommended to use service traces only when you receive an error message that instructs you to start service trace, or when directed to do so by your IBM Support Center.

Chapter 11. Debugging message flows

The debugger is a problem determination tool that can be readily applied to complex message flows. For example, it can track single messages through a message flow one step at a time. Alternatively, by putting a breakpoint on one filter output, you can trap one 'rogue' message out of hundreds of 'normal' messages thus revealing an unexpected condition.

Note: A breakpoint is a position in a message flow. When using the debugger you can set breakpoints in a message flow to review the properties of any node at that specific point in the flow.

In practice, the debugger can handle a wide variety of possible bugs. Here are some examples:

- Nodes may be 'wired-up' incorrectly, for example outputs connected to incorrect inputs
- Filter nodes, which are the message flow version of a conditional branch, may have incorrect conditions
- Compute nodes may have incorrect logic
- Database nodes may make incorrect entries into their target databases
- Messages generated by applications using the flow may be incorrect, or may have contents that do not match the expectations of the message flow
- Programs may have 'feedback' loops that result in unintended infinite loops
- User programmed plug-ins may have errors in them, or may fail to be reentrant, which means they fail to allow concurrent use by two or more tasks

Authorization

The debugger works on assigned message flows. To assign message flows, you need to be a member of the **mqbrasgn** security group and have a Control Center role of **Message flow and message set assigner**. If you want to run the debugger and someone else has assigned the flows, you have to be a member of **mqbrdevt** and **mqbrops** and have the Control Center role of **Message flow and message set developer**.

Debugger View

In order to debug a flow, you must first open the debugger screen. This is accessed from the Message Flows view by pressing the **Debugger** button. You are presented with the basic debugger screen shown in Figure 13 on page 120.

Debugger View

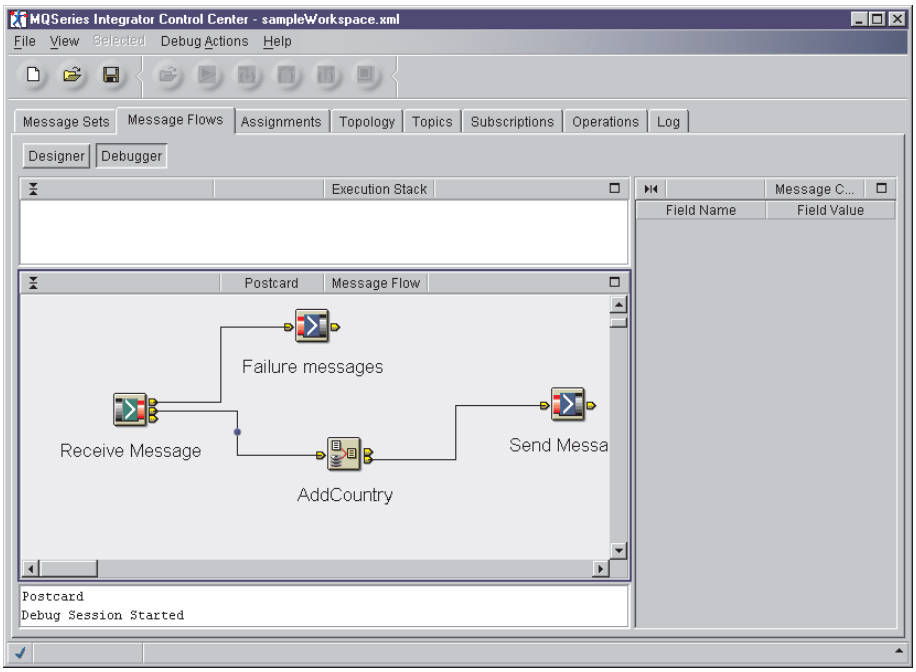


Figure 13. The Debugger screen

You may perform a number of debugging tasks from this screen.

From the **Debug Actions** menu (and corresponding toolbar buttons where appropriate) you can:

- Open and close an assigned message flow using a smart guide
- View the properties of a message flow node
- Start debugging it
- Stop debugging
- Set debug options:
  - Set communications options for the debug nodes
  - Set debug tracing requirements (these are *not* the same as normal trace settings)

On the **Message Flow** pane you can set and remove breakpoints.

The **Stack** pane displays the message execution stack when the debugger reaches a break point.

From the **Message** pane you can view and edit a message as it progresses through the flow.

Set debugger options

Use **Debug Actions**->**Settings**, to bring up a SmartGuide. Default values are set for all options (see below). You only need to use the SmartGuide if you need to change a default value. The SmartGuide has three panes which allow you to set the following:

- The data and error ports the Control Center uses to connect to the debug plug-in
- Tracing level
- The file to use for tracing, if enabled

You can view the debugger settings at any time, but you can not reset them while a message flow is being debugged.

Select ports used to connect to the debug plug-in

Select the communication ports to be used to make connections between the debug plug-in and the Control Center. Separate ports are used for normal message data and for unexpected errors. Default port numbers are 5000 for data and 3500 for errors

It is not necessary to change the default port numbers if you only want to run a debug session in a single Control Center on a particular Windows NT workstation and as long as they do not conflict with port numbers used by other applications. If you wish to run debug sessions in multiple Control Centers on the same Windows NT workstation at the same time, you must adjust the port numbers on the options panes for each of those Control Centers so that they are not the same.

Select debug trace level

Because the debug nodes do not have access to the normal trace log. The debugger maintains its own trace levels. You can choose between 'None', 'Normal' and 'Debug'.

The default level is 'None'.

Select file to use for tracing

If debug tracing is enabled, data is written to the trace file you have specified. The format used in writing to the trace file is XML, so that it can be read using the standard command for formatting the trace file (**mqsiformatlog** - see *MQSeries Integrator Administration Guide*.

Debugging a message flow

1. Check that the message flow you want to debug is not locked (checked out) to another user. You cannot debug a message flow that is currently checked out by another user
2. Check that the message flow you want to debug is assigned to an execution group
3. Open the Debugger. You do this by pressing the **Debugger** button from the **Message Flows** view.

   The screen is divided into four sections. These are initially blank but display information when you start debugging:

   - The **Execution Stack** pane shows the name of the flow being debugged. If you are working with a message flow that contains sub flows, you can see the hierarchy of flows in this pane
   - The **Message Content** pane shows the message in the message flow at the point at which a breakpoint is triggered. The pane shows the field name and field value of any headers as well as the body of the message. You can edit the message in this pane as it progresses from breakpoint to breakpoint
   - The **Message Flow** pane shows the name of the message flow that was opened in this debug session, displays the message flow, and the progress of the debugger through the breakpoints. You set and remove breakpoints in this pane. You can view properties of any node in the message flow but you cannot change them from here

- The **Information** pane shows information messages from the debugger, that the debug session has started or ended or that a breakpoint is encountered

4. Select **Open Message Flow** on the **Debug Action** menu (or press the corresponding toolbar button) and then use the SmartGuide to choose an assigned message flow

   Note: If you open a message flow but have not started debugging it, you can close it by selecting

   ```
   Debug Actions->Close Message Flow
   ```

   or using the equivalent toolbar icon.

5. In the **Message Flow** pane , right-click on a connection and select **Break** or right-click on a message processing node and select **Break Before** or **Break After**. (You cannot set **Break Before** on an input node or **Break After** on an output node.)

   A small blue dot appears on the connection to show that the breakpoint is set. If you try to set **Break After** on a node that has unconnected terminals, for example, its failure terminal, you see an informational message

   ```
   Breakpoints are not allowed at failure ports
   ```

   The breakpoints you set on all connected terminals are active.

   To clear breakpoints, right-click on the connection and select **Clear** or right-click on a message processing node and select **Clear All**. If, for example, you selected **Break After** on an input node that has four routes branching off its out terminal, breakpoints are set on the connections between the input node and the next node in each of those four routes. Selecting **Clear All** removes these four breakpoints.

6. To start the debug session select **Debug Actions**->**Start Debugging** or use the equivalent toolbar icon.

   An instrumented copy of the message flow (containing the debugging nodes) is deployed to the same execution group as the assigned, original message flow. You are informed that the deploy has been initiated by the standard pop-up message

   ```
   BIP1520I Deploy operations successfully initiated.
   ```

   Note: Starting to debug deploys the complete contents of the execution group containing the message flow being debugged. The Operations view shows the status of the debug version of the message flow.

7. To send a message into the flow, put a message to the message flow input queue using the application that the message flow is designed for. While you are debugging, only one message at a time can go through the flow, even if you have multiple input nodes. You can stack the messages up on the input queue(s) but they will not be processed until any previous message has finished.

8. When the message flow processing reaches a breakpoint, the content of the message at that point is displayed in the **Message Content** pane.

   You can change the message, except the message header type and the message type, before continuing. To edit the message, expand the **Message Content** pane, click on the field value to be changed and put in your new value.

   Note: The fields in the debugger take a certain range of values and you are not advised to edit these values unless you are fully aware of the possible

range. Changing the value to one outside the range may have undesired effects on further processing of the message through the flow.

If a problem causes a message processing node to throw an exception, this exception is displayed in the **Message Content** pane. It shows standard information for all exceptions (message number of 2230 and trace text) and also the node and function which threw the exception. This can be helpful if the exception is caught by a breakpoint some way from the point at which the exception actually occurred. For example, if the processing is rolling back through the message flow or if there are a number of nodes between breakpoints. When an exception is thrown, you can either check the NT Event Viewer (application view) for messages giving further information about the cause of the exception, or set a breakpoint on the catch terminal of the input node. By setting a breakpoint on the catch terminal this will display the full exception list when the message is rolled back to the input node, and further information on the exception can be displayed in the **Message Content** pane.

Note: If you wish to open a sub flow you can either

a. Click on a lower stack level shown in the **Stack** pane to cause that nested flow to be displayed in the **Message Flow** pane.

   Note: This is only possible when a message flow is being debugged.
   or

b. Right-click in the **Flow** pane directly over a flow node which is a nested flow. A pop-up menu appears, from which you should select:
   - **Open Sub Flow**

   Selecting this item causes the **Message Flow** pane to display the subflow.

To return to the parent flow either

- Click on a higher stack level shown in the **Execution Stack** pane to cause that parent flow to be displayed in the **Message Flow** pane.

  Note: This is only possible when a message flow is being debugged.
  or
- Right-click in the **Message Flow** pane *not* directly on a connection or a flow node. A pop-up menu appears, from which you should select:
  – **Return to Parent Flow**

  Selecting this menu item causes the parent of the current flow, if there is one, to be displayed. If there is no parent flow, in other words the displayed flow is the top level flow, this menu item is not be available.

There are a number of functions to control the progress of the message through the message flow. These options are active only when a message flow is stopped at a breakpoint. Select the options from Debug Actions or use the equivalent toolbar icons:

a. **Go** Go to the next breakpoint or to the output node, if there are no further breakpoints set in the message flow.

b. **Step into** Step into a nested flow. The message proceeds into the nested flow and the flow is displayed in the **Message Flow** pane.

c. **Return** Return from a nested flow to the main flow, ignoring any other breakpoints in the nested flow, to a point just beyond the output terminal of the nested flow.

d. **Step over** Step over the next node or nested flow, even if the nested flow contains breakpoints.

e. **Run to Completion** Process the message to the output queue, ignoring all further breakpoints. If no problems are encountered in the message flow during the debug session, the session ends and you see `Debug Session ended` in the information pane.

You can continue debugging as many messages, one at a time, as desired before ending the debug session.

9. To stop debugging, select **Debug Actions** —>**Stop Debugging** or use the equivalent toolbar icon. The deployed instrumented (`<name>_debug_`) flow stops communicating with the Control Center while it completes processing any outstanding messages, then redeploys the original (uninstrumented) message flow. `BIP15201` is displayed to show that the operations are initiated.

   Stop Debugging is only active when a message flow is being debugged.

Part 2. Concepts and references

Chapter 12. Control Center concepts

This chapter introduces the Control Center by describing its role in an MQSeries Integrator broker domain, and defining those concepts that you need to understand as a Control Center user. For a comprehensive description of MQSeries Integrator concepts, see the *MQSeries Integrator Introduction and Planning*.

The Control Center has two main functions in a broker domain. These are:

- The creation, manipulation, and deployment of configuration data for a broker domain
- The monitoring and management of the operational state of the same broker domain

The workspace

The concept of the *workspace* is key to the operation of the Control Center. It is the term given to the "snapshot" of that part of the shared configuration data that you, as a Control Center user, want to work with. The shared configuration can consist of many brokers, collectives, execution groups, message flows, message sets, and topics, many of which may be of no interest to you. The workspace allows you to work with a subset of this overall set of configuration data.

All brokers, collectives, execution groups, and topics in the shared configuration always appear in your workspace. However, you can choose which message flows and message sets you want to appear, to make your view of the shared configuration more manageable. For example, if there are 500 message flows defined in the shared configuration, you can choose to see only the 10 that are owned by you. You do this using an operation called *add*. Similarly you can *remove* any configuration resource from your workspace.

In summary, the workspace is a collection of references to specific objects in the configuration.

Working with configuration data

When a broker is created using the **mqsicreatebroker** command, and started for the first time using the **mqsistart** command, it has no configuration to run. A broker can perform useful functions only when it has been given a configuration to run by the Control Center user.

Configuration data is of three types:

Assignments data
: Is the assignment of: execution groups to brokers; message flows to execution groups; and message sets to brokers.

Topology data
: Is the relationship between brokers and collectives in a publish/subscribe network in the broker domain.

Topics data
: Is topics and associated Access Control List (ACL) entries used in a publish/subscribe network in the broker domain.

Configuration and message repositories

Configuration data of all three types is created by Control Center users, and is managed by the *Configuration Manager* in two repositories called the *configuration repository* and the *message repository*.

- The message repository contains definitions of message sets.
- The configuration repository contains all other configuration data.

There is only ever one Configuration Manager in a broker domain, but there can be any number of instances of the Control Center.

Shared and deployed configurations

The Configuration Manager manages two versions of the configuration data. These are the *shared configuration* and the *deployed configuration*.

Shared configuration
Consists of configuration data as created by one or more Control Center users and made visible to other Control Center users in the broker domain.

Deployed configuration
Is the configuration data that is operational, or having an effect, in the broker domain.

Configuration data in the shared configuration is sent to brokers by the Configuration Manager under the direction of Control Center users, by means of an operation called *deploy*. If deployment is successful, the Configuration Manager updates its deployed configuration accordingly.

Chapter 13. Concepts of message flows

This chapter introduces concepts which you might need to understand when working with message flows. One term which you might encounter is **Execution Group**. Execution group describes the run-time environment, provided by a broker, for a set of deployed message flows. For more information about execution groups, see the *MQSeries Integrator Introduction and Planning* book.

Using the IBM supplied message flows

Before you can use the supplied message flows, there are several actions you must complete. The message flows and the topology definitions in the import file make some assumptions. For full details of these assumptions, and how to use the message flows, see the *MQSeries Integrator Installation Guide* for your operating system. A summary of actions is provided here.

1. Import the Postcard message set into the message repository using the command **mqsiimpexpmsgset**. You must restart the Configuration Manager to force it to pick up these changes.
2. Check out the Topology. Import the message flow and topology definitions from the supplied file and save them into the shared configuration repository.
3. If you want to run the Postcard verification program, you must assign the Postcard message set to the broker.
4. Deploy the message flows and the Postcard message set to the broker.
5. Check the success of the deployment: select the *Log* view and refresh the contents (you can click the green refresh icon or select **View —> Refresh**). It can take a few minutes for the deployment messages and responses flowing between the Configuration Manager and the broker to be displayed. Keep refreshing this view until you see the completion messages.

Copying the default message flows

If you want to deploy either of the default message flows, you are recommended to make a copy of it. This preserves the default message flow in your configuration repository for future reuse.

To make a copy of a default message flow:

1. Click the **Message Flows** tab.
2. Select the message flow you want to use from the folder IBM Default Message Flows, and click **Copy**.
3. **Paste** the message flow into your folder.
4. A copy of the default message flow is created as a new message flow (with a new icon beside it). Click on the copy and select **Rename** to give it a unique name.
5. Make the changes you need to tailor your new copy of the supplied message flow.
6. Save your new message flow in the configuration repository using either **Check In** or one of the **File —> Check In** menu options.

Chapter 14. Concepts of deployment

This chapter describes the concepts behind the deployment tasks:

Types of deployment

You can deploy assignments data, topics data, topology data, or all three types of data at once. For each of these types of configuration data, you can request:

- A complete deployment
- A delta deployment

In addition, you can request a forced deployment. This type of deployment deploys all data and is valid at anytime.

Complete deployment

A complete deployment:

1. Deletes all configuration data of that type that is currently deployed on the target brokers
2. Creates new configuration data from the shared configuration

For example, if you request a complete deployment of topics data, the Configuration Manager deploys instructions to all brokers to delete *all* currently deployed topics data and create a new set of topics data from those in the shared configuration.

Delta deployment

When you request a delta deployment, the Configuration Manager compares the configuration data of that type that is currently deployed on the target brokers with the shared configuration, and deploys only the differences between the two versions. Therefore, the delta deployment is better for performance, especially when you have a large amount of configuration data in the shared configuration.

Forced deployment

The forced deployment, which overrides any outstanding deployment request, is used typically to correct error situations. Therefore, to maintain consistency of the configuration data throughout the broker domain, a forced deployment is allowed only when deploying all types of configuration data. A forced deployment is always a complete deployment.

A summary of deployment actions

Table 3 summarizes the available deployment actions, showing:

- The type of deployment supported for each type of configuration data
- The Control Center view from which the deployment can be requested
- The brokers to which the deployment can be targeted

Table 3. Deployment summary

Data deployed	Complete	Delta	Forced	From Control Center view	Target
Assignments	Yes	Yes	No	Assignments	Single broker, single execution group and all brokers
Topics	Yes	Yes	No	Topics	All brokers
Topology	Yes	Yes	No	Topology	All brokers

Table 3. Deployment summary (continued)

Data deployed	Complete	Delta	Forced	From Control Center view	Target
All types	Yes	Yes	Yes		All brokers
Note: The Topics, Topology, and All types deployments must apply to *all* brokers to maintain consistent configuration data throughout the broker domain.					

The stages of the deployment process

Deployment of configuration data takes place in two stages.

Stage one of deployment

During stage one of deployment, which is synchronous, the Configuration Manager sends a configuration data stream to the SYSTEM.BROKER.ADMIN.QUEUE of each target broker. When the configuration data has been sent to all relevant brokers, control is returned to you.

If the first stage is successful, message BIP1520I is displayed identifying the brokers to which the data was deployed.

However, if an error is detected during the first stage of deployment, the deployment is abandoned: no configuration data is sent to any broker, and an appropriate error message is displayed in a Control Center dialog box.

Stage two of deployment

During stage two of the deployment process, which is asynchronous, the target brokers process the received configuration data and return a response on the Configuration Manager's SYSTEM.BROKER.ADMIN.REPLY queue. The Configuration Manager then updates its record of the deployed configuration.

Deployment of data to a target broker might be only partially successful. This is because the unit of deployment on a broker is the execution group: the deployment of one execution group to a broker might succeed, but the deployment of another to the same broker might fail. A unit of deployment is transactional, however, so either all changes are made to a given execution group or no change is made.

For deployment purposes, topics and topology data are considered to belong to a separate unit of deployment, so either all changes are made to both topics and topology, or no change is made.

Which data is deployed?

When a deployment of any type of configuration data takes place, the data of that type that has been checked into the shared configuration by all Control Center users in the broker domain is that which is deployed to the configuration repository. Data that has not been checked in is not deployed. Note also that descriptive text that you can supply when defining Control Center resources is not deployed.

If some data has not been checked in

If some data has not been checked-in, leaving the shared configuration in an inconsistent state, the deployment is likely to fail. If the Configuration Manager detects an inconsistency, you receive a message indicating that some Control Center resources are not checked in.

To help avoid this situation occurring, you can request a list of all resources in your workspace that have not been checked in (using the **File —> Check In List** action) before you deploy. You can also check in all checked-out configuration data in your workspace using the **File —> Save to Shared** action. Of course, if multiple users are creating shared configuration data, that activity must cease while a deployment takes place, and all users must check in any checked-out resources before the deployment is requested.

Chapter 15. Concepts of debugging

This section covers the concepts behind the debugger problem determination tool.

Display panes

The debugger, accessed as a alternative screen under the Message Flows tab, has four panes: Stack, Flow, Message and Information.

The Stack pane

The Stack pane displays the execution stack of the message when it is at a breakpoint. This stack consists of the main flow plus any nested flows that contain the breakpoint. Clicking on a stack level causes that nested flow or main flow to be displayed in the Flow pane.

The Flow pane

The Flow pane displays the flow selected in the stack window (by default, the most nested flow). The Flow pane also allows you, with pop-up menus, to descend into a nested flow or to return to the parent flow, if any. You can set breakpoints in the Flow pane, which is also used to display where the message is currently stopped. A maximum of 50 breakpoints can be set in any given message flow. When a breakpoint is encountered, it will be displayed in a breakpoint icon in the Flow pane.

The Message pane

The Message pane shows the current message at the place where the flow is stopped due to a breakpoint (or an action like step-over). You can change the content of the message in this window before allowing the message to continue.

The Information pane

The Information pane displays relevant information, such as when a message (that had been previously stopped at a breakpoint) has been delivered.

Basic operation

While it is activated (from the time you start the debug session to the time you stop it), the debugger is continuously in a state of waiting for a message to be placed on any of the flow's input queues. The debugger does not place these messages itself, because it is assumed that you already have a message-generating application that the message flow was designed for.

When a message appears on an input queue, it will progress through the flow in a normal manner until a breakpoint is encountered. Breakpoints can be set and cleared at any time. At that time, the contents of the message as well as the destination list and exception list will be displayed by the control center and you will have an opportunity to change the message before continuing. If an exception is thrown during message processing, and if the backwards flow passes through a node that has a breakpoint set, the exception will be shown to the user, who then has an opportunity to change the exception (for example, the label or the message number) before continuing.

Only one message is allowed to proceed through the flow at a time, even if there are multiple input nodes, and even if the users have requested additional thread instances in their configuration.

Multiple simultaneous debug sessions

The following restrictions apply if you want to debug more than one flow at a time:

- You may only debug one flow at a time from a single Control Center.
- As many as 10 different users may be debugging simultaneously in the same execution group. (If you attempt to debug a message flow in an execution group already being debugged by 10 other users, you will receive a message informing you that too many users are currently using that execution group.)
- The same flow cannot be debugged at the same time by multiple users, because the deployed flows would compete for input queues. (For that reason, a flow being debugged is locked by the Configuration Manager).
- There is no limit to the number of different execution groups that may be in use for debugging.
- If you want to run debug sessions in multiple control centers on the same Windows NT workstation at the same time, you must adjust the port numbers on the options panes of those Control Centers so that they are not the same.

If you are running on a UNIX system we recommend that your system administrator sets two system services in the **etc/services** file on any machine running a broker. The services are **mqsidbgmin** and **mqsidbgmax**. These services are set to the minimum and maximum range of port numbers to be used by the Control Center when sending breakpoint information to the broker at runtime for debugging purposes. Each execution group uses one port number, so there should be a big enough range to accommodate the number of execution groups on that machine. If the range of ports is found to be insufficient at run time (or nonexistent because the services were not set in **etc/services**), the broker which is running, chooses ports by looking for unused ports starting at port number 25000. Any free port equal to or greater than 25000 becomes a candidate port, and a sufficient number will be chosen to accommodate the number of execution groups on that machine.

Error handling

When debugging a given message flow '*mf1*' (which must have already been assigned to an execution group), the debugger will generate a new flow called '*mf1*_debug_'. '*mf1*_debug_' will be assigned to the same execution group and deployed when the debug session starts. When the debug session ends, the original *mf1* will be redeployed to the selected execution group. This also happens if the Control Center is exited normally (that is, it will issue an automatic ″Stop Debugging″ action if necessary.) If you stop the broker while '*mf1*_debug_' is deployed, the Control Center will issue a pop-up warning indicating that a ″Stop Debugging″ action should be issued when the broker is restarted.

You may occasionally see an '*mf1*_debug_' flow deployed to an execution group even when a debug session is not active. This could be the result of the Control Center crashing, or some other extraordinary event. Such an 'orphaned' debug flow will behave the same as a normal, uninstrumented, flow (in other words, the debug nodes that instrument the flow will be inactive).

If you change a message in the message pane during a debug session, but accidentally enter an invalid value (for example, text where a number is expected), a pop-up message will indicate the error and you will be given another chance to enter a valid value before continuing.

Chapter 16. Concepts of XML messages

Self-defining or generic XML messages are those whose content are documents that adhere to the XML specification. The following sections describe how these messages are represented in a tree of syntax elements.

The following topics are discussed:
- "XML Declaration"
- "Document Type Declaration" on page 138
- "The XML message body" on page 142

The name elements used in this description (for example, XmlDecl) are provided by MQSeries Integrator for symbolic use within the ESQL. The ESQL defines the processing of message content that is to be performed by the nodes, such as filter node, within a message flow. They are not a part of the XML specification itself. You can find examples of ESQL syntax in *MQSeries Integrator ESQL Reference*.

The information provided here does not provide a full definition or description of XML terminology, concepts, and message constructs: it is a summary that highlights aspects that are important when you use XML messages with MQSeries Integrator. For further information about XML, see the IBM web site at:

`http://www.ibm.com/developer/xml`

XML Declaration

The beginning of an XML message must contain what is called an XML declaration.

An XML declaration might take the following form in the XML bit-stream:

```
<?xml version="1.0" standalone="yes" encoding="UTF-8"  ?>
```

The XML declaration must be at the beginning of every XML message.

XmlDecl

This is a name element that corresponds to the XML declaration itself. The XmlDecl element must be a child of the root element, and is the element that is written to a bit-stream first. This element can have three children of the following types:

1. Version

   The version element is a value element and stores the data corresponding to the version string in the actual declaration. It is always a child of the XmlDecl element. For example, for the declaration shown above the version element would contain the string value ″1.0″.

2. Standalone

   The standalone element is a value element and stores the data corresponding to the value of the standalone string in the declaration. It is always a child of the XmlDecl element. The values for the standalone element must be the string ″yes″ or ″no″.

   ″no″ is the default: this means that processing of the message depends on an external (DTD) reference.

3. Encoding

   The encoding element is also a value element and is always a child of the XmlDecl element. The value of the encoding element is a string which corresponds to the value of the encoding string in the declaration. In the example shown above the encoding element would have a value of ″UTF-8″.

   Note: MQSeries encodings cannot be specified in this element.

Document Type Declaration

The document type declaration (DTD) of an XML message is represented by a syntax element of type DocTypeDecl and its children and descendants. These comprise the DOCTYPE construct.

Only internal DTD subsets are represented in the syntax element tree. External DTD subsets (identified by the SystemID or PublicId elements described below) can be referenced in the message but those referenced are not resolved in the MQSeries Integrator run-time environment.

DocTypeDecl

The DocTypeDecl is a named element and must be a child of the root element. It is written to the bit-stream before the element that represents the body of the document during serialization. The following can be specified within this element:

1. IntSubset

   IntSubset is a named element that groups all of those elements that represent the DTD constructs contained in the internal subset of the message. Although the IntSubset element is a named element its name is not relevant.

2. SystemId

   SystemId is a value element and is used to represent a general system identifier construct found in an XML message. It can be a part of a DocTypeDecl or a NotationDecl element. The value of the SystemId is a URI, and is typically a URL or the name of a file on the current system. A system identifier of the form `SYSTEM "Note.dtd"` has a string value of ″Note.dtd″

3. PublicId

PublicId represents a general public identifier construct found in an XML message. It can be a part of a DocTypeDecl or a NotationDecl element. The value of the PublicId is typically a URL.

NotationDecl

The NotationDecl element represents a notation declaration in an XML message. It is a name element whose name corresponds to the name given with the notation declaration. It must have a SystemId as a child, and it can optionally have a child element of type PublicId.

Entities

Entities in the DTD are represented by one of five named element types described below.

1. ParameterEntityDecl

   The ParameterEntityDecl represents a parameter entity definition in the internal subset of the DTD. It is a named element and has a single child element that is of type EntityDeclValue. For parameter entities the name of the entity does not include the percent sign %. In XML a parameter entity declaration takes the form:

   ```
   <!ENTITY % inline "#PCDATA | emphasis | link">
   ```

2. ExternalParameterEntityDecl

   The ExternalParameterEntityDecl represents a parameter entity definition where the entity definition is contained externally to the current message. It is a named element and has a child of type SystemId. It can also have a child of type PublicId. The name of the entity does not include the percent sign %. In XML an external parameter entity declaration takes the form:

   ```
   <!ENTITY % bookDef SYSTEM "BOOKDEF.DTD">
   ```

   This is represented by an ExternalParameterEntityDecl element of name bookDef with a single child of type SystemId with a string value of “BOOKDEF.DTD”.

3. EntityDecl

   The EntityDecl element represents a general entity and is declared in the internal subset of the DTD. It is a named element and has a single child element which is of type EntityDeclValue.

   An entity declaration of the form:

   ```
   <!ENTITY bookTitle "User Guide">
   ```

   has an EntityDecl element of name ″bookTitle″, and a child element of type EntityDeclValue with a string value of “User Guide”.

4. ExternalEntityDecl

   The ExternalEntityDecl element represents a general entity where the entity definition is contained externally to the current message. It is a named element and has a child of type SystemId. It can also have a child of type PublicId.

   An external entity declaration of the form:

   ```
   <!ENTITY  bookAppendix SYSTEM "appendix.txt">
   ```

   has an EntityDecl element of name ″bookAppendix″ and a child element of type SystemId with a string value of ″appendix.txt″.

5. UnparsedEntityDecl

An unparsed entity is an external entity whose external reference is not parsed by an XML processor.

The UnparsedEntityDecl is named element. It has a child of type SystemId and optionally a child of type PublicId. The presence of NDATA after the SystemId in the entity declaration indicates that this entity is not parsed by the XML processor. After NDATA is the name of a corresponding notation declaration. In XML an unparsed entity declaration takes the form:

```
<!ENTITY pic SYSTEM "scheme.gif" NDATA gif>
```

- NotationReference

  The NotationReference name element represents a reference to a notation declaration from within an UnparsedEntityDecl element. It is always a child of an UnparsedEntityDecl element.

EntityDeclValue

This value element represents the value of an EntityDecl, or a ParameterEntityDecl defined internally in the DOCTYPE construct. It is always a child of an element of one of those types, and is a value element. For the following entity:

```
<!ENTITY  bookTitle "User Guide">
```

the EntityDeclValue element has the string value ″User Guide″.

ElementDef

The ElementDef name-value element represents the <!ELEMENT construct in a DTD. The name of the element that is defined corresponds to the name member of the syntax element. The value member corresponds to the element definition.

AttributeList

The AttributeList name element represents the <!ATTLIST construct in a DTD. The name of the AttributeList element corresponds to the name of the element for which the list of attributes is being defined.

AttributeDef

The AttributeDef name element describes the definition of an attribute within a <!ATTLIST construct. It is always a child of the AttributeList element. The name of the syntax element is the name of the attribute being defined. It can have three children:

1. AttributeDefValue

   For attributes of type CDATA (see AttributeDefType below) the AttributeDefValue gives the default value of the attribute.

2. AttributeDefDefaultType

   The AttributeDefDefaultType syntax element is a value element which represents the attribute default as defined under the attribute definition. The value can be one of the following strings:
   - #REQUIRED
   - #IMPLIED
   - #FIXED

3. AttributeDefType

   The AttributeDefType syntax element is a name-value element whose name corresponds to the attribute type found in the attribute definition. Possible values for the name are:
   - CDATA
   - ID

- IDREF
- IDREFS
- ENTITY
- ENTITIES
- NMTOKEN
- NMTOKENS
- NOTATION

If there is an enumeration present for the attribute definition the entire enumeration string is held as a string in the value member of the name-value syntax element. The value string starts with an open bracket "{" and ends with a close bracket "}". Each entry in the enumeration string will be separated by a 'l' character. For an enumerated type that is not a NOTATION, the name member of the syntax element is empty.

DocTypePI and ProcessingInstruction

The DocTypePI element represents a processing instruction found within the DTD. The ProcessingInstruction element represents a processing instruction found in the XML message body.

Both of these elements are name-value elements. In both cases, the name of the element is used to store the processing instruction target name, and the value contains the character data of the processing instruction. The value of the element can be empty. The name cannot be the string ″XML″ or any uppercase or lowercase variation of ″XML″.

DocTypeWhiteSpace and WhiteSpace

The DocTypeWhiteSpace element represents whitespace found inside the DTD that is not represented by any other element. The WhiteSpace element represents any white space characters found in the message body that is not represented by any other element. Both are value elements.

For example, white space within the body of the message is reported as element content using the pcdata element type, but white space characters found between the XML declaration and the beginning of the message body are represented by the WhiteSpace element.

```
<?xml version="1.0"?>       <BODY>....</BODY>
```

The characters between ″1.0″?>″ and <BODY> are represented by the WhiteSpace element. White space characters found within a DocType between two definitions are represented by the DocTypeWhiteSpace element.

```
| <!DOCTYPE Note SYSTEM "Note.DTD">
| <!ENTITY % bookDef SYSTEM "BOOKDEF.DTD">  <!ENTITY bookTitle "User Guide"> ]>
```

The characters between DTD″> and <!ENTITY are represented by the DocTypeWhiteSpace element.

DocTypeComment and Comment

Comments in the XML message are represented by the Comment and DocTypeComment elements. The former is used within the message body, the latter within the DTD. Both element types are value elements where the value string contains the comment text.

The XML message body

Every XML message must have a body element. The body element is a top level XML element which encapsulates the whole of the body. XML elements are represented in the syntax element tree with a type of “tag”.

- tag

  The tag syntax element is the default name element supported by the XML parser and is the most common element. This element can have many children of many different types. XML attributes that are attached to an XML element are represented by a series of ″attr″ elements that are children of the tag element. Similarly, sections of PCDATA which are content of the XML element are represented by syntax elements of type ″pcdata″. ″tag″ elements can also have other tag elements as children.

  – attr

    The attr element is the principal name-value element supported by the XML parser. It is used to represent attributes that are associated with elements in the XML message. The name and value of the syntax element correspond to the name and value of the attribute being represented. ″attr″ elements have no children and must always be children of a ″tag″ element.

  – pcdata

    Element content is represented by the pcdata value element. There can be more than one pcdata element child of a single tag element. In these cases they would be separated by any syntax elements that represent XML constructs allowed within element content, including ″tags″, ″ProcessingInstruction″, ″Cdata″, ″EntityDecl″.

The following XML illustrates an extract of message body:

```
<PERSON age="32" height="172cm">
  <FIRSTNAME>Cormac</FIRSTNAME>
  <SECONDNAME>Keogh</SECONDNAME>
</PERSON>
```

This is represented in the syntax element tree as:

- One **“tag”** element with a name of ″PERSON″. This tag has seven children.
  1. Two **attr** (name-value) with names ″age″ and ″height″ and string values ″32″ and ″172cm″ respectively.
  2. One **pcdata** (value) element with string value containing the white space character data found between ″172cm″> and <FIRST.
  3. One **tag** (name) with a name ″FIRSTNAME″. This tag has one child:
     – One **pcdata** (value) containing the string value ″Cormac″.
  4. One **pcdata** (value) element with string value containing the white space character data found between TNAME> and <SECOND.
  5. One **tag** (name) with a name ″SECONDNAME″. This tag has 1 child:
     – One **pcdata** (value) containing the string value ″Keogh″.
  6. One **pcdata** (value) element with string value containing the white space character data found between DNAME> and </PERSO.

ProcessingInstruction

This is described in “DocTypePI and ProcessingInstruction” on page 141.

WhiteSpace

This is described in “DocTypeWhiteSpace and WhiteSpace” on page 141.

Comment

This is described in “DocTypeComment and Comment” on page 141.

AsisElementContent

Normally an XML processor must replace any occurrences of the characters ampersand (&), less than (<), greater than (>), double quote (”), and apostrophe (’) with an escape sequence that is used to represent them (&, <, >, ", and '). The escape sequences are defined as entities.

The AsisElementContent is a value element that is similar to the pcdata element but provides a means to suppress this behavior for the content of an element. Occurrences of any of the characters in the value of an AsisElementContent element are substituted by their appropriate entity reference.

To illustrate how AsisElementContent might be used, please see the following example:

```
SetOutputRoot.XML.Message.(XML.AsisElementContent) = '&';
```

this gives: `<Message>&</Message>`.

```
SetOutputRoot.XML.(XML.AsisElementContent).Message = '&';
```

gives: `&`. This shows that AsisElementContent causes only the literal data to be written.

Note: Any path name elements after AsisElementContent are discarded.

CDataSection

CData sections in the XML message are represented by the CDataSection value element. The content of the CDataSection element is the value of the CDataSection element without the <![CDATA[that marks the beginning, and without the]]> that marks the end of the Cdata section.

For example, the following Cdata section:

```
<![CDATA[<greeting>Hello, world!</greeting>]]>
```

is represented by a CDataSection element with a string value of:

```
"<greeting>Hello, world!</greeting>"
```

Unlike pcdata, occurrences of <, >, &, “, and ’ are not translated to their escape sequences when the Cdata section is written out to a serialized message.

EntityReferenceStart and EntityReferenceEnd

When an entity is encountered in the XML message it is reported in the syntax element tree in expanded form. In order to determine if a section of the tree has been derived from an expanded entity, a couple of marker elements are placed in the tree to denote the beginning and end of an entity’s expansion.

- The EntityReferenceStart element is a value element that marks the beginning of an entity expansion.
- The EntityReferenceEnd element is a value element which marks the end of an entity expansion.

The value of both elements corresponds to the name of the entity being expanded. Any syntax elements found between these two place holders, and their children have been derived from the expansion of the entity in question.

Chapter 17. Concepts of NEONRules and NEONFormatter Support for MQSeries Integrator

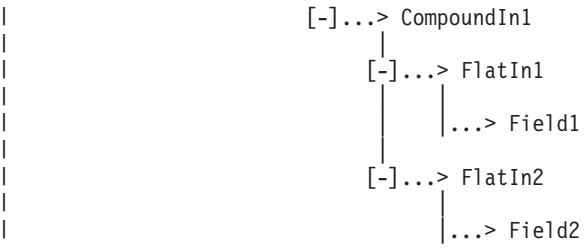
The support for NEONRules and NEONFormatter in MQSeries Integrator V2.0.2 has been substantially enhanced over previous versions of MQSeries Integrator and is now based on version 5.2 of the NEONRules and NEONFormatter product. All the improvements in NEONRules and NEONFormatter functionality which V5.2 of that product offers over and above previous versions can be leveraged by MQSeries Integrator V2.0.2. In addition to this, MQSeries Integrator V2.0.2 contains technology aimed specifically at integrating NEONRules and NEONFormatter fully into the MQSeries Integrator message flow paradigm. The combination of NEONRules and NEONFormatter V5.2 with the MQSeries Integrator specific technology is referred to as the NEONFormatter and NEONFormatter Support for MQSeries Integrator.

The NEONMSG parser

At the heart of the new Rules and Formats technology is the NEONMSG message parser. NEONMSG is the new message domain for messages defined in the Rules and Formats database, and is intended to replace the old NEON domain. The old domain is still present in MQSeries Integrator V2.0.2, however it is intended for backward compatibility purposes only and it cannot be used to access any of the new Rules and Formats Support functionality. Like other message parsers (such as XML or MRM), the NEONMSG parser translates wire format messages in its domain into MQSeries Integrator message trees, and vice versa. However it has some particular capacities and restrictions.

Parsing a NEON Format message into an MQSeries Integrator message tree

The NEONMSG parser can only parse messages defined as input formats in the Rules and Formats database. It uses the MQSeries Integrator Message Type property to identify the input format in the database. If an input format with the name specified in the Message Type property does not exist, an exception will be thrown. The structure of the MQSeries Integrator message tree generated by the NEONMSG parser is two-dimensional. The top level format name is omitted from the tree, as this information is contained in the message properties. Each component input format below the top level is added as a vertical layer to the tree. Fields of a flat input format are added as children of the owning format. For example, a format defined as following the NEONFormatter User Interface:



Will result in the generation of a message tree as follows:

```
(0x1000000)NEONMSG = (
  (0x1000002)FlatIn1 = (
    (0x3000001)Field1 = 'some data'
  )
```

Parsing a NEON Format message into an MQSeries Integrator message tree

```
        (0x1000002)FlatIn2 = (
          (0x3000001)Field2 = 'some more data'
        )
      )
```

It is instructive to compare this with the message tree that would be generated by the old, deprecated, NEON domain parser for the same input format:

```
(0x1000000)NEON = (
  (0x3000000)Field1 = 'some data'
  (0x3000000)Field2 = 'some more data'
 )
```

As can be seen the new NEONMSG parser creates vertical levels in the message tree to represent the component formats of a compound format[2] message, whereas the old NEON parser flattens all the fields in the message to a single vertical level. The data content of the fields depends of course on what values were actually present in the incoming message.

Reserializing a message tree into a NEONFormatter message format

The old NEON domain parser was unable to reserialize an MQSeries Integrator message tree into a NEONFormatter message format. The NEONMSG parser does have this capability; however it can only reserialize messages defined as output formats in the Rules and Formats database, or messages defined as input formats in the Rules and Formats database which have not been modified by the message flow. It is not possible to reserialize an input format message where the body of the message has been modified in any way. The wire format of the reserialized message will be as defined in the NEONFormatter User Interface.

Using the NEONMSG parser with ESQL

The fact that the NEONMSG parser generates a different message tree from the old NEON parser obviously has consequences for the way message fields in a NEONMSG domain message are referenced from ESQL

Referencing fields in a NEONMSG domain message

Using the example message format above, if this were parsed by the NEONMSG parser the field **Field1** would be referred to from a Compute node as:

```
InputRoot.NEONMSG.FlatFmt1.Field1
```

or

```
InputBody.FlatFmt1.Field1
```

Compare this with the way the same field would be referred to if the message had been parsed by the deprecated NEON parser:

```
InputRoot.NEON.Field1
```

or

```
InputBody.Field1
```

The extra vertical layers which the NEONMSG parser generates can make processing a NEONMSG message in a message node such as Compute a much

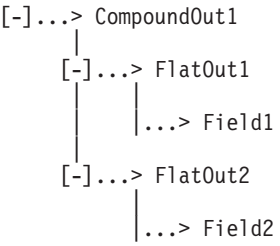
---

2. ″Compound format″ is a NEON-specific term. NEON formats are divided into ″flat″ and ″compound″ formats. A flat format contains only data fields. A compound format contains both flat formats and other compound formats.

simpler task. However if you are migrating from the old NEON parser then changes may be required to your ESQL to reflect the existence of these new vertical layers.

Creating a NEONMSG domain message

It is possible to create a NEONMSG domain message in the Compute node. This was not possible with the old NEON domain parser. The requirements for creating a NEONMSG domain message are the same as those for creating any other domain message: the message body tree must be created and populated, and the message properties set to reflect the message attributes. As the NEONMSG domain parser only reserializes output format message trees, or input format messages which have a valid bitstream associated with them, it is only possible to create NEONMSG messages which are defined as output formats in the Rules and Formats database; input format messages may not be created. If the following output format were defined in the NEONFormatter User Interface:



Then to create a message in this format which would be successfully reserialized by the NEONMSG parser, the following ESQL would be required:

```
SET OutputRoot.NEONMSG.FlatOut1.Field1 = 'some data';
SET OutputRoot.NEONMSG.FlatOut2.Field2 = 'some more data';
SET OutputProperties.MessageType = 'CompoundOut1';
```

An important point to note is that when a NEONMSG message is created in this way, the output controls associated with the format of the created message (as defined in the NEONFormatter User Interface) are not applied to the message data until it is reserialized. Thus if **Field1** had an associated output control to upper case the contents of the field, querying the contents of **Field1** as it passed through the message flow would reveal the value 'some data'. However, when the message was reserialized and placed on an output queue, the contents of the field would be seen to be 'SOME DATA'.

The NEONTransform and NEONMap nodes

The NEONTransform node is intended as a direct replacement for the deprecated NEONFormatter node. As such it is capable of reproducing all the function of that node. However the underlying changes to the NEONRules and NEONFormatter support for MQSeries Integrator V2.0.2 mean that slightly different usage may be required in some circumstances.

Map Name and Map Version

Part of the extra functionality provided by the NEONRules and NEONFormatter Support for MQSeries Integrator V2.0.2 package is the implementation of explicit Map object. These are defined in the NEONFormatter User Interface and provide a mechanism for explicitly mapping input message fields to output message fields. See the *NEONRules and NEONFormatter Support: User Guide* for more details.

Other attributes

A comparison of the NEONFormatter and NEONTransform node will reveal that the Output Wireformat attribute has been removed from the latter. Furthermore, although the Output Domain, Output Message Set and Output Message Type attributes remain, they have a slightly different effect. In order to understand the reason for these changes it is necessary to understand the difference between the old NEON domain parser and the new NEONMSG domain parser. The NEON parser could not reserialize an output format message, therefore the NEONFormatter node had to generate an already-serialized bitstream, rather than a message tree as the body of its output message. Effectively this was equivalent to having a ResetContentDescriptor node embedded in the NEONFormatter node. Therefore the Output Domain, Output Message Set, Output Message Type and Output Wireformat attributes were intended to implement the function of the embedded ResetContentDescriptor node.

The NEONMSG parser is capable of reserializing an output format message, therefore the NEONTransform node generates a message tree, rather than a bitstream, as the body of its output message. When this message tree reaches a point in the message flow where it needs to be reserialized, the NEONMSG parser does so. The ″embedded ResetContentDescriptor node″ from the NEONFormatter node is not required in the NEONTransform node and has therefore been removed. If ResetContentDescriptor node functionality is required, for example, resetting the domain of the output message to BLOB, or changing the Output Message Type to a different output format, a real ResetContentDescriptor node should be attached to the appropriate output terminal of the NEONTransform node.

Output Domain

The Output Domain of a message can be set to one of two values: NEONMSG (the default) or XML. A message with an Output Domain of NEONMSG is output as a normal NEONMSG domain message. Any subsequent modification or reparsing of this message will be done using the NEONMSG domain parser. If a message is output with an Output Domain of XML, the message tree structure generated by the NEONMSG parser is used to create an XML message. For example, a message in the 'CompoundOut1' format as detailed above, would, if output in the XML domain from a NEONTransform node, consist of the following XML:

```
<CompoundOut1>
       <FlatOut1>
             <Field1>some data</Field1>
       </FlatOut1>
       <FlatOut2>
             <Field2>some more data</Field2>
       </FlatOut2>
</CompoundOut1>
```

As can be seen, the outermost XML element consists of the outermost format name; the other XML elements reflect the structure of the 'CompoundOut1' output format.

Output Message Type and Output Message Set

By default, the Message Type of a message output from the NEONMap and NEONTransform nodes is set to the value of the Target Format attribute. If the Output Message Type attribute is explicitly set to a different value than the Target Format attribute then it sets up a ″delayed transform″ to be applied to the message upon reserialization. Thus if a NEONTransform node were to transform a message from input format 'WireIn1' to output format 'ContentModel1', setting the Output

Message Type attribute to 'WireOut1', then on reserialization (at an ResetContentDescriptor or MQOutput node), the message would be transformed a second time, into the 'WireOut1' output format.

The names chosen for the example above are intended to demonstrate the major intent of the ″delayed transform″ behavior. An aspect of the underlying differences between the NEONRules and NEONFormatter Support for MQSeries Integrator V2.0.2 and that included with previous versions of MQSeries Integrator is a shift in the NEONFormatter paradigm so that it integrates better into the overall product architecture. The message processing model for MQSeries Integrator is three-stage: a message is parsed, processed, and reserialized. The processing takes place on a generic message tree which holds the data content of the message in a wire format-independent form. Until now the NEONFormatter paradigm within MQSeries Integratorhas been essentially two-stage: a message is reformatted from one wire format to another.

With the release of MQSeries Integrator V2.0.2 the NEONFormatter functionality has been upgraded in such a way that it can more easily support a three-stage parse-process-reserialize paradigm. Thus a message in the NEONMSG domain may be input to a message flow in a particular input wire format; a NEONTransform or NEONMap node converts this into a generic format, specifying as part of this operation the ultimate Output Message Type (output wire format) in which the message should be discharged from the message flow. When the message is reserialized in anticipation of this discharge (or due to the intervention of an ResetContentDescriptor node), the NEONMSG parser will automatically perform the necessary transform operation to convert the message into the specified output wire format. Of course, it is still possible to use the NEONMap and NEONTransform nodes as simple two-stage reformat operations if required.

The Output Message Type and Output Message Set attributes have no meaning within the XML domain and therefore, although the Output Message Type and Output Message Set standard properties will be set in the output message as specified, they will not affect the subsequent processing of the XML message in any way. Setting the Output Message Set attribute when the Output Domain is NEONMSG has the effect of changing the application group (as defined in the NEONRules User Interface) to which the message belongs.

The NEONMap node

The NEONMap node is identical to the NEONTransform node except that it performs only the mapping stage of a reformat. This means that message data is mapped from the fields of the input format to the fields of the specified target format according to either the default mapping (defined for that target format) or the specified Map Name and Map Version. After performing this mapping step, the NEONTransform node goes on to apply whatever output operations were specified for each field in the target format (as defined in the NEONFormatter User Interface). The NEONMap node does not perform this second step. For example, if a field in an output format had an upper casing output operation specified, the incoming data `some data` would be turned by the NEONTransform node into `SOME DATA` when it was mapped to that output format. By contrast, the NEONMap node would not apply the upper casing output operation and the incoming data would remain as `some data`.

The NEONRulesEvaluation node

The NEONRulesEvaluation node is intended as a direct replacement for the deprecated NEONRules node. Its attributes are identical to its predecessor, however the changes in the underlying NEONRules and NEONFormatter functionality mean that some differences in the behavior of particular Subscription Actions (as defined in the NEONRules User Interface) will be observed.

Map and Transform actions

These are identical in behavior to the NEONMap and NEONTransform nodes respectively. However it must be understood that it is now a message tree, rather than a bitstream, which is being Mapped and Transformed. This has implications for the situation where multiple Maps or Transforms are executed within a single subscription. To illustrate the differences between the NEONRules and NEONRulesEvaluation nodes' behavior in this circumstance, consider the following three formats:

- **Input1**: flat input format with a single, semicolon delimited field, **Field1**.
- **Output1**: flat output format with a single field, also named **Field1**. The output control for this field upper cases the field data and suffixes a semicolon to the end. The mapping for this format is such that **Field1** in the output format maps its data from **Field1** in the input format.
- **Output2**: flat output format with a single field, **Field2**. The output control for this field suffixes a semicolon to the end. The mapping for this format is such that **Field2** in the output format maps its data from **Field1** in the input format.

Given an input message of format **Input1** with the bitstream body `some data;`, the effect of applying two Reformat actions in the NEONRules node can be compared against the effect of applying two Transform actions in the NEONRulesEvaluation node:

Table 4. Comparison of the functions of the NEONRulesEvaluation node and NEONRules node

NEONRulesEvaluation node	NEONRules node

Transform
`Target Format = Output1`
	Reformat
	`INPUT_FORMAT = Input1`
	`TARGET_FORMAT = Output1`

PutQueue When serialized, the output wire format message body will consist of `SOME DATA;`	PutQueue When serialized, the output wire format message body will consist of `SOME DATA;`

Transform
`Target Format = Output2`
	Reformat
	`INPUT_FORMAT = Input1`
	`TARGET_FORMAT = Output2`

PutQueue When serialized, the output wire format message body will consist of `SOME DATA;;` **Note:** two semicolons.	PutQueue When serialized, the output wire format message body will consist of `SOME DATA;` **Note:** one semicolon..

There are numerous points for discussion here:

The Transform action has no equivalent of the Reformat's INPUT_FORMAT option. This is because the Transform action in the NEONRulesEvaluation node takes a message tree as input and produces a message tree as output, while the Reformat action in the NEONRules node takes a bitstream as input and produces a bitstream as output. The Reformat action has to perform three steps: parse the incoming bitstream, map the resulting input fields to the output format fields (applying any output operations), then reserialize the output format fields into a new bitstream. Because the Transform action works directly with message trees, it has no need to perform the first and third of these steps, and therefore does not need to be told the format of the input message. It will be instructive to consider in detail what takes place as each of the above actions are executed:

Table 5. behavior of the NEONRules node reformat action

Action	Behavior
INPUT_FORMAT = Input1 TARGET_FORMAT = Output1	• The incoming message bitstream, `some data;` is parsed as input format **Input1**, generating the field **Field1** with data `some data`. • **Field1** from the input format is mapped to **Field2** in the output format **Output1**. • **Output1** is reserialized, with output operations applied to produce the bitstream `SOME DATA;`
PutQueue	The message is propagated to the putqueue terminal. It is already serialized so no further reserialization will take place when it reaches the MQOutput node.

Reformat
`INPUT_FORMAT = Input1`
`TARGET_FORMAT = Output2`
	• Following the previous reformat, the message bitstream now consists of
	`SOME DATA;`
	This is a valid bitstream for input format **Input1**, so it is successfully parsed, generating the field **Field1** with data `SOME DATA`
	• **Field1** from the input format is mapped to **Field3** in the output format **Output2**.
	• **Output2** is reserialized, with output operations applied to produce the bitstream
	`SOME DATA;`

PutQueue	The message is propagated to the putqueue terminal. It is already serialized so no further reserialization will take place when it reaches the MQOutput node.

Table 6. behavior of the NEONRulesEvaluation node transform action

Action	Behavior

Transform
`Target Format = Output1`
	The incoming message tree (generated by the NEONMSG parser) consists of:
	`(0x1000000)NEONMSG = (`
	`(0x3000001)Field1 = 'some data' )`
	
	This is transformed to:
	`(0x1000000)NEONMSG = (`
	`(0x3000001)Field1 = 'SOME DATA;' )`

PutQueue	**Output1** is propagated to the putqueue terminal with the above message tree. When it reaches the MQOutput node the NEONMSG parser reserializes it into `SOME DATA;`

Transform
`Target Format = Output2`
	The incoming message tree (as generated by the previous Transform action) consists of:
	`(0x1000000)NEONMSG = (`
	`(0x3000001)Field1 = 'SOME DATA;' )`
	
	This is transformed to:
	`(0x1000000)NEONMSG = (`
	`(0x3000001)Field2 = 'SOME DATA;;' )`
	
	The output control associated with **Field2** causes a semicolon to be added to the data from the input field **Field1**. **Field1** contained the data
	`SOME DATA;`
	
	therefore adding an extra semicolon results in `SOME DATA;;`.

PutQueue	**Output2** is propagated to the putqueue terminal with the above message tree. When it reaches the MQOutput node the NEONMSG parser reserializes it into `SOME DATA;;`.

It can be seen that single NEONRulesEvaluation Transform actions function identically to single NEONRules Reformat actions, however multiple Transform actions within a single subscription may function differently to multiple Reformat actions within a single subscription. This means that the behavior of subscriptions imported from previous versions of MQSeries Integrator may change when they are executed by the NEONRulesEvaluation node in MQSeries Integrator V2.0.2. This is an unavoidable consequence of the fundamental redesign work which has gone into the NEONRules and NEONFormatter Support for MQSeries Integrator V2.0.2. One way to avoid or reverse such changes in behavior is to place each Transform action in a separate subscription and attach the resulting multiple subscriptions to a single rule. This will avoid any problems caused by performing multiple Transforms in a single subscription, while the overall effect of that rule on the message, remains unchanged. Further information about the NEONRulesEvaluation node can be found in “The NEONRulesEvaluation node” on page 150.

Propagate, Put Queue and Route actions

These three actions all result in a message being propagated to the appropriate output terminal of the NEONRulesEvaluation node. Therefore they all support the Message Domain, Message Set and Message Type options. The effect of setting these options is identical to the effect of setting the equivalent attribute on the NEONMap and NEONTransform nodes. In addition these actions support the Output CCSID and Output Encoding attributes. The effect of the Output CCSID and Output Encoding attributes is to set the CCSID and Encoding of the body of the outgoing message to the specified values.

Propagate

This has no attributes in addition to those described above. It causes a message to be output from the NEONRulesEvaluation node via the propagate terminal.

Put Queue

The Put Queue action causes a message to be output from the NEONRulesEvaluation node via the putqueue terminal. Before it is output, its Destination List is updated to reflect the values of the Target Queue and Target Queue Manager attributes. If the message output from the putqueue terminal subsequently reaches an MQOutput node which is configured with a Destination Mode of 'Destination List', it will be placed on the MQSeries queue specified in the Target Queue attribute, belonging to the MQSeries queue manager specified in the Target Queue Manager attribute.

The Put Queue action has the following additional attributes:

- **MQS Format**. Sets the Format field of the MQSeries header immediately preceding the output message body to the specified value.
- **MQS Propagate.** If this is set to NO_PROPAGATE, any RFH or RFH2 header on the input message is not copied to the output message. This is the default. If it is set to PROPAGATE, then any RFH or RFH2 header is copied to the output message.
- **MQS Persist**. This may be set to PERSIST or NO_PERSIST. It sets the Persistence of the output message (as defined in its standard properties folder) to TRUE or FALSE respectively. If no value is specified for this attribute the output message retains the persistence of the input message.
- **MQS Expiry.** If this is set to NO_PROPAGATE, the expiry time of the output message is set to Unlimited (in other words, it will never expire). This it the default. If it is set to PROPAGATE, the expiry time of an input message is copied across to the output message.

Route

The Route action causes a message to be output from the route terminal of the NEONRulesEvaluation node. Before it is output, its Destination List is updated to reflect the value of the Label Name attribute. If the message output from the route terminal subsequently reaches a RouteToLabel node, it may be routed to the Label node with the name specified in the Label Name attribute, depending on whether any other Label values have been appended to the Destination List of the message and the configuration of the RouteToLabel node. See "Using a RouteToLabel node" on page 66.

Access to Rules and Formats

In order to function the NEON nodes and parsers must be able to locate and access a database containing the definitions of theNEON Rules and Formats which they are to use. Creating and initializing this database is a once-only administrative operation and is covered in the *MQSeries Integrator Administration Guide*. The migration of existing Rules and Formats from previous versions of MQSeries Integrator is also covered in the same book. New Rules and Formats can be created using the NEONFormatter and NEONRules User Interfaces supplied with MQSeries Integrator V2.0.2.

To locate the Rules and Formats database:

The Rules and Formats database is deployed to a broker, so it is necessary for the environment variable, `NN_CONFIG_FILE_PATH`, to be defined by the user ID which is running the broker.

The `NN_CONFIG_FILE_PATH` variable must contain the full path to the directory containing the configuration file which holds database connection details for the Rules and Formats database. This configuration file must be called `neonreg.dat`. There is a sample of the `neonreg.dat` file:

- Under the MQSeries Integrator root installation directory in `sample/NEON` on the UNIX platform
- Under the MQSeries Integrator root installation directory in `examples\NEON` on the Windows NT platform

The value to use in the `NN_CONFIG_FILE_PATH` variable, so it points to `neonreg.dat` is:

- `MQSI_ROOT/sample/NEON` on the UNIX platform
- `MQSI_ROOT\examples\NEON` on the Windows NT platform

Chapter 18. C and COBOL default mappings

This appendix describes the defaults that the C and COBOL importers use when mapping C datatypes or COBOL datatypes to MRM datatypes. The data designer defining a message set in the Control Center might want to follow these defaults, but this decision will depend on the business usage of the data.

The MRM:

- Does not support pointer datatypes.
- Does not support the COBOL construct REDEFINES.
- Does not support the COBOL datatypes DBCS, external floating point, or binary items that have a PIC declaration greater than 9 digits.
- Does not fully support the C datatype long double.

Mapping C datatypes to MRM datatypes

Table 7 on page 156 defines the datatype mappings for C structures.

Notes:

1. Long Double is outside the scope of the importer.

Table 7. C datatypes and their default settings in the MRM

C datatype	MRM logical type	Physical type	Length	Sign	String justification	Repeat
Long	Integer	Integer	4	Signed		
Char	String	Fixed Length	1			
Char[10]	String	Fixed Length	10		Left justify	
Char[10][3]	String	Fixed Length	3		Left justify	10
Char[10][3][6]	String	Fixed Length	6		Left justify	30
Int	Integer	Integer	4	Signed		
Int[2]	Integer	Integer	4	Signed		2
Int[2][3]	Integer	Integer	4	Signed		6
Unsigned Int	Integer	Integer	4	Unsigned		
Unsigned Short	Integer	Integer	2	Unsigned		
Float	Float	Float	4			
Double	Float	Float	8			
Short	Integer	Integer	2	Signed		
Unsigned char	Integer	Integer	1			
Unsigned char[2]	Binary	Binary	2			
`(#define)BOOL int`	Boolean	Boolean				
`(#define)Boolean`	Boolean	Boolean				

Mapping COBOL datatypes to MRM datatypes

Columns 1 to 5 in Table 8 on page 158 describe some examples of COBOL data definitions. Columns 6 to 12 describe the equivalent data mappings used to store these definitions in the MRM.

Notes:

1. Column 5 (of 12) Internal representation assumes an ASCII Big Endian code page.
2. Column 12 (of 12) **Jst.** indicates the justification of the datatype.
3. The following datatypes are outside the scope of the importer:
   - Binary (10 to 18 digits)
   - External floating point
   - DBCS

Table 8. COBOL datatypes and their default settings in the MRM

COBOL datatype	Permitted symbols	PICTURE and USAGE and optional SIGN clause	Value	Internal representation	MRM Logical type	Physical type	Length in bytes	Sign	Virtual dec. point	Pad. char.	Jst.
External decimal (Zoned Decimal)	9 P S V				If > 9 digits: float If a fraction: float Else: integer	extended decimal	=num of digits. If sign is separate, add 1				
		PIC S9999 DISPLAY	+1234	31 32 33 34	Integer	extended decimal	4	Y including trailing	0		
			-1234	31 32 33 74							
			1234	31 32 33 34							
		PIC 9999 DISPLAY	1234	31 32 33 34	integer	extended decimal	4	N	0		
		PIC 99V99 DISPLAY	1234	31 32 33 34	float	extended decimal	4	N	2		
		PIC S9999 DISPLAY SIGN LEADING	+1234	31 32 33 34	integer	extended decimal	4	Y including leading	0		
			-1234	71 32 33 34							
		PIC S9999 DISPLAY SIGN LEADING SEPARATE	+1234	2B 31 32 33 34	integer	extended decimal	5	Y separate leading	0		
			-1234	2D 31 32 33 34							
		PIC S9999 DISPLAY SIGN TRAILING SEPARATE	+1234	31 32 33 34 2B	integer	extended decimal	5	Y separate trailing	0		
			-1234	31 32 33 34 2D							
Binary	9 P S V				integer	integer	if <5 decimal digits, 2 bytes If 5 thru 9 digits, 4 bytes				
		PIC S9999 BINARY or COMP or COMP-4 or COMP-5	+1234	04 D2	integer	integer	2	Y			
			-1234	FB 2E							
		PIC 9999 BINARY or COMP or COMP-4 or COMP-5	1234	04 D2	integer	integer	2	N			

Table 8. COBOL datatypes and their default settings in the MRM (continued)

COBOL datatype	Permitted symbols	PICTURE and USAGE and optional SIGN clause	Value	Internal representation	MRM Logical type	Physical type	Length in bytes	Sign	Virtual dec. point	Pad. char.	Jst.
Internal Decimal (Packed Decimal)	9 P S V				If > 9 digits, float If a fraction, float Else integer		Rounded down result of (Num of digits+2)/2				
		PIC S9999 PACKED-DECIMAL or COMP-3	+1234	01 23 4C	integer	packed decimal	3	Y			
			-1234	01 23 4D							
		PIC 9999 PACKED-DECIMAL or COMP-3	1234	01 23 4F	integer	packed decimal	3	N			
Internal floating point											
	no PIC clause	COMP-1	+1234	44 9A 40 00	float	float	4	Y			
			-1234	C4 9A 40 00							
	no PIC clause	COMP-2	+1234	40 93 48 00 00 00 00 00	float	float	8	Y			
			-1234	C0 93 48 00 00 00 00 00							
Alphabetic	A	PIC A(3) DISPLAY	ABC	41 42 43	string	fixed length	3 chars			space	L
Alphanumeric	X	PIC XXXX DISPLAY	DEF	44 45 46 20	string	fixed length	4 chars			space	default L
		PIC X(4) JUSTIFIED RIGHT	DEF	20 44 45 46	string	fixed length	4 chars			space	R
		JUST RIGHT									
Alphanumeric edited	X B 0 9 /	PIC BX/9 DISPLAY	A/3	20 41 2F 33	string	fixed length	4 chars			space	L
Numeric edited	B P V Z 9 0/ comma symbol period symbol + - CR DB * $						length in chars=sum of chars in PIC string excluding V			space	default R
		PIC 9999	0123	30 31 32 33	string	fixed length	4			space	R
		PIC ZZZ9	123	20 31 32 33							

Table 8. COBOL datatypes and their default settings in the MRM (continued)

COBOL datatype	Permitted symbols	PICTURE and USAGE and optional SIGN clause	Value	Internal representation	MRM Logical type	Physical type	Length in bytes	Sign	Virtual dec. point	Pad. char.	Jst.
		PIC $$Z9	$123	24 31 32 33							
			$12	20 24 31 32							
		PIC 999V9	1234	31 32 33 34							
		PIC ZZZV9	1234	31 32 33 34							
		PIC $ZZZ,ZZ9V.99	$123,456.78	24 31 32 33 2C 34 35 36 2E 37 38	string	fixed length	11			space	R

Part 3. Appendixes

Appendix. Notices

This information was developed for products and services offered in the United States. IBM may not offer the products, services, or features discussed in this information in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this information. The furnishing of this information does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the information. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this information at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs

and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM United Kingdom Laboratories,
Mail Point 151,
Hursley Park,
Winchester,
Hampshire,
England
SO21 2JN.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Programming License Agreement, or any equivalent agreement between us.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Trademarks

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

AIX	AS/400	DB2
DB2 Universal Database	IBM	IBMLink
MQSeries	OS/390	SupportPac

Lotus is a trademark of Lotus Development Corporation in the United States, other countries, or both.

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

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

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

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

Glossary of terms and abbreviations

This glossary defines MQSeries Integrator terms and abbreviations used in this book. If you do not find the term you are looking for, see the index or the *IBM Dictionary of Computing*, New York: McGraw-Hill, 1994.

This glossary includes terms and definitions from the *American National Dictionary for Information Systems*, ANSI X3.172-1990, copyright 1990 by the American National Standards Institute. Copies may be ordered from the American National Standards Institute, 11 West 42 Street, New York, New York 10036. Definitions are identified by the symbol (A) after the definition.

A

Access Control List (ACL). The list of principals that have explicit permissions (to publish, to subscribe to, and to request persistent delivery of a publication message) against a topic in the topic tree. The ACLs define the implementation of topic-based security.

ACL. Access Control List.

AMI. Application Messaging Interface.

Application Messaging Interface (AMI). The programming interface provided by MQSeries that defines a high level interface to message queuing services. See also *MQI* and *JMS*.

B

blob. Binary Large OBject. A block of bytes of data (for example, the body of a message) that has no discernible meaning, but is treated as one solid entity that cannot be interpreted. Also written as BLOB.

broker. See *message broker*.

broker domain. A collection of brokers that share a common configuration, together with the single Configuration Manager that controls them.

C

callback function. See *implementation function*.

category. An optional grouping of messages that are related in some way. For example, messages that relate to a particular application.

check in. The Control Center action that stores a new or updated resource in the configuration or message respository.

check out. The Control Center action that extracts and locks a resource from the configuration or message respository for local modification by a user. Resources from the two repositories can only be worked on when they are checked out by an authorized user, but can be viewed (read only) without being checked out.

collective. A hyperconnected (totally connected) set of brokers forming part of a multi-broker network for publish/subscribe applications.

configuration. In the broker domain, the brokers, execution groups, message flows and message sets assigned to them, topics and access control specifications.

Configuration Manager. A component of MQSeries Integrator that acts as the interface between the configuration repository and an executing set of brokers. It provides brokers with their initial configuration, and updates them with any subsequent changes. It maintains the broker domain configuration.

configuration repository. Persistent storage for broker configuration and topology definition.

connector. See *message processing node connector*.

content-based filter. An expression that is applied to the content of a message to determine how the message is to be processed.

context tag. A tag that is applied to an element within a message to enable that element to be treated differently in different contexts. For example, an element could be mandatory in one context and optional in another.

Control Center. The graphical interface that provides facilities for defining, configuring, deploying, and monitoring resources of the MQSeries Integrator network.

D

datagram. The simplest form of message that MQSeries supports. Also known as *send-and-forget*. This type of message does not require a reply. Compare with *request/reply*.

debugger. A facility on the *Message Flows* view in the Control Center that enables message flows to be debugged.

deploy. Make operational the configuration and topology of the broker domain.

destination list. A list of internal and external destinations to which a message is sent. These can be

nodes within a message flow (for example, when using the RouteToLabel and Label nodes) or MQSeries queues (when the list is examined by an MQOutput node to determine the final target for the message).

distribution list. A list of MQSeries queues to which a message can be put using a single statement.

Document Type Definition (DTD). The rules that specify the structure for a particular class of SGML or XML documents. The DTD defines the structure with elements, attributes, and notations, and it establishes constraints for how each element, attribute, and notation can be used within the particular class of documents. A DTD is analogous to a database schema in that the DTD completely describes the structure for a particular markup language.

DTD. Document Type Definition

E

e-business. A term describing the commercial use of the Internet and World Wide Web to conduct business (short for electronic-business).

element. A unit of data within a message that has business meaning, for example, street name

element qualifier. See *context tag*.

ESQL. Extended SQL. A specialized set of SQL statements based on regular SQL, but extended with statements that provide specialized functions unique to MQSeries Integrator.

exception list. A list of exceptions that have been generated during the processing of a message, with supporting information.

execution group. A named grouping of message flows that have been assigned to a broker. The broker is guaranteed to enforce some degree of isolation between message flows in distinct execution groups by ensuring that they execute in separate address spaces, or as unique processes.

Extensible Markup Language (XML). A W3C standard for the representation of data.

external reference. A reference within a message set to a component that has been defined outside the current message set. For example, an integer that defines the length of a string element might be defined in one message set but used in several message sets.

F

field reference. A sequence of period-separated values that identify a specific field (which might be a structure) within a message tree. An example of a field reference might be something like `Body.Invoice.InvoiceNo`.

filter. An expression that is applied to the content of a message to determine how the message is to be processed.

format. A format defines the internal structure of a message, in terms of the fields and order of those fields. A format can be self-defining, in which case the message is interpreted dynamically when read.

G

graphical user interface (GUI). An interface to a software product that is graphical rather than textual. It refers to window-based operational characteristics.

I

implementation function. Function written by a third-party developer for a plug-in node or parser. Also known as a *callback function*.

input node. A message flow node that represents a source of messages for the message flow.

installation mode. The installation mode can be Full, Custom, or Broker only. The mode defines the components of the product installed by the installation process on Windows NT systems.

J

Java™ Database Connectivity (JDBC). An application programming interface that has the same characteristics as **ODBC** but is specifically designed for use by Java database applications.

Java Development Kit (JDK). A software package that can be used to write, compile, debug, and run Java applets and applications.

Java Message Service (JMS). An application programming interface that provides Java language functions for handling messages.

Java Runtime Environment (JRE). A subset of the Java Development Kit (JDK) that contains the core executables and files that constitute the standard Java platform. The JRE includes the Java Virtual Machine, core classes and supporting files.

JDBC™. Java Database Connectivity.

JDK™. Java Development Kit.

JMS. Java Message Service. See also *AMI* and *MQI*.

JRE. Java Runtime Environment.

L

local error log. A generic term that refers to the logs to which MQSeries Integrator writes records on the local system. On Windows NT, this is the Event log. On UNIX systems, this is the syslog. See also *system log*. Note that MQSeries records many events in the log that are not errors, but information about events that occur during operation, for example, successful deployment of a configuration.

M

message broker. A set of execution processes hosting one or more message flows.

messages. Entities exchanged between a broker and its clients.

message dictionary. A repository for (predefined) message type specifications.

message domain. The value that determines how the message is interpreted (parsed). The following domains are recognized:

- MRM, which identifies messages defined using the Control Center
- NEONMSG[3], which identifies messages created using the NEONFORMATTER user interfaces.
- XML, which identifies messages that are self-defining
- BLOB, which identifies messages that are undefined

You can also create your own message domains: if you do so, you must supply your own message parser.

message flow. A directed graph that represents the set of activities performed on a message or event as it passes through a broker. A message flow consists of a set of message processing nodes and message processing node connectors.

message flow component. See *message flow*.

message parser. A program that interprets a message bitstream.

message processing node. A node in the message flow, representing a well defined processing stage. A message processing node can be one of several primitive types or can represent a subflow.

message processing node connector. An entity that connects the output terminal of one message processing node to the input terminal of another. A message processing node connector represents the flow of control and data between two message flow nodes.

message queue interface (MQI). The programming interface provided by MQSeries queue managers. The programming interface allows application programs to access message queuing services. See also *AMI* and *JMS*.

message repository. A database holding message template definitions.

message repository manager (MRM). A component of the Configuration Manager that handles message definition and control. A message defined to the MRM has a message domain set to MRM.

message set. A grouping of related messages.

message template. A named and managed entity that represents the format of a particular message. Message templates represent a business asset of an organization.

message type. The logical structure of the data within a message. For example, the number and location of character strings.

metadata. Data that describes the characteristic of stored data.

MQe. MQSeries Everyplace.

MQI. Message queue interface.

MQIsdp. MQSeries Integrator SCADA device protocol. A lightweight publish/subscribe protocol flowing over TCP/IP.

MQRFH. An architected message header that is used to provide metadata for the processing of a message. This header is supported by MQSeries Publish/Subscribe.

MQRFH2. An extended version of MQRFH, providing enhanced function in message processing.

MQSeries Everyplace. A generally available MQSeries product that provides proven MQSeries reliability and security in a mobile environment.

MRM. Message Repository Manager.

multilevel wildcard. A wildcard that can be specified in subscriptions to match any number of levels in a topic.

N

node. See *message processing node*.

O

ODBC. Open Database Connectivity.

3. The message domain NEON is also recognized for compatibility with previous releases.

Open Database Connectivity. A standard application programming interface (API) for accessing data in both relational and non-relational database management systems. Using this API, database applications can access data stored in database management systems on a variety of computers even if each database management system uses a different data storage format and programming interface. ODBC is based on the call level interface (CLI) specification of the X/Open SQL Access Group.

output node. A message processing node that represents a point at which messages flow out of the message flow.

P

plug-in. An extension to the broker, written by a third-party developer, to provide a new message processing node or message parser in addition to those supplied with the product. See also *implementation function* and *utility function*.

point-to-point. Style of messaging application in which the sending application knows the destination of the message. Compare with *publish/subscribe*.

POSIX. Portable Operating System Interface For Computer Environments. An IEEE standard for computer operating systems (for example, AIX and Sun Solaris).

predefined message. A message with a structure that is defined before the message is created or referenced. Compare with *self-defining message*.

primitive. A message processing node that is supplied with the product.

principal. An individual user ID (for example, a log-in ID) or a group. A group can contain individual user IDs and other groups, to the level of nesting supported by the underlying facility.

property. One of a set of characteristics that define the values and behaviors of objects in the Control Center. For example, message processing nodes and deployed message flows have properties.

publication node. An end point of a specific path through a message flow to which a client application subscribes. A publication node has an attribute, subscription point. If this is not specified, the publication node represents the default subscription point for the message flow.

publish/subscribe. Style of messaging application in which the providers of information (publishers) are decoupled from the consumers of that information (subscribers) using a broker. Compare with *point-to-point*. See also *topic*.

publisher. An application that makes information about a specified topic available to a broker in a publish/subscribe system.

Q

queue. An MQSeries object. Message queuing applications can put messages on, and get messages from, a queue. A queue is owned and maintained by a queue manager. Local queues can contain a list of messages waiting to be processed. Queues of other types cannot contain messages: they point to other queues, or can be used as models for dynamic queues.

queue manager. A system program that provides queuing services to applications. It provides an application programming interface (the MQI) so that programs can access messages on the queues that the queue manager owns.

R

retained publication. A published message that is kept at the broker for propagation to clients that subscribe at some point in the future.

request/reply. Type of messaging application in which a request message is used to request a reply from another application. Compare with *datagram*.

rule. A rule is a definition of a process, or set of processes, applied to a message on receipt by the broker. Rules are defined on a message format basis, so any message of a particular format will be subjected to the same set of rules.

S

SCADA. Supervisory, Control, And Data Acquisition.

self-defining message. A message that defines its structure within its content. For example, a message coded in XML is self-defining. Compare with *pre-defined message*.

send and forget. See *datagram*.

setup type. The definition of the type of installation requested on Windows NT systems. This can be one of **Full**, **Broker only**, or **Custom**.

shared. All configuration data that is shared by users of the Control Center. This data is not operational until it has been deployed.

signature. The definition of the external characteristics of a message processing node.

single-level wildcard. A wildcard that can be specified in subscriptions to match a single level in a topic.

stream. A method of topic partitioning used by MQSeries Publish/Subscribe applications.

subscriber. An application that requests information about a specified topic from a publish/subscribe broker.

subscription. Information held within a publication node, that records the details of a subscriber application, including the identity of the queue on which that subscriber wants to receive relevant publications.

subscription filter. A predicate that specifies a subset of messages to be delivered to a particular subscriber.

subscription point. An attribute of a publication node that differentiates it from other publication nodes on the same message flow and therefore represents a specific path through the message flow. An unnamed publication node (that is, one without a specific subscription point) is known as the default publication node.

Supervisory, Control, And Data Acquisition. A broad term, used to describe any form of remote telemetry system used for gathering data from remote sensor devices (for example, flow rate meters on an oil pipeline) and for the near real time control of remote equipment (for example, pipeline valves).

system log. A generic term used in the MQSeries Integrator messages (BIPxxx) that refers to the local error logs to which records are written on the local system. On Windows NT, this is the Event log. On UNIX systems, this is the syslog. See also *local error log*.

T

terminal. The point at which one node in a message flow is connected to another node. Terminals enable you to control the route that a message takes, depending whether the operation performed by a node on that message is successful.

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

topic based subscription. A subscription specified by a subscribing application that includes a topic for filtering of publications.

topic security. The use of ACLs applied to one or more topics to control subscriber access to published messages.

topology. In the broker domain, the brokers, collectives, and connections between them.

transform. A defined way in which a message of one format is converted into one or more messages of another format.

U

Uniform Resource Identifier. The generic set of all names and addresses that refer to World Wide Web resources.

Uniform Resource Locator. A specific form of URI that identifies the address of an item on the World Wide Web. It includes the protocol followed by the fully qualified domain name (sometimes called the host name) and the request. The Web server typically maps the request portion of the URL to a path and file name. Also known as Universal Resource Locator.

URI. Uniform Resource Identifier

URL. Uniform Resource Locator

User Name Server. The MQSeries Integrator component that interfaces with operating system facilities to determine valid users and groups.

utility function. Function provided by MQSeries Integrator for the benefit of third-party developers writing plug-in nodes or parsers.

W

warehouse. A persistent, historical datastore for events (or messages). The **Warehouse** node within a message flow supports the recording of information in a database for subsequent retrieval and processing by other applications.

wildcard. A character that can be specified in subscriptions to match a range of topics. See also *multilevel wildcard* and *single-level wildcard*.

wire format. This describes the physical representation of a message within the bit-stream.

W3C. World Wide Web Consortium. An international industry consortium set up to develop common protocols to promote evolution and interoperability of the World Wide Web.

X

XML. Extensible Markup Language.

Glossary

Bibliography

This section describes the documentation available for all current MQSeries Integrator products.

MQSeries Integrator Version 2.0.2 cross-platform publications

The MQSeries Integrator cross-platform publications are:

- *MQSeries Integrator Introduction and Planning*, GC34-5599
- *MQSeries Integrator Using the Control Center*, GC34-5602
- *MQSeries Integrator Messages*, GC34-5601
- *MQSeries Integrator Programming Guide*, SC34-5603
- *MQSeries Integrator Administration Guide*, SC34-5792
- *MQSeries Integrator ESQL Reference*, SC34-5923

These books are all available in hardcopy.

You can order publications from the IBMLink™ Web site at:

`http://www.ibm.com/ibmlink`

In the United States, you can also order publications by dialing 1-800-879-2755.

In Canada, you can order publications by dialing 1-800-IBM-4YOU (1-800-426-4968).

For further information about ordering publications contact your IBM authorized dealer or marketing representative.

MQSeries Integrator Version 2.0.2 platform-specific publications

Each MQSeries Integrator product provides one platform-specific installation guide, which is supplied in hardcopy.

MQSeries Integrator for AIX Version 2.0.2

MQSeries Integrator for AIX Installation Guide, GC34-5841

MQSeries Integrator for HP-UX Version 2.0.2

MQSeries Integrator for HP-UX Installation Guide, GC34-5907

MQSeries Integrator for Sun Solaris Version 2.0.2

MQSeries Integrator for Sun Solaris Installation Guide, GC34-5842

MQSeries Integrator for Windows NT Version 2.0.2

MQSeries Integrator for Windows NT Installation Guide, GC34-5600

MQSeries Everyplace publications

If you intend to connect MQSeries Everyplace applications to message flows that include the MQSeries Everyplace message flow nodes, you will find the following publications useful:

- *MQSeries Everyplace for Multiplatforms Version 1.1 Introduction*, GC34-5843
- *MQSeries Everyplace for Multiplatforms Version 1.1 Programming Guide*, SC34-5845
- *MQSeries Everyplace for Multiplatforms Version 1.1 Programming Reference*, SC34-5846
- *MQSeries Everyplace for Multiplatforms Version 1.1 Native Client Information*, SC34-5880

You can find these books on the MQSeries Web site (see “MQSeries information available on the Internet” on page 173). Translated versions of these books are also available in some languages from the same Web site.

NEONRules and NEONFormatter Support for MQSeries Integrator publications

The following publications are supplied on the product CD in PDF format, and are installed with the Documentation component.

- *NEONRules and NEONFormatter Support for MQSeries Integrator User's Guide*
- *NEONRules and NEONFormatter Support for MQSeries Integrator System Management Guide*
- *NEONRules and NEONFormatter Support for MQSeries Integrator Programming Reference for NEONRules*

Bibliography

- *NEONRules and NEONFormatter Support for MQSeries Integrator Programming Reference for NEONFormatter*
- *NEONRules and NEONFormatter Support for MQSeries Integrator Application Development Guide*

These books are provided in US English only.

Softcopy books

All the MQSeries Integrator books are available in softcopy formats.

Portable Document Format (PDF)

All books in the MQSeries Integrator library are supplied in US English only in a searchable PDF library on the product CD.

You can install the library as follows:

- On AIX, invoke `install -d` and select the documentation fileset. After installation, run the command `mqsidocs`. This launches Acrobat Reader and opens the PDF package.
- On HP-UX, invoke `swinstall -d` and select `MQSI-DOCS` from the menu. After installation, run the command `mqsidocs`. This launches Acrobat Reader and opens the PDF package.
- On Sun Solaris, invoke `pkgadd -d` and select `mqsi-docs` from the menu. After installation, run the command `mqsidocs`. This launches Acrobat Reader and opens the PDF package.
- On Windows NT, select the `Online Documentation` component on a custom installation, or do a full installation. After installation, select *Start—>Programs—>IBM MQSeries Integrator 2.0—>Documentation*.

In addition, PDF files for books that have been translated are installed into the location `mqsi_root/bin/book/pdf/<locale>` (on UNIX) or `mqsi_root\bin\book\pdf\<locale>` (on Windows NT) where `<locale>` is one of the following:

- **de_DE** for German
- **en_US** for US English
- **es_ES** for Spanish
- **fr_FR** for French
- **it_IT** for Italian
- **ja_JP** for Japanese
- **ko_KR** for Korean
- **pt_BR** for Brazilian Portuguese
- **zh_CN** for Simplified Chinese
- **zh_TW** for Traditional Chinese

An index file (in HTML format) that provides a link to each book is supplied for each language. For example, the French index file is called `indexfr.htm`. The files are stored in the following directory:

- On UNIX, `<mqsi_root>/docs/`
- On Windows NT, `<mqsi_root>\bin\book`

Each index file has an entry for every book: if a particular book has not been translated into the appropriate language for that index file, a link to the English PDF is included. You can use any Web browser to view the index file. On Windows NT, you can also access the index file through the *Start* menu.

The PDF file names for the English books are shown in Table 9.

Table 9. File names of MQSeries Integrator book PDFs

Book title	File name
MQSeries Integrator for AIX Installation Guide	bipaac04.pdf
MQSeries Integrator for HP-UX Installation Guide	bipcac00.pdf
MQSeries Integrator for Sun Solaris Installation Guide	bip7ac03.pdf
MQSeries Integrator for Windows NT Installation Guide	bipyac03.pdf
MQSeries Integrator Introduction and Planning	bipyab02.pdf
MQSeries Integrator Administration Guide	bipyag04.pdf
MQSeries Integrator Using the Control Center	bipyar03.pdf
MQSeries Integrator ESQL Reference	bipyae00.pdf
MQSeries Integrator Programming Guide	bipyal02.pdf
MQSeries Integrator Messages	bipyao02.pdf

The fifth character of the file name indicates the language of the book (**a** indicates US English). You can deduce the file names of translated books by using the following substitutions for the fifth character:

- **g** for German
- **s** for Spanish
- **f** for French
- **i** for Italian
- **j** for Japanese
- **k** for Korean

- **b** for Brazilian Portuguese
- **z** for Simplified Chinese
- **t** for Traditional Chinese

PDF files can be viewed and printed using the Adobe Acrobat Reader.

If you cut and paste examples of commands from PDF files to a command line for execution, you must check that the content is correct before you press `Enter`. Some characters might be corrupted by local system and font settings.

If you need to obtain the Adobe Acrobat Reader, or would like up-to-date information about the platforms on which the Acrobat Reader is supported, visit the Adobe Systems Inc. Web site at:

`http://www.adobe.com/`

PDF versions of all current MQSeries Integrator books are also available from the MQSeries product family Web site at:

`http://www.ibm.com/software/mqseries/`

MQSeries information available on the Internet

The MQSeries product family Web site is at:

`http://www.ibm.com/software/mqseries/`

By following links from this Web site you can:

- Obtain latest information about the MQSeries product family.
- Access the MQSeries books in HTML and PDF formats.
- Obtain information about complementary offerings by following these links:
  - IBM Business Partners
  - Partner Offerings (within *Related links*)
- Download an MQSeries SupportPac.

Index

A

B

C

D

E

F

G

I

L

M

N

O

P

R

S

T

U

W

X

Z

Sending your comments to IBM

If you especially like or dislike anything about this book, please use one of the methods listed below to send your comments to IBM.

Feel free to comment on what you regard as specific errors or omissions, and on the accuracy, organization, subject matter, or completeness of this book.

Please limit your comments to the information in this book and the way in which the information is presented.

To make comments about the functions of IBM products or systems, talk to your IBM representative or to your IBM authorized remarketer.

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

You can send your comments to IBM in any of the following ways:

- By mail, to this address:

  User Technologies Department (MP095)
  IBM United Kingdom Laboratories
  Hursley Park
  WINCHESTER,
  Hampshire
  SO21 2JN
  United Kingdom
- By fax:
  - From outside the U.K., after your international access code use 44–1962–842327
  - From within the U.K., use 01962–842327
- Electronically, use the appropriate network ID:
  - IBM Mail Exchange: GBIBM2Q9 at IBMMAIL
  - IBMLink: HURSLEY(IDRCF)
  - Internet: idrcf@hursley.ibm.com

Whichever method you use, ensure that you include:

- The publication title and order number
- The topic to which your comment applies
- Your name and address/telephone number/fax number/network ID.

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

SC34-5602-03