MQSeries® Integrator for AIX®

Installation Guide

Version 2.0.1

GC34-5841-03

MQSeries® Integrator for AIX®

Installation Guide

Version 2.0.1

GC34-5841-03

Note!

Before using this information and the product it supports, be sure to read the general information under "Appendix E. Notices" on page 73.

Third edition (December 2000)

This edition applies to IBM MQSeries Integrator for AIX Version 2.0.1 and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

Figures

Tables

About this book

This book provides an overview of IBM® MQSeries Integrator Version 2.0.1. It explains how to plan for, install, and verify installation of the product.

"Chapter 1. Installation overview" on page 1 provides a brief installation overview of MQSeries Integrator for AIX® Version 2.0.1 at a high level.

"Chapter 2. Planning for installation" on page 3 describes the preparation you need to complete prior to product installation.

"Chapter 3. Installing MQSeries Integrator" on page 13 provides detailed installation information for MQSeries Integrator for AIX Version 2.0.1.

"Chapter 4. Configuring a broker domain" on page 21 explains how you configure a basic broker domain.

"Chapter 5. Verifying your installation" on page 41 explains how to deploy your broker network and verify its operation using supplied verification programs.

The appendixes cover the configuration established by the default installation options, guidelines for preparing to use the NEONFormatter and NEONRules nodes, and details of servicing and removing the product.

For information about installing Windows NT®, refer to the *MQSeries Integrator for Windows NT Version 2.0.1 Installation Guide*

For further information about the product, and planning for its use, refer to the *MQSeries Integrator Version 2.0.1 Introduction and Planning* book.

For details of administrative tasks, including configuration and problem determination, see the *MQSeries Integrator Version 2.0.1 Administration Guide*.

Who this book is for

This book is for administrators of systems on which MQSeries Integrator Version 2.0.1 components will be installed and tested.

About this book

What you need to know to understand this book

To understand this book, you need to be familiar with the system facilities of AIX. You also need to be familiar with the administration facilities of MQSeries® for AIX V5.1.

The *MQSeries for AIX V5.1 Quick Beginnings*, *MQSeries for Windows NT V5.1 Quick Beginnings*, *MQSeries System Administration*, and *MQSeries Integrator Version 2.0.1 Administration Guide* books provide useful reference information for installation and post-installation tasks.

You need to be familiar with the database product that you intend to use to support your MQSeries Integrator for AIX Version 2.0.1 components.

Terms used in this book

All references to MQSeries Integrator are to MQSeries Integrator Version 2.0.1 unless otherwise stated.

Terms are defined in "Glossary of terms and abbreviations" on page 77.

The book uses the following shortened names:

- MQSeries: a general term for IBM MQSeries messaging products.
- MQSeries Publish/Subscribe: the MQSeries Publish/Subscribe SupportPac™; available on the Internet for several MQSeries server operating systems (the Internet URL is given in "MQSeries information available on the Internet" on page xiii).
- DB2®; a general term to encompass IBM DB2 Universal Database®; Enterprise Edition, Connect Enterprise Edition, and Extended Enterprise Edition.

Where to find more information

Becoming familiar with the MQSeries Integrator library will help you accomplish MQSeries Integrator tasks quickly. The library covers planning, installation, administration, and client application tasks.

The library also contains references to complementary product libraries, including the MQSeries Family library.

Note: If you cut and paste examples of commands from the Portable Document File (PDF) of a book, to a command line for execution, you must check that the content is correct before you press the Enter key. Some characters might be corrupted by local system and font settings.

MQSeries Integrator publications

The following books make up the MQSeries Integrator Version 2.0.1 library:

- *IBM MQSeries Integrator Version 2.0.1 Introduction and Planning, GC34-5599*
- *IBM MQSeries Integrator for AIX Version 2.0.1 Installation Guide, GC34-5841* (this book)
- *IBM MQSeries Integrator for Windows NT Version 2.0.1 Installation Guide, GC34-5600*
- *IBM MQSeries Integrator for Sun Solaris Version 2.0.1 Installation Guide, GC34-5842*
- *IBM MQSeries Integrator Version 2.0.1 Messages, GC34-5601*
- *IBM MQSeries Integrator Version 2.0.1 Using the Control Center, SC34-5602*
- *IBM MQSeries Integrator Version 2.0.1 Programming Guide, SC34-5603*
- *IBM MQSeries Integrator Version 2.0.1 Administration Guide, SC34-5792*

This book you are reading is provided in hardcopy with the product. The *MQSeries Integrator Introduction and Planning* book is also available in hardcopy.

All books in the MQSeries Integrator Version 2.0.1 library are provided in softcopy, in Adobe Portable Document Format (PDF) in a searchable PDF library. You can:

- Install the library (by doing a full installation or by specifying the `Documentation` component on a custom installation).
- Access the library directly from the `mqsi_aix_documentation` subdirectory (the `mqsi_aix` subdirectory on Windows NT) under the root directory on the supplementary CD-ROM without installing them. .

The MQSeries Integrator Version 1.1 publications are also supplied as PDFs and can be installed with MQSeries Integrator Version 2.0.1 (the `Documentation` component). They can also be retrieved from the MQSeries Web site given in “MQSeries information available on the Internet” on page xiii.

- *IBM MQSeries Integrator Version 1.1 Installation and Configuration Guide, GC34-5503*
- *IBM MQSeries Integrator Version 1.1 User’s Guide, GC34-5504*
- *IBM MQSeries Integrator Version 1.1 System Management Guide, SC34-5505*
- *IBM MQSeries Integrator Version 1.1 Programming Reference for NEONRules, SC34-5506*
- *IBM MQSeries Integrator Version 1.1 Programming Reference for NEONFormatter, SC34-5507*
- *IBM MQSeries Integrator Version 1.1 Application Development Guide, SC34-5508*

MQSeries family publications

You can read PDFs using Adobe Acrobat Reader or in a Web browser (with Acrobat Reader as a plug-in). Version 4 is required. You can also print your own copies of these books.

You can download a free copy of Acrobat Reader from the Adobe Web site at:

`http://www.adobe.com`

MQSeries publications

The following books are referred to in this book to point you to the information you need to complete MQSeries Messaging product tasks as part of MQSeries Integrator tasks.

For AIX installation tasks you might need:

- *IBM MQSeries for AIX V5.1 Quick Beginnings, GC33-1867*.

  This book is included, in hardcopy, in the MQSeries Integrator package.

  It provides detailed planning and installation guidance.

For Windows NT installation tasks you might need:

- *IBM MQSeries for Windows NT V5.1 Quick Beginnings, GC34-5389*.

  It provides detailed planning and installation guidance.

For planning and configuration tasks you might need:

- *IBM MQSeries Command Reference, SC33-1369*.

  This book contains the syntax of the MQSC commands.
- *IBM MQSeries System Administration, SC33-1873*.

  This book supports day-to-day management of local and remote MQSeries objects.
- *IBM MQSeries Clients, GC33-1632*.

  This book describes how to install, configure, use, and manage MQSeries clients.
- *IBM MQSeries Intercommunication, SC33-1872*.

  This book describes MQSeries Intercommunication between different platforms.

For a complete list of MQSeries product publications, refer to the information on the MQSeries Web site (given in “MQSeries information available on the Internet” on page xiii).

MQSeries Publish/Subscribe publications

If you have installed MQSeries Publish/Subscribe, and plan to migrate to MQSeries Integrator Version 2, or to establish a mixed broker network, refer to:

- *IBM MQSeries Publish/Subscribe User’s Guide, GC34-5269*

This book and the MQSeries Publish/Subscribe SDK package are avaliable on the MQSeries Web site (given in “MQSeries information available on the Internet”).

MQSeries Workflow publications

The MQSeries Workflow product has a comprehensive library. Refer to the following book for introductory information, and for details about other product publications:

- *IBM MQSeries Workflow Concepts and Architecture, GH12-6285*

For a complete list of MQSeries Workflow product publications, refer to the information on the MQSeries Web site (given in “MQSeries information available on the Internet”).

DB2 publications

The following DB2 publications are referenced in this book.

- *IBM DB2 Quick Beginnings, GC09-2835*
- *IBM DB2 Message Reference, GC09-2846*
- *IBM DB2 TroubleShooting Guide, SI0J-8169*

You can download these publications from the DB2 Web site at

`http://www.ibm.com/software/data/db2`

MQSeries information available on the Internet

The MQSeries Business Solution, of which MQSeries Integrator is a part, has a Web site at:

`http://www.ibm.com/software/ts/mqseries`

By following links from this Web site you can:

- Obtain the latest information about all MQSeries family products.
- Access all the books for the MQSeries family products.
- Download MQSeries SupportPacs.

MQSeries family publications

Summary of changes

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

Changes for this edition (GC34-5841-03)

The changes are summarized below:

- Migration information has been removed from the book. Refer to the *MQSeries Integrator Version 2.0.1 Administration Guide* for guidance on migration.
- The information about verifying your installation has been removed from “Chapter 4. Configuring a broker domain” on page 21 and put into a new separate chapter. See “Chapter 5. Verifying your installation” on page 41
- The information on using NEONFormatter and NEONRules nodes has been moved to the *MQSeries Integrator Version 2.0.1 Administration Guide*.
- Minor editorial changes have also been made throughout the book.

Changes for the previous edition (GC34-5841-02)

The changes are summarized below:

- All installation information for MQSeries Integrator for Windows NT was removed. Refer to the *MQSeries Integrator for Windows NT Version 2.0.1 Installation Guide* for information on planning, installation, configuration, and verification of MQSeries Integrator for Windows NT.
- Information on migration from MQSeries Integrator Version 1 and MQSeries Publish/Subscribe was updated. See “Migration considerations” on page 11
- Information about the running the Postcard application was added. See “Running the Postcard application” on page 49
- Broker database support using Oracle8 information was added. See “Appendix B. Setting up an Oracle8 broker database on MQSeries Integrator” on page 61.

Changes

Chapter 1. Installation overview

MQSeries Integrator provides comprehensive facilities to create, configure, and manage a broker domain on AIX.

On AIX, a broker domain consists of:

- One or more message brokers that support diverse applications exchanging information in many formats.

  The message brokers work with an optional component, the User Name Server, that provides access control for publish/subscribe applications. These components are installed together to provide the runtime support.
- On Windows NT, the two components that provide configuration and management support are the Configuration Manager and the Control Center.
  - The Configuration Manager owns and controls the configuration of the broker domain, the procedures (message flows or business rules) that are to operate within your brokers, and the definition of message formats that can be manipulated by those procedures.
  - The Control Center is a sophisticated graphical interface that allows controlled access to the resources defined to the Configuration Manager to create, change, delete, and deploy those resources, and to monitor and manage their operational status.

A full description of the components of MQSeries Integrator, the facilities they provide, and the formats of information supported, is provided in *MQSeries Integrator Introduction and Planning Guide*.

Installing the runtime support

The runtime support (the message broker and the User Name Server) must be installed on AIX. You can install and configure one or more message brokers on one or more AIX systems, subject to your licence agreement.

You must install and configure one User Name Server on AIX.

“Chapter 2. Planning for installation” on page 3 provides details about:

- The hardware and software prerequisites for MQSeries Integrator runtime support
- The database support that is required by the brokers

“Chapter 3. Installing MQSeries Integrator” on page 13 tells you how to install the runtime support. Before you install the configuration support, you are recommended to install the runtime support, using the information presented in “Chapter 2. Planning for installation” on page 3 and “Chapter 3. Installing MQSeries Integrator” on page 13.

Installing the configuration components

The Configuration Manager and the Control Center must be installed and operated on Windows NT. These components are supplied on the MQSeries Integrator for Windows NT Version 2.0.1 product CD, and you must refer to that CD and to the *MQSeries Integrator for Windows NT Version 2.0.1 Installation Guide* for full instructions on hardware and software prerequisites and installation procedures for these components.

You must install a single Configuration Manager in your domain. You can install one or more Control Centers on one or more systems running Windows NT.

Post-installation configuration and verification

When you have installed a broker on AIX, and have installed the Configuration Manager and Control Center on Windows NT, you can verify your installation.

“Chapter 4. Configuring a broker domain” on page 21 gives detailed step-by-step instructions for configuring your broker domain to verify your installation. “Chapter 5. Verifying your installation” on page 41 gives information on using the supplied verification programs to introduce you to some of the basic concepts and facilities of MQSeries Integrator.

Chapter 2. Planning for installation

Careful planning of your installation will help you clarify your requirements and the actions needed to achieve the environment you want. This chapter includes:

1. “System setup”.
2. “Product components” on page 6.
3. “Security considerations” on page 10.
4. “Migration considerations” on page 11.

System setup

This section provides details of the prerequisite products for installation, and related planning and setup information.

Hardware requirements

The general hardware requirements for MQSeries Integrator for AIX are:

- IBM RS/6000®; POWERserver®;
- IBM RS/6000 POWERstation
- IBM Scalable POWERparallel®; systems

Any communications hardware supporting NetBIOS, SNA LU 6.2, SPX, and TCP/IP

A minimum of 512 MB of RAM to support run-time operation of components

Diskspace requirements depend on the components you install and how much working space you need. This in turn depends primarily on your use of MQSeries resources such as queues and persistent messages.

You require 420 MB of storage space on your machine for the single installable image. You then use the System Management Interface Tool (SMIT) to select the components that you want to install.

Software requirements

Prerequisite software

The following products are prerequisites:

- **AIX Version 4.3.3.**
- **CSD 1 for MQSeries Integrator Version 2**.
- **IBM MQSeries for AIX Version 5.1 server**.

You **must** consult the `readme.txt` supplied on the media for MQSeries Integrator for AIX, for any additional information concerning product requisites, Program Temporary Fixes (PTFs), or patches. If you want to run the Scribble or Postcard samples on AIX, you need to have installed the MQSeries Client for Java™ on that machine.

MQSeries as well as MQSeries Client for Java must be at service level Corrective Service Diskette (CSD) 4. The product and CSD are supplied in the MQSeries Integrator package.

The server installation program checks that you have MQSeries for AIX Version 5.1 installed, and that it is at the correct service level, but the program does not check for the MQSeries Client for Java.

If you do not have the MQSeries components you need, you are recommended to install these before you continue with MQSeries Integrator installation.

Note: Version 5.0 of MQSeries for AIX is not supported at any service level.

- **Java Runtime Environment Version 1.1.8 PTF 8**

  This must be at least at the specified level, and is supplied in the MQSeries Integrator package.

- **A database product to support your broker or brokers.** This can be:
  - IBM DB2 Universal Database. Version 6.1 is provided on the product CD and can be installed by running the `db2setup` script, found in the `db2_v6.1` directory.
  - Oracle8
  - Sybase 11.5 or 12

Optional products

The following products are options, not prerequisites.

- **Connectivity**

  The network protocols supported are TCP/IP and UDP.

  TCP/IP is supplied as part of the base operating system

  UDP is supplied as part of the operating system or as part of the TCP/IP suite you are using.

- **Databases**

  A database product to support NEONRules and NEONFormatter. The following databases are supported:

  - **DB2 Version 6.1**

    This is the only DBMS supported by MQSeries Integrator that permits a database to participate as a Resource Manager in a distributed XA transaction, and coordinated by MQSeries as the XA Transaction Manager. In MQSeries Integrator, this is referred to as supporting a globally coordinated message flow.

Check the readme.txt file for your product to check if a Fixpack is required.

- **Oracle Versions 7.3.4, 8.1.5 and 8.1.6**

  Oracle 7.3.4 is **not** supported for use as a broker internal database but is for user data.
- **Sybase Versions 11.5 and 12**

  Sybase 12 is not supported by NEONRules and NEONFormatter nodes.

MQSeries Integrator broker requires access to a database for internal caching and for storing internal control information. The remaining components do not need access to a database.

DB2 has no additional prerequisite products, but it does require an additional 250 MB of disk storage. DB2 is a prerequisite of the Windows NT components only.

Note: The supplied DB2 product has **restricted license terms and agreements**. You must use this DB2 installation in association only with your licensed use of MQSeries Integrator for message management, and only the MQSeries Integrator components can make calls to the DB2 database.

The use of a database by the MQSeries Integrator components is independent of the use of databases by your applications and message flows. You are not restricted to the databases listed here for application and data storage and retrieval.

Note: These databases are for message definitions created through the NEONFormatter only. The databases required for internal product use are under “Prerequisite software” on page 3.

- Application programming support

  The following software compilers are supported:
  - IBM C compiler for AIX Version 4.4
  - IBM COBOL Set for AIX V1.1
  - Micro Focus COBOL Compiler Version 4.0 for UNIX®
  - IBM PL/I Set for AIX Version 1.1
  - IBM C and C++ compiler Version 5.0
  - IBM VisualAge®; C++ 5.0
  - IBM VisualAge Java™ Enterprise Edition for AIX V2.0

License information

Under the terms of the MQSeries Integrator Version 2.0.1 license agreement, you can install one instance of each component at any one time on any one system, with the exception of the Control Center. You can install the Control

Center on multiple systems providing that each Control Center is interacting with the same single Configuration Manager. You can create multiple brokers on a single system.

National language support

MQSeries Integrator Version 2.0.1 is enabled for national language support, but the user interface and message catalogs are currently available in US English only.

MQSeries Integrator Version 2.0.1 can process and construct messages in any code page supported by MQSeries for AIX Version 5.1.

Note: The NEONRules and NEONFormatter nodes support only the Latin1 code page in ASCII and EBCDIC. If you include these nodes within a message flow, this might restrict the messages that can be processed.

MQSeries Integrator interacts with MQSeries installed in any supported language. All languages for the MQSeries messaging products are included on the single MQSeries for AIX Version 5.1 CD.

All messages generated for internal intercomponent message exchange are generated in code page 1208.

DB2 Version 6.1 is NLS-enabled.

Product components

MQSeries Integrator for AIX Version 2.0.1 has one primary and four secondary components. For a description of the primary component, see “Primary components of MQSeries Integrator for AIX” and for a description of the secondary components, see “Secondary components” on page 7.

Primary components of MQSeries Integrator for AIX

MQSeries Integrator for AIX has a fileset that you can select using SMIT. A set of common files is also installed with the fileset of the primary component.

The primary fileset is `mqsi.base.runtime`

For a functional description of the sub-components, see “Chapter 1. Installation overview” on page 1.

Runtime

The Runtime component must be installed before creating brokers or a User Name Sever. The Runtime fileset is made up of:

- **Broker** After installing the Runtime fileset, you can create the broker sub-component on one or more systems subject to your license agreement (see “License information” on page 5 for details). You can configure and activate any number of brokers on each of the systems on which you install the sub-component, subject to system resource constraints.

You must create and configure each broker individually. Each broker requires its own queue manager. However, a single queue manager can host a single broker, and the User Name Server, but they must have been created on the same system as the broker.

Each broker requires access to a database to create and maintain internal data in tables. The tables hold information about the broker’s current configuration (for example, the message flows that are assigned to it). You are advised to use a local database server for performance reasons, although client connection to a remote DB2 server is supported. If you use a DB2 client connection, you must consider network loading and reliability because delays will significantly impact the performance in the broker domain. Supported databases are detailed in 4 and “Appendix B. Setting up an Oracle8 broker database on MQSeries Integrator” on page 61 describes how to set up a broker database on Oracle8.

- **User Name Server**. You are recommended to install and configure one User Name Server within your broker domain.

  The User Name Server requires an MQSeries queue manager to be assigned to it, but does not require access to a database.

Secondary components

There are four optional components that you can install in your broker domain if you choose and all of them can be installed on both AIX and Windows NT. These are:

Table 1. Optional components

Component	Fileset
Samples and Software Developers’ Kit (SDK)	`mqsi.base.sdk`
NEON	`mqsi.compat.mqi110` You must have MQSeries Integrator Version 1.1 on your machine to install this.
Online documentation.	`mqsi.doc.en_US.pdf`
Tivoli® management support	`mqsi.compat.tivoli`

The components have no MQSeries connectivity requirements. You can install these components without a previous installation of MQSeries or a database.

Samples and Software Developers’ Kit (SDK)

This component comprises a set of application samples, and samples that illustrate how to use the plug-in extensions.

- **Application samples**

  These applications illustrate the basic techniques of application programming to take full advantage of MQSeries Integrator.

  - Sample programs

    Sample programs are supplied in C and Java. These programs are fully operational, and both source and executables are supplied.

    If you choose to use these samples and run them in your broker domain, you must ensure they are running in an environment in which MQSeries connectivity is available. Check the details of the operating systems and application programming languages supported by MQSeries clients, and by applications local to queue managers.

    You can also copy and modify these examples to create your own applications, or you can add sections of their code to existing applications to exploit MQSeries Integrator function.

    - The verification programs **Scribble**, **Postcard**, and **Soccer** are provided to help you test out your initial installation. These are described in “Running the predefined verification applications” on page 44.

      The set of programs that make up the **Soccer** application are used in the *MQSeries Integrator Programming Guide* to illustrate the various publish/subscribe programming techniques available to your application programmers.

  - Libraries and header files

    Library files required for building applications are included in this component.

    Headers required by applications written to the Message Queue Interface (MQI) or Application Messaging Interface (AMI) are included. Their use is illustrated in the application samples.

- **Software Developers’ Kit**

  This kit contains working examples of the plug-in extensions that you can create to enhance MQSeries Integrator. Source code is provided to illustrate the programming to use the system interfaces introduced by MQSeries Integrator, for both message parser and message processing node. Executable code is also provided. The headers and library files required by parsers and message processing nodes are also supplied.

  Parsers and processing nodes execute only on a system on which an MQSeries Integrator broker is installed.

Online documentation

Information for MQSeries Integrator is provided for online viewing using the Acrobat Reader application from Adobe. Every information unit is supplied in Portable Document Format (PDF). A searchable library in PDF, with a cross book index and search facility, is also provided. You can access the documentation without installing the product, the publications are in the mqsi_aix_documentation subdirectory in the root directory of the supplementary CD.

To read the documentation on AIX, ensure you have Acrobat Reader installed. Iinstall the Online documentation and from a command line type:

```
mqsidocs
```

You can install a copy of Acrobat Reader from the MQSeries Integrator Version 2.0.1 for AIX supplementary CD, or download a free copy (which must be at Version 4) from the Adobe Web site at:

```
http://www.adobe.com
```

This component can be installed on any system, including one that has no other MQSeries Integrator component installed. For example, you can choose to install one copy of the documentation on a central LAN server for all users to share.

For details of all publications supplied, see “MQSeries Integrator publications” on page xi.

NEON Interface

This option enables you to install NEON runtime support on its own to allow you to run the NEON rules and formats for migration to installed brokers.

To use the NEON rules and formats you must install the NEON runtime support component. This can be found on the MQSeries Integrator Version 2.0.1 for AIX CD in the NEON/<database type>/mqi110 directory. This component is equivalent to MQSeries Integrator Version 1.1.0 and includes CSD01. You can install this NEON component over an existing installation of MQSeries Integrator Version 1.1.0 using smit.

Tivoli management support

You can install the filesets required by the Tivoli product when you install MQSeries Integrator for AIX from the product CD. The fileset is:

```
mqsi.compat.tivoli
```

This option installs Tivoli configuration files and an Adobe PDF file that enable you to run Tivoli applications, providing that you have installed the Tivoli product.

For details on how to use the package, read the supplied PDF document.

Security considerations

Security control of MQSeries Integrator components, resources, and tasks depends on the definition of users and groups of users (*principals*) to the security subsystem of the operating system. MQSeries Integrator always creates a set of groups on the system on which it is installed.

The group is:
- **mqbrkrs**

You must assign users (or other groups) to this local group to allow them to perform specific tasks.

Users must also have the appropriate authority to MQSeries resources (queues and queue managers) and to the databases being used by the broker.

Table 2 provides a summary of authorizations in the UNIX environment.

Table 2. Summary of authorization in the UNIX environments

User is...	UNIX domain
Creating broker, User Name Server	• Member of **mqbrkrs** • The broker or User Name Server will run under the service user ID specified on the create command in most situations; however, ‘root’ can nominate any user to run the broker
Installing	User must be a superuser
Uninstalling	User must be a superuser
Changing broker, User Name Server	User that the broker or User Name Server runs as, or ‘root’
Deleting broker, User Name Server	User that the broker or User Name Server runs as, or ‘root’
Starting and stopping broker, User Name Server	Member of **mqbrkrs**. The broker or User Name Server will run under the service user ID specified in the create command
Listing broker, User Name Server	Member of **mqbrkrs**
Changing, displaying, retrieving trace information	Member of **mqbrkrs**
Running User Name Server (service user ID)	Member of **mqbrkrs**
Running broker (service user ID)	Member of **mqbrkrs**
Clearing, joining, listing MQSeries publish/subscribe brokers	Member of **mqbrkrs**

Table 2. Summary of authorization in the UNIX environments (continued)

User is...	**UNIX domain**
Running publish/subscribe applications	Any user, subject to MQSeries Integrator topic and MQSeries queue access control

Migration considerations

For guidance on planning migration or integration of brokers, see the *MQSeries Integrator Introduction and Planning* book. For details of actions to take for integration and migration, see the *MQSeries Integrator Administration Guide*.

Chapter 3. Installing MQSeries Integrator

This chapter tells you how to install MQSeries Integrator for AIX Version 2.0.1 under these headings:

- "Delivery media".
- "Preparing for installation" on page 14.
- "Installation procedure" on page 16.
- "Setting up for remote installation" on page 18.
- "What to do if something goes wrong during installation" on page 19.

Delivery media

The MQSeries Integrator for AIX Version 2.0.1 package includes the following:

- MQSeries Integrator for AIX Version 2.0.1 CD

  This CD contains:

  – MQSeries Integrator for AIX Version 2.0.1

  – MQSeries Integrator for AIX Version 2.0.1 documentation

    You can install the product documentation separately from `mqsi.doc.en_US.pdf` in the `mqsi` directory on this CD.

  – DB2 for AIX Version 6.1 (English only version)

  – NEON Interface

  – Tivoli Interface

  Up-to-date details of the service levels required are included in the MQSeries Integrator Version 2.0.1 `readme.txt` file this CD.

- MQSeries Integrator for AIX Version 2.0.1 supplemental CD:

  This CD contains:

  – MQSeries for AIX Version 5.1 CSD4.

  – MQSeries for Windows NT Version 5.1 CSD4.

    The CSDs are provided to enable you to upgrade existing installations of MQSeries for AIX or Windows NT Version 5.1.

  – IBM DB2 Universal Database fixpack 4 for AIX

  – Adobe Acrobat reader Version 4.05

  – Product documentation

    This can be viewed from the `mqsi_aix_documentation` subdirectory.

  – Additional product service, if required.

- MQSeries for AIX Version 5.1 CD

If you are installing MQSeries for AIX Version 5.1, you must also install MQSeries for AIX Version 5.1 CSD4. This CSD can be found in the /mqm-csd4 subdirectory on the MQSeries Integrator for AIX supplemental CD. This product is provided in all available national languages.

- MQSeries Clients Version 5.1 CD

  The followings CDs are also included in the package. Refer to the *MQSeries Integrator for Windows NT Version 2.0.1 Installation Guide* for information about the Windows NT components.

- MQSeries Integrator for Windows NT Version 2.0.1

  This CD includes the following:

  - MQSeries Integrator for Windows NT Version 2.0.1
  - The IBM DB2 Universal Database Client for Windows NT.

    The Administration client and the Run-time client are included in all available national languages.
  - NEON Interface

- MQSeries for Windows NT Version 5.1

  If you are installing MQSeries for Windows NT Version 5.1, this version already has CSD4 applied. If you want to update an existing installation of MQSeries, the CSD can be found in the \Mqcsd4nt subdirectory on the MQSeries Integrator for AIX supplemental CD. This product is provided in all available national languages.

For details about these products, and their use with MQSeries Integrator, see “Software requirements” on page 3.

Preparing for installation

This section informs you of the steps you must take before you install and use MQSeries Integrator for AIX.

Before starting to install MQSeries Integrator for AIX you must review readme.txt, which you can find in the root directory of the CD.

If the machine is in a network information service (NIS) domain, you:

- Must create a user group with the name mqbrkrs

  To do this, either type smitty and select:

  ```
  Security & Users
    Groups
      Add a Group
  ```

  or, use the fastpath command smitty mkgroup.

In the ‘Group NAME’ field enter mqbrkrs. In the ‘USER’ list enter root and any other users who are going to use the MQSeries Integrator. Ensure that these users also have the appropriate authorizations to MQSeries resources (for example queue manager and queues) and to the databases that will be used by the broker or as part of the message flow.

Note: You can use an existing ID but you must add it to mqbrkrs and mqm

For stand-alone machines, you can create the new user and group IDs locally. For machines administered in a network information services (NIS) domain, you can create the user and group IDs on the NIS master server machine.

- Are recommended to create and mount a /var/mqsi journaled file system. This is required for the local-specific data (variables). For example, /var/mqsi will hold the file that defines the ODBC connections for that installation.

After installation, the user bin, and group bin, own the directories and files that contain the resources associated with this product. This group and user must be defined for any machine on which the MQSeries Integrator software is to be installed, whether the machine is a client or a server machine.

You can also create another file system for product code (a minimum of 60MB is required). If, for example, you do not want to have the product code installed in the default of /usr/opt/mqsi file system because it is too small , you can do one of two things:

1. Create a new file system and mount it as /usr/opt/mqsi.
2. Create a new directory anywhere on your machine that is large enough to contain the product, and create a symbolic link from /usr/opt/mqsi to this new directory. For example:

   ```
   mkdir /bigdisk/mqsi
   ln -s /bigdisk/mqsi /usr/opt/mqsi
   ```

Notes:

1. Whichever of these options you pick, you **must** do it before installing the product code.
2. The file system into which the code is installed can be a remote network device, for example NFS, provided that the mount options are defined on that device to allow **setuid** programs - including root access - to be run.

The most usual way to install this product is direct from the CD. You can also set up for installation from a remote drive. This is described in “Setting up for remote installation” on page 18.

Preparing for installation

You are guided through the installation process, and are prompted for any information required for completion.

Prerequisites

MQSeries Integrator for AIX installation checks for the presence of all the prerequisite software required by your installation choices. If any prerequisite is not found, you are presented with a message detailing which prerequisite is missing. You can terminate installation at this point and install the prerequisite from the relevant CD.

You are advised to check the full details of prerequisites for each component given in “Software requirements” on page 3.

Installation procedure

Notes:

1. If you have previously installed MQSeries Integrator for AIX on your system, you must remove the product using the System Management Interface Tool before installing again. Refer to “Appendix C. Uninstalling MQSeries Integrator” on page 65 for details.
2. If the product is present, but not installed correctly, you must manually delete the files and directories contained in:
   - `/var/mqsi`
   - `/usr/opt/mqsi`

Carry out the following procedure:

1. Become root
2. Use the System Management Interface Tool to install MQSeries Integrator for AIX
   1. Type `smitty`
   2. Select:

      ```
      Software Installation and Maintenance
        Install and Update Software
          Install and Update from Latest Available Software
      ```

      Ensure that you have set the `AUTOMATICALLY Install requisite software` flag to `yes`.

   or, use the fastpath command `smitty install_latest` and continue to step 5 on page 17.
3. Press **F4**
4. From `Input device/directory for software`, press F4 to list all the available options. Select your CD-ROM drive from the list

5. Select SOFTWARE to iInstall. Press F4 to list the options.
6. Follow the on-screen instructions to select the components you want to install

   Note: In particular, you can ignore the fileset mqsi.compat.mqi110 if you do not have MQSeries Integrator Version 1.1 nstalled.
7. Press the Enter key to install the product

During installation MQSeries Integrator for AIX checks that the following prerequisite products have been installed:

- MQSeries for AIX Version 5.1 with CSD4

It also checks that the following group has been created:

- mqbrkrs

If any of the above are missing or not created, the installation process will terminate with an error. You must then satisfy the prerequisite before attempting to install MQSeries Integrator for AIX again.

Setting specific environment variables

View the readme.txt to ensure you have the latest sample profile shipped with this product /opt/mqsi/sample/profiles

Setting up a database

After installation you must set up your required database for use with MQSeries Integrator. There is a script provided called mqsi_setupdatabase in the /usr/opt/mqsi/bin directory, this will accept a database type and home directory of a database as arguments. For example, to set up DB2 for use with MQSeries Integrator the command would be:

```
mqsi_setupdatabase db2 /usr/lpp/db2_06_01
```

where /usr/lpp/db2_06_01 is the directory in which DB2 is installed.

Other supported database types are **oracle** (for Oracle Version 7 or 8) and **sybase11** or **sybase12**.

- For further information about setting up a DB2 database, see “Chapter 4. Configuring a broker domain” on page 21.
- For further information about setting up an Oracle database, see “Appendix B. Setting up an Oracle8 broker database on MQSeries Integrator” on page 61.
- For further information about setting up a Sybase database, see the *MQSeries Integrator Administration Guide* and Sybase documentation.

If an invalid database type is entered an database <database type> error message and usage information will be displayed, this will list all available database types. If the required libraries cannot be found at the specified location a the following error message will be displayed:

```
<libraries> not found in lib directory of <database home directory>
but links created anyway
```

This might happen if the product is not accessible because it is installed on a mounted directory. The links created will **NEVER** be added to the package database and you will have to manually remove them from `/var/mqsi/lib`

Setting up for remote installation

On AIX you can choose one of the following two methods to make the MQSeries Integrator installation files accessible on a remote server:

- You can make the MQSeries Integrator CD-ROM drive shareable.
- You can copy the product files from the CD-ROM to the server by following these steps:
  1. Create a directory on the server to store the installation files. For example:

     ```
     mkdir /instmqsi
     ```
  2. Mount the MQSeries Integrator CD.
  3. Copy the entire CD to the new directory. For example:

     ```
     cp -rf /cdrom. /instmqsi
     ```

     This copies the complete contents of the CD to the specified location on the server.
  4. Give all licensed users access to the directory that now contains the CD-ROM image.
  5. Export the directory:
     a. Type `smitty`

     b. Select:

     ```
     Communications Applications and Services
       NFS
          Network File System (NFS)
            Add a Directory to Exports List
     ```

     or, use the fastpath command `smitty mknfsexp`

     c. Complete the fields as appropriate

     d. Press the Enter key
  6. On the target machine, create a directory that will mount the exported directory. For example:

     ```
     mkdir /remotemqsiimage
     ```
  7. From a command prompt on the target machine:

- – Mount the remote directory using the **mount** command. For example:

  ```
  mount <machine name>:/instmqsi /remotemqsiimage
  ```

  Where <machine name> is the name of the target machine.

8. Change to the installation directory:
   a. Become root
   b. Use the System Management Interface Tool to install MQSeries Integrator for AIX
      1) Type:

         ```
         smitty
         ```

      2) Select:

         ```
         Software Installation and Maintenance
           Install and Update Software
             Install and Update from Latest Available Software
         ```

      or, use the fastpath command `smitty install_latest`
   c. Enter the path to the remote install images in the `INPUT device / directory for software` field
   d. Select from the list of `SOFTWARE to Install`.
   e. Press the PF4 key to select the components that you want to install

      Note: In particular, you can ignore the `mqsi.compat.mqi110` selection if you do not have MQSeries Integrator installed.
   f. Press the Enter key to install the product
9. Follow the installation prompts.

What to do if something goes wrong during installation

If you encounter any problems during installation, you are advised to check the following:

- Review the `readme.txt` file supplied on the CD. This has the most up-to-date information available for product installation and operation. There might be last minute changes to the installation process that you must follow. You might also find additional information on the MQSeries Web site. (The address is given in “MQSeries information available on the Internet” on page xiii).
- Check the MQSeries logs logs and syslog for errors.
- If a message is displayed with the MQSeries Integrator prefix of BIP, check the *MQSeries Integrator Messages* book to determine the cause of the error and the action you need to take to correct it.
- Additionally, you should:
  1. Type `smitty`

2. Select

   ```
   Software Installation and Maintenance
     Software Maintenance and Utilities
       Verify Software Installation and Requisites
   ```

   or, use the fastpath command `smitty maintain_software`, together with `Verify Software Installation and Requisites`. Select the fileset to check the installation of the software.
3. Use the `Check Software File Sizes After Installation` option and select the fileset to check, to verify that the files were copied correctly to the system.

Note: If the installation was interrupted before installation completed, use the `Clean Up After Failed or Interrupted Installation` option to restore the system to its previous state. You can then try installing the product again.

When you have identified and corrected the error, or errors, you can run the installation program again. If this does not work, you are advised to follow the steps for manual uninstallation to ensure that your system is in a consistent state before you retry.

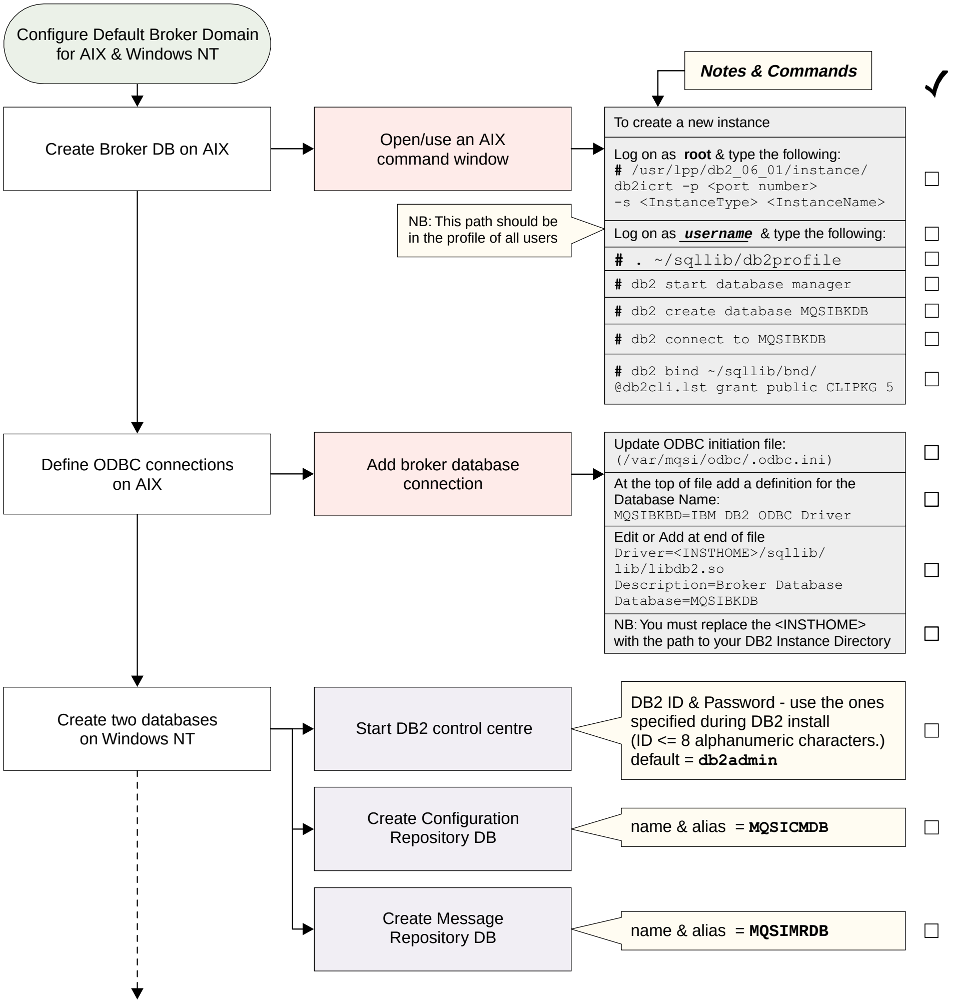
If you are unable to resolve the problems, after checking the possible sources of error listed above, contact your IBM Support Center. See “Contacting your IBM Support Center” on page 65 for further information.

Chapter 4. Configuring a broker domain

This chapter takes you through the tasks you must complete, on AIX and Windows NT, to set up the minimum resources required in the broker domain, after you have installed MQSeries Integrator for AIX.:

There are configuration assumptions and nine tasks:
- “Reviewing the assumptions about this configuration” on page 25
- “Task 1: Creating and connecting to the databases” on page 27
- “Task 2: Setting up database authorizations” on page 30
- “Task 3: Creating the Configuration Manager on Windows NT” on page 30
- “Task 4: Configuring the syslog on AIX” on page 34
- “Task 5: Creating a broker on AIX” on page 34.
- “Task 6: Connect the Windows NT and AIX queue managers” on page 36
- “Task 7: Start the Configuration Manager and the Broker” on page 37
- “Task 8: Starting the Control Center” on page 38
- “Task 9: Connect the Broker to the Domain” on page 39

These tasks are shown diagrammatically, starting with Figure 1 on page 22. You might find it helpful to review these figures before you start the tasks detailed in this section, and use it to check off the tasks as you complete them. You must complete all the steps in each task. The figures and the text that follows describe the tasks to be carried out on each platform in a logical sequence. You might prefer to carry out all the tasks on AIX followed by all the tasks on Windows NT: the outcome is the same.


Configure Default Broker Domain for AIX & Windows NT
Notes & Commands
✓
Create Broker DB on AIX
Open/use an AIX command window
NB: This path should be in the profile of all users
Define ODBC connections on AIX
Add broker database connection
Create two databases on Windows NT
Start DB2 control centre
Create Configuration Repository DB
Create Message Repository DB

To create a new instance

Log on as **root** & type the following: □

```
# /usr/lpp/db2_06_01/instance/
db2icrt -p <port number>
-s <InstanceType> <InstanceName>
```

Log on as ***username*** & type the following: □

```
# . ~/sqllib/db2profile
```
□

```
# db2 start database manager
```
□

```
# db2 create database MQSIBKDB
```
□

```
# db2 connect to MQSIBKDB
```
□

```
# db2 bind ~/sqllib/bnd/
@db2cli.lst grant public CLIPKG 5
```
□

Update ODBC initiation file: (/var/mqsi/odbc/.odbc.ini) □

At the top of file add a definition for the Database Name: □

```
MQSIBKBD=IBM DB2 ODBC Driver
```

Edit or Add at end of file □

```
Driver=<INSTHOME>/sqllib/
lib/libdb2.so
Description=Broker Database
Database=MQSIBKDB
```

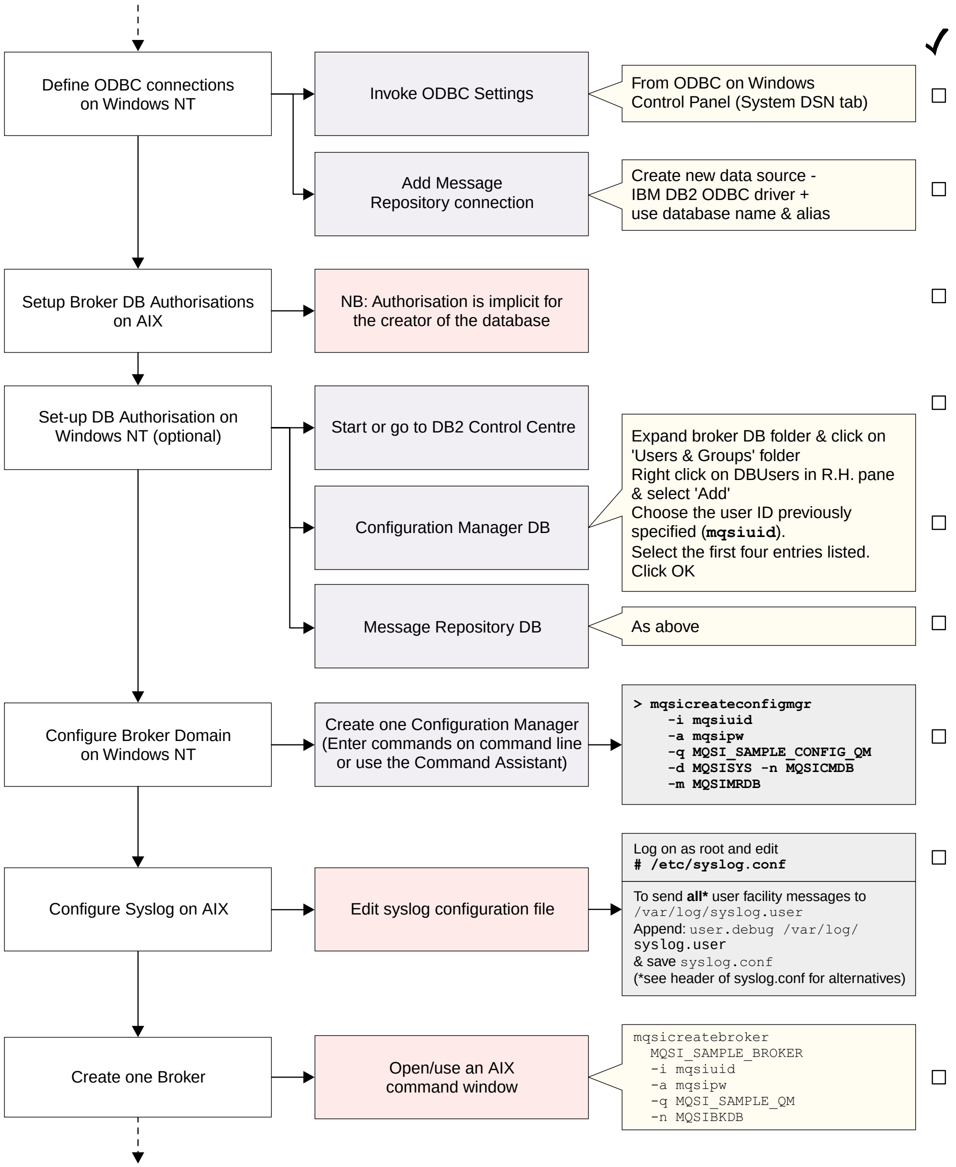
NB: You must replace the <INSTHOME> with the path to your DB2 Instance Directory □

DB2 ID & Password - use the ones specified during DB2 install (ID <= 8 alphanumeric characters.) default = **db2admin** □

name & alias = **MQSICMDB** □

name & alias = **MQSIMRDB** □

Figure 1. Configuring a simple broker domain



✓

Define ODBC connections on Windows NT → Invoke ODBC Settings — From ODBC on Windows Control Panel (System DSN tab) □

→ Add Message Repository connection — Create new data source - IBM DB2 ODBC driver + use database name & alias □

Setup Broker DB Authorisations on AIX → NB: Authorisation is implicit for the creator of the database □

Set-up DB Authorisation on Windows NT (optional) → Start or go to DB2 Control Centre □

→ Configuration Manager DB — Expand broker DB folder & click on 'Users & Groups' folder Right click on DBUsers in R.H. pane & select 'Add' Choose the user ID previously specified (**`mqsiuid`**). Select the first four entries listed. Click OK □

→ Message Repository DB — As above □

Configure Broker Domain on Windows NT → Create one Configuration Manager (Enter commands on command line or use the Command Assistant) → □

```
> mqsicreateconfigmgr
    -i mqsiuid
    -a mqsipw
    -q MQSI_SAMPLE_CONFIG_QM
    -d MQSISYS -n MQSICMDB
    -m MQSIMRDB
```

Configure Syslog on AIX → Edit syslog configuration file →

Log on as root and edit

```
# /etc/syslog.conf
```

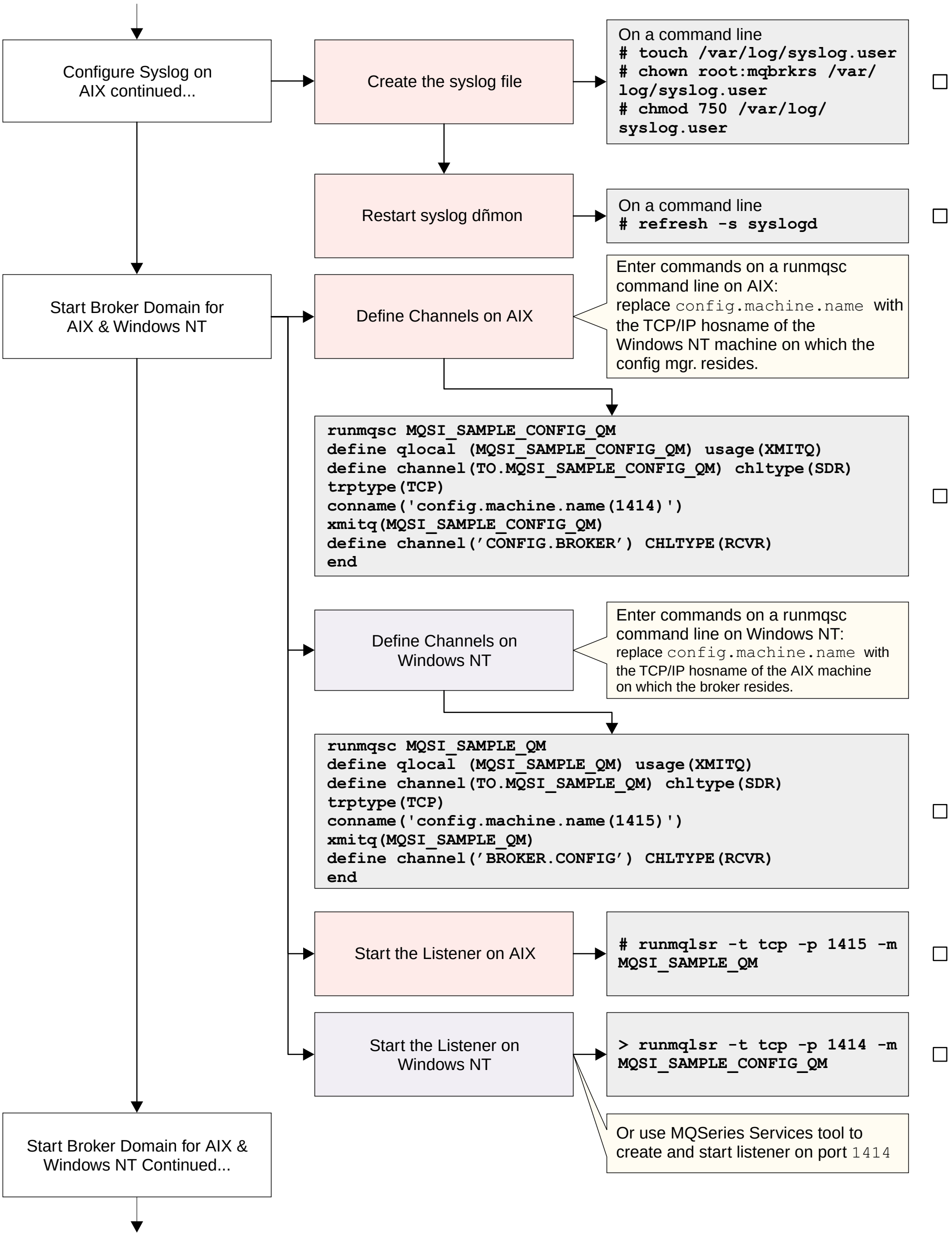
□

To send **all*** user facility messages to `/var/log/syslog.user`
Append: `user.debug /var/log/syslog.user`
& save `syslog.conf`
(*see header of syslog.conf for alternatives)

Create one Broker → Open/use an AIX command window — □

```
mqsicreatebroker
  MQSI_SAMPLE_BROKER
  -i mqsiuid
  -a mqsipw
  -q MQSI_SAMPLE_QM
  -n MQSIBKDB
```

Figure 2. Configuring a simple broker domain continued



Configure Syslog on AIX continued...

Create the syslog file

On a command line

```
# touch /var/log/syslog.user
# chown root:mqbrkrs /var/
log/syslog.user
# chmod 750 /var/log/
syslog.user
```

□

Restart syslog dñmon

On a command line

```
# refresh -s syslogd
```

□

Start Broker Domain for AIX & Windows NT

Define Channels on AIX

Enter commands on a runmqsc command line on AIX: replace `config.machine.name` with the TCP/IP hosname of the Windows NT machine on which the config mgr. resides.

```
runmqsc MQSI_SAMPLE_CONFIG_QM
define qlocal (MQSI_SAMPLE_CONFIG_QM) usage(XMITQ)
define channel(TO.MQSI_SAMPLE_CONFIG_QM) chltype(SDR)
trptype(TCP)
conname('config.machine.name(1414)')
xmitq(MQSI_SAMPLE_CONFIG_QM)
define channel(’CONFIG.BROKER’) CHLTYPE(RCVR)
end
```

□

Define Channels on Windows NT

Enter commands on a runmqsc command line on Windows NT: replace `config.machine.name` with the TCP/IP hosname of the AIX machine on which the broker resides.

```
runmqsc MQSI_SAMPLE_QM
define qlocal (MQSI_SAMPLE_QM) usage(XMITQ)
define channel(TO.MQSI_SAMPLE_QM) chltype(SDR)
trptype(TCP)
conname('config.machine.name(1415)')
xmitq(MQSI_SAMPLE_QM)
define channel(’BROKER.CONFIG’) CHLTYPE(RCVR)
end
```

□

Start the Listener on AIX

```
# runmqlsr -t tcp -p 1415 -m
MQSI_SAMPLE_QM
```

□

Start the Listener on Windows NT

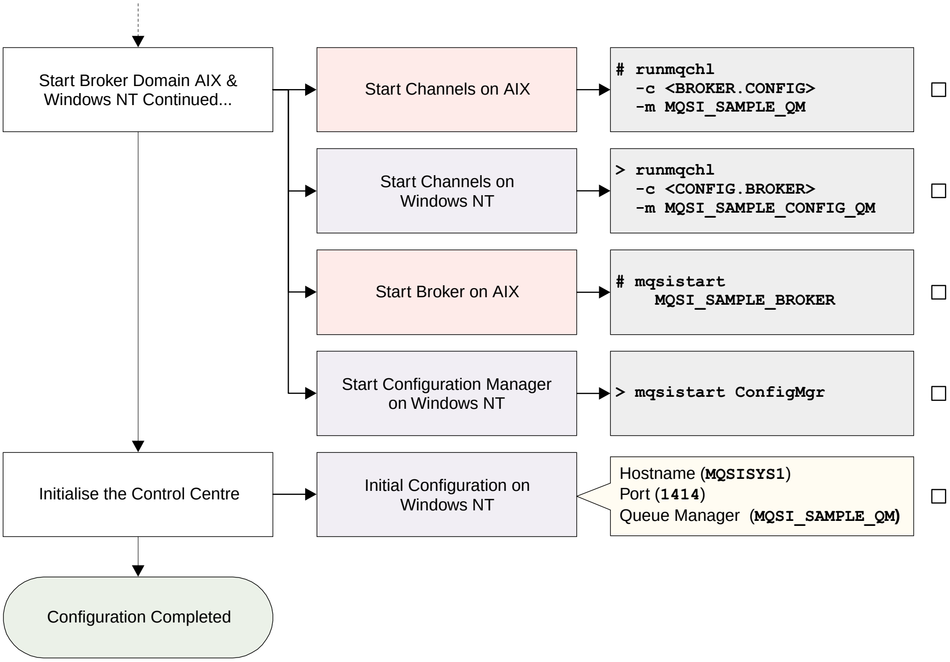
```
> runmqlsr -t tcp -p 1414 -m
MQSI_SAMPLE_CONFIG_QM
```

□

Or use MQSeries Services tool to create and start listener on port `1414`

Start Broker Domain for AIX & Windows NT Continued...

Figure 3. Configuring a simple broker domain continued


Start Broker Domain AIX & Windows NT Continued...
Start Channels on AIX
Start Channels on Windows NT
Start Broker on AIX
Start Configuration Manager on Windows NT
Initialise the Control Centre
Initial Configuration on Windows NT
Configuration Completed
□
□
□
□
□

```
# runmqchl
  -c <BROKER.CONFIG>
  -m MQSI_SAMPLE_QM
```

```
> runmqchl
  -c <CONFIG.BROKER>
  -m MQSI_SAMPLE_CONFIG_QM
```

```
# mqsistart
    MQSI_SAMPLE_BROKER
```

```
> mqsistart ConfigMgr
```

Hostname (**MQSISYS1**)
Port (**1414**)
Queue Manager (**MQSI_SAMPLE_QM)**

Figure 4. Configuring a simple broker domain continued

Most of these steps make use of commands supplied by MQSeries Integrator. On AIX and Windows NT, some of these commands can be invoked using the MQSeries Integrator Command Assistant; on both platforms, all of them can be invoked from the command line. The MQSeries Integrator Command Assistant screens are illustrated, and the commands are given in full. On AIX and Windows NT you can choose which method you want to use to issue for some of these commands.

The Command Assistant and the configuration commands are described in detail in the *MQSeries Integrator Administration Guide*, which provides further reference and guidance material, and describes the actions you must take if you experience any errors in completing the tasks illustrated here.

Reviewing the assumptions about this configuration

Before you start to define any resources, review the assumptions made about the simple configuration that is created. If you want to understand more about MQSeries Integrator configuration in general, refer to *MQSeries Integrator Introduction and Planning*.

The assumptions for this configuration cover resource names, user IDs, and so on. If you want to override any of the assumptions, make a note of changes you want to make and apply those changes as you complete the tasks illustrated. For example, the names used for the broker and its queue manager are for illustration only. You are recommended to follow any existing naming conventions you have for MQSeries (or any other) resources. See *MQSeries Integrator Introduction and Planning* for more information about defining a naming convention.

This chapter assumes:

- You have installed the Runtime component on AIX.
- You have installed the Configuration Manager and the Control Center components on Windows NT.
- You have installed the product on systems with TCP/IP hostnames of MQSISYS1 and MQSISYS2. You **must** replace these name wherever they are used with the hostnames of your systems, if they are different.
- The MQSeries ports 1414 and 1415 are available. You **must** replace these ports with different ports wherever they are used, if these default ports are not available.
- The local systems MQSISYS1 and MQSISYS2 define the security domain relevant to this configuration (that is, all users and groups are defined in the local account security domain).

  Note: This illustrates the simplest security scenario. You can find more comprehensive security information in *MQSeries Integrator Introduction and Planning*, and more complex scenarios illustrated in the *MQSeries Integrator Administration Guide*.
- You must ensure that your current logon ID is a member of:
  - The `mqbrkrs` group on AIX
  - The **Administrator** group on Windows NT

  or that you are logged on with the same user ID that you used to install MQSeries Integrator.

  You also need authorization to the MQSeries resources that link the Windows NT and AIX components (queues and queue mangers) and to the databases. You need the appropriate DB2 authorizations as well.If you want to run any administration commands, for example, **mqsicreatebroker** (create broker) or **mqsistart** (start MQSeries Integrator component), your user ID must be a member of group `mqbrkrs`.
- You have decided to create a new user ID, the ‘service userid’ (*mqsiuid*), to use as the user ID under which the MQSeries Integrator services (the Configuration Manager and the broker) will run, and as the user ID under which all MQSeries Integrator databases are accessed.

In most cases, you are unable to change these user IDs after your configuration has been set up, so you are advised to check where you use them very carefully.

- The configuration includes one broker, installed on AIX, and the Configuration Manager, installed on Windows NT, communicating through MQSeries.
- A set of sample names and other default values are used for MQSeries Integrator, MQSeries, and database resources. You can use the sample names and defaults exactly as they are shown, or you can decide to use your own names, to follow the naming conventions you have in place.

  If you choose to use your own names, you **must** change the names and default values to match your configuration, whenever they are used in the tasks illustrated.
- You are using DB2 for all database requirements. DB2 has been installed as part of your installation procedure, and you have since restarted your system and verified your DB2 installation.

After you have verified your installation, and understood and implemented the basic principles of operating your broker domain, you will probably need a more complex MQSeries network for your broker domain. Your brokers and the Configuration Manager are likely to be located on different physical machines, and you are likely to include a User Name Server in your broker domain. For more detailed guidance and instructions, refer to the *MQSeries Integrator Administration Guide*.

Task 1: Creating and connecting to the databases

Note

On multiway machines you must bind the db2cli package to the configuration-manager database, by opening a DB2 Command Line Processor window and carrying out the following procedure:

1. Connect to the database name.
2. Issue the command `Bind c:\sqllib\bnd\@db2cli.lst`, blocking all grant public.
3. Connect and reset.

where `c:\` is the drive on which you installed DB2.

You must take the following steps:

If you ran `db2setup`, a default instance has been created. You are recommended to use the default instance, but, if you wish to create a separate instance, use the command in step 1. Otherwise start at step 2.

Creating and connecting to the database on AIX

1. Log on as root, using the following commands:

   ```
   /usr/lpp/db2_06_01/instance/db2icrt -p <port number> -s <InstanceType> <InstanceName>
   ```

2. Log on as <username>, using the following command. Press enter at the end of each of these lines:

   ```
   . ~/sqllib/db2profile
   db2 start database manager
   db2 create database MQSIBKDB
   db2 connect to MQSIBKDB
   db2 bind ~/sqllib/bnd/@db2cli.lst grant public CLIPKG 5
   ```

   <username> is the instance created in step 1 and is used to create the broker.

   Note that . ~/sqllib/db2profile should be in the profile of all users.

3. To enable MQSeries Integrator brokers to use the database, you must update the ODBC configuration file (/var/mqsi/odbc/.odbc.ini) to contain definitions for the database.

   Edit the file and add the following line.

   - At the end of the file replace <INSTHOME> with the path to the home directory of the username for whom the DB2 instance was created.

     ```
     Driver=<INSTHOME>/sqllib/lib/db2.o
     Description=Broker Database
     Database=MQSIBKDB
     ```

     The path identified by the Driver definition is specific to your installation, so you must replace the <INSTHOME> with the path to your DB2 Instance directory.

See the *MQSeries Integrator Version 2.0.1 Administration Guide* for more information about ODBC connections.

Creating and connecting to the database on Windows NT

1. Start the DB2 Control Center from the Start menu (*Start->Programs->DB2 for Windows NT->Control Center*).

   You must enter a valid user ID and password on the Sign On dialog presented. Use the the DB2 Administrator ID you specified when you installed DB2. If you accepted the default, the user ID is db2admin. If you changed this to another user ID, enter that user ID. Enter the password for the user ID you are using.

2. Create the databases.

   Expand the Object tree in the DB2 Control Center until you find Databases. Right-click Databases and select *Create Database using Smartguide*.

Note: DB2 database names are limited to eight characters.

When you have created the database or databases, click **Done**. A confirmation message indicating successful completion of the create command appears at the bottom of the window.

You are recommended to create two databases, to hold two independent sets of tables. You can let all remaining options take the default values.

- The configuration repository

  Enter the name and alias of your database. This chapter assumes you specify MQSICMDB as both the name and the alias. If you decide to use another name, enter that name here and in all other steps in which this database is referenced.
- The message repository

  Enter the name and alias of your database. This chapter assumes you specify MQSIMRDB as both the name and the alias. If you decide to use another name, enter that name here and in all the other steps in which this database is referenced.

If you prefer, you can create a single database to hold all the tables required. Whatever scheme you choose, ensure you specify the correct name whenever you are asked to specify a database in subsequent commands.

3. Define the ODBC connections.
   a. From the Windows NT Start menu, select *Start->Settings->Control Panel*.
   b. Within the Control Panel, double-click the ODBC icon (labelled *ODBC* or *ODBC Datasources*).
   c. Click the *System DSN* tab.
   d. You must add an ODBC connection for the message repository. The configuration repository does not need an ODBC connection if it is created as a separate database. However, if you have created a database that will be used for the message repository as well as the configuration repository, you must create an ODBC connection for that database.
      1) Click the Add button. The *Create New Data Source* window appears.
      2) Double-click IBM DB2 ODBC DRIVER.
      3) Choose the data source (database) name from the drop-down list.
      4) Click **OK**.

      When you have completed these steps for the message repository database, click **OK**.

Task 2: Setting up database authorizations

The next task is to authorize selected user IDs to access the databases you have created, to allow the MQSeries Integrator resources to operate successfully. The steps you need to complete are shown below. If you need further guidance about any of these tasks, use the online help facility of the DB2 Control Center.

Note: You can omit this step if you choose to specify your DB2 administrator ID and password for the datasource and database IDs and passwords on the create broker and create Configuration Manager commands. This option is not illustrated in this book. See the *MQSeries Integrator Administration Guide* for further information.

1. Start the DB2 Control Center, if it is not already active. Log on with the DB2 administrator user ID you used
2. Complete the following tasks for each database you created
   a. Expand the object tree until you find the database.

   b. Expand the tree under this database and click the *User and Group Objects* folder. The *DB Users* and *DB Groups* folders are displayed in the right pane.

   c. Right-click the *DB Users* folder in the right pane and select *Add* from the pop-up menu. The Add User notebook opens.

   d. Select the user ID `mqsiuid` (or the ID you are using for MQSeries Integrator database access) from the drop-down list.

      Select the appropriate options from the choices in the box labelled *Choose the appropriate authorities to grant to the selected user* to all the databases you have created for MQSeries Integrator.

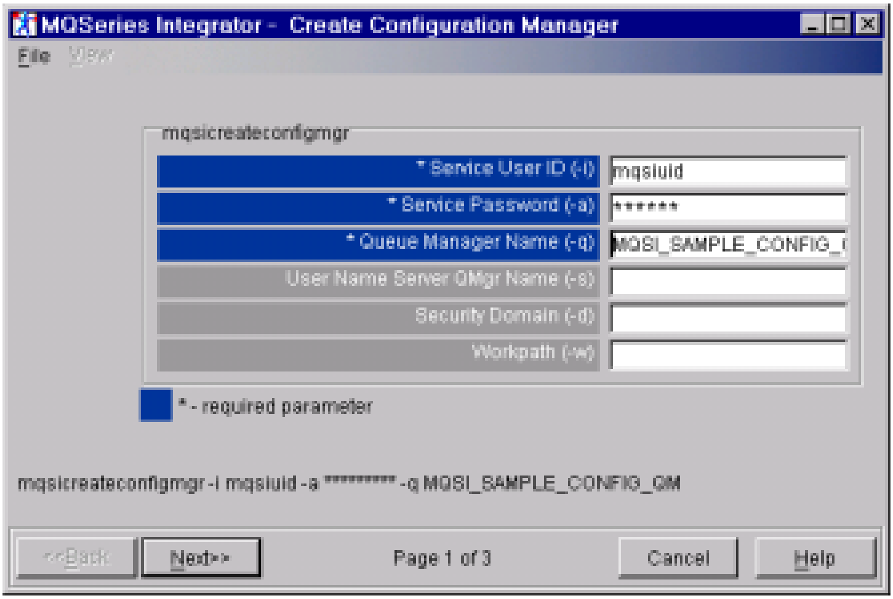
      The ID you specify as the `ServiceUserID` on the create commands, `mqsiuid` (or the user ID you are using in place of this sample ID), must have the following authority to all the databases you have created for MQSeries Integrator:

      - Connect database
      - Create tables
      - Create packages
      - Register functions to execute in database manager’s process

   e. Click **OK**. The authority or authorities are granted. The dialog is closed.
3. Close the DB2 Control Center.

Task 3: Creating the Configuration Manager on Windows NT

Start the Command Assistant to create the Configuration Manager (select *Start->Programs->IBM MQSeries Integrator 2.0.1->Command Assistant->Create Configuration Manager*). You must complete the fields on two screens, shown in Figure 5 on page 31 and Figure 6 on page 31. You can then review and check the full command (compare it to the command shown) and click **Finish** on the third summary screen.



mqsicreateconfigmgr

* Service User ID (-i) mqsiuid

* Service Password (-a) ******

* Queue Manager Name (-q) MQSI_SAMPLE_CONFIG_(

User Name Server QMgr Name (-s)

Security Domain (-d)

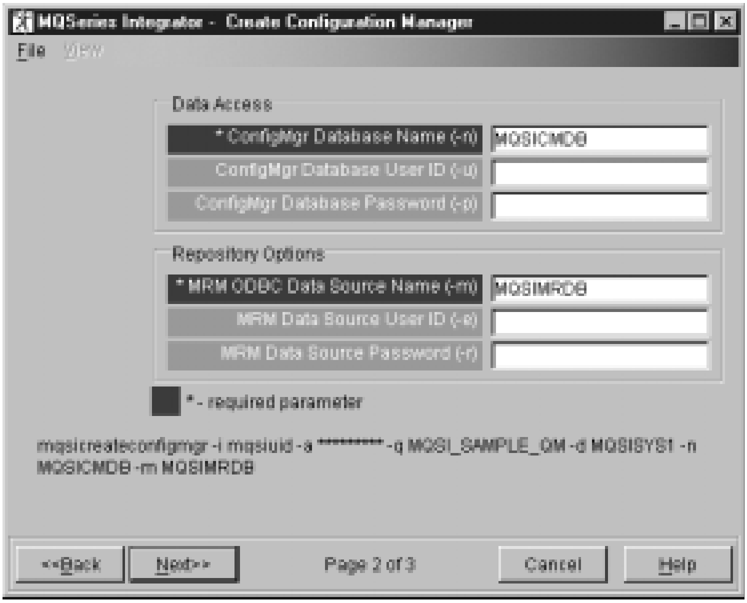
Workpath (-w)

▇ * - required parameter

```
mqsicreateconfigmgr -i mqsiuid -a ********* -q MQSI_SAMPLE_CONFIG_QM
```

Page 1 of 3

Figure 5. Creating the Configuration Manager: screen 1



Data Access

* ConfigMgr Database Name (-n) MQSICMDB

ConfigMgr Database User ID (-u)

ConfigMgr Database Password (-p)

Repository Options

* MRM ODBC Data Source Name (-m) MQSIMRDB

MRM Data Source User ID (-e)

MRM Data Source Password (-r)

▇ * - required parameter

```
mqsicreateconfigmgr -i mqsiuid -a ********* -q MQSI_SAMPLE_QM -d MQSISYS1 -n
MQSICMDB -m MQSIMRDB
```

Page 2 of 3

Figure 6. Creating the Configuration Manager: screen 2

Configuration manager creation

If you prefer, you can enter the following command:

```
mqsicreateconfigmgr -i mqsiuid -a mqsipw -q MQSI_CONFIG_QM
-d MQSISYS1 -n MQSICMDB -m MQSIMRDB
```

This identifies the queue manager (flag -q) that will host the Configuration Manager, the service user ID (flag -i) and password (flag -a) that the Configuration Manager will run under (as a Windows NT service), the security domain (flag -d) within which user authority is checked (in this case, the local account security domain defined by the hostname of this system), the database for the configuration repository (flag -n), and the datasource name for the message repository (flag -m).

Note: The queue manager will be created for you if it does not already exist.

The service user ID and password are also used as the user ID and password for both the configuration repository and the message repository. If you are using a different user ID and password for access to these repositories, you must specify these here (flags -u and -p for the configuration repository and flags -e and -r for the message repository).

When you type the password, it appears on the command line exactly as you type it. However, when you type it into the Command Assistant, and when it is stored in the Windows NT registry, it is displayed as asterisks for security reasons.

If you are using different names or values for any parameter on this command, you **must** replace the appropriate values with your own.

The command might take a short while to complete. The command generates the following expected responses, unless you are using the Command Assistant:

```
MQSeries queue manager created.
Creating or replacing default objects for MQSI_CONFIG_QM.
Default objects statistics : 29 created. 0 replaced. 0 failed.
Completing setup.
Setup completed.
MQSeries queue manager 'MQSI_SAMPLE_QM' started.
The setmqaut command completed successfully.
The setmqaut command completed successfully.
The setmqaut command completed successfully.
The setmqaut command completed successfully.
The setmqaut command completed successfully.
The setmqaut command completed successfully.
The setmqaut command completed successfully.
The setmqaut command completed successfully.
The setmqaut command completed successfully.
The setmqaut command completed successfully.
The setmqaut command completed successfully.
The setmqaut command completed successfully.
The setmqaut command completed successfully.
The setmqaut command completed successfully.
security properties not found. using defaults.
BIP8071I: Successful command completion.
```

Note: Only the message is shown in the log.

If the command detects any errors, or is unable to complete, it returns an error message on the command line, or in the Windows NT Event viewer (Application View) which includes the explanation and action to take in full. It is possible that the error has been caused by another component that MQSeries Integrator interacts with to complete this command (Windows NT, DB2, or MQSeries), so check for errors from these products, too. (The response `security properties not found. using defaults` shown above is not an error: you can ignore this informational message.)

Note: In some circumstances, you might see the following error message issued by the Java Runtime Environment (JRE).

```
address: [B@964f60
     security properties not found. using defaults.
     Can't get saved UUID state:java.io.FileNotFoundException:
     <mqsi_root>\bin\..\UUID
```

This error does not cause the **mqsicreateconfigmgr** command to fail, because the required file is created dynamically. You can therefore ignore this message.

On completion, the following have been done:

1. The Configuration Manager has been created, and the Windows NT service for it added to the Services (viewable from the Windows NT Control Panel). The service is called *IBM MQSeries Broker ConfigMgr*. It has a default start up status of manual: you can change this to automatic if you want to.
2. The queue manager MQSI_SAMPLE_QM has been created and started. You can check the existence and status of this queue manager using MQSeries Services from the Start menu (*Start->Programs->IBM MQSeries->MQSeries Services*).
3. The MQSeries resources required by the Configuration Manager have been defined on the queue manager. These resources are detailed in “Default MQSeries resources” on page 58.
4. The authorizations required by the Configuration Manager to access MQSeries resources have been set (the *setmqaut* messages seen in the responses to the command).
5. The database tables required by the configuration repository have been set up in the database MQSICMDB.

6. The database tables required by the message repository have been set up in the database MQSIMRDB.
7. The Windows NT registry has been updated to record the Configuration Manager creation.

Task 4: Configuring the syslog on AIX

Before creating a broker on AIX, you are recommended to configure syslog to write all ‘user’ messages to a file, because all MQSI V2 messages, not generated by the command line utilities, are written to the syslog. You can select any location for the file where you want to log the messages.

To configure the syslog you need to (as root) edit the file `/etc/syslog.conf` because this file contains definitions on where to write messages written to the syslog.

All MQSeries Integrator messages are written to the ‘user’ facility, so you need to add a line starting with the text “user” and then select what level of messages you want to see. For example you can choose information, debug, warning, or error. Debug is selected in the example below.

Initially, you are advised to send all levels of message to a file. However, as you become used to the system, you may want to direct high severity message to a different file or even have them mailed to an administrator.

To direct all user facility messages to the file `/var/log/syslog.user`, create the file `syslog.user` and add the following line to the end of the syslog.conf file: `user.debug /var/log/syslog.user`.

Before committing the changes by restarting syslogd you must log in as root and create the following file:

```
touch /var/log/syslog.user
chown  root:mqbrkrs /var/log/syslog.user
chmod 750 /var/log/syslog.user
```

You must restart the syslog daemon (as root) for your changes to take effect, using the command:

```
refresh -s syslogd
```

Task 5: Creating a broker on AIX

To create a broker, enter the following command:

```
mqsicreatebroker MQSI_SAMPLE_BROKER -i mqsiuid -a mqsipw
-q MQSI_SAMPLE_CONFIG_QM -n MQSIBKDB
```

This identifies the broker (MQSI_SAMPLE_BROKER), the broker’s queue manager (flag `-q`) and specifies the database that has been created for the broker tables (flag `-n`). It also identifies the user ID (flag `-i`) and password (flag `-a`) that the broker will run under.

The service user ID and password are also used as the user ID and password for the broker database. If you want to use a different user ID (flag `-u`) and password (flag `-p`) for access to this database, you must specify them here and grant the user ID access to the database (described in “Task 2: Setting up database authorizations” on page 30).

When you type the password on the command line, it appears on the screen exactly as you type it.

If you are using different names or values for any parameter on this command, you **must** replace the appropriate values with your own.

The command might take a short while to complete. The expected responses generated by this command are:

```
MQSeries queue manager created.
Creating or replacing default objects for MQSI_SAMPLE_CONFIG_QM.
Default objects statistics : 29 created. 0 replaced. 0 failed.
Completing setup.
Setup completed.
MQSeries queue manager 'MQSI_SAMPLE_QM' started.
The setmqaut command completed successfully.
The setmqaut command completed successfully.
The setmqaut command completed successfully.
The setmqaut command completed successfully.
The setmqaut command completed successfully.
The setmqaut command completed successfully.
The setmqaut command completed successfully.
The setmqaut command completed successfully.
security properties not found. using defaults.
BIP8071I: Successful command completion.
```

If the command detects any errors, or is unable to complete, it returns an error message on the command line, or in the syslog, which includes the explanation and action in full. The error might have been caused by another component that MQSeries Integrator interacts with to complete this command (DB2, or MQSeries), so check for errors from these products too.

On completion, the following have been done:

-
- The broker MQSI_SAMPLE_BROKER has been created.
- The queue manager MQSI_SAMPLE_QM has been created and started.
- The MQSeries resources required by the broker have been defined. These resources are detailed in “Default MQSeries resources” on page 58.

- The required authorizations for the MQSeries resources have been set (the *setmqaut* messages seen in the responses to the command).
- The database tables required by the broker have been set up in the database MQSIBKDB. These tables are listed in Table 5 on page 57.

Task 6: Connect the Windows NT and AIX queue managers

Most of the resources you need to support this simple configuration have been created for you when you invoked the create broker and create Configuration Manager commands. The next stage is to enable the Configuration Manager to communicate with the Broker.

Define the channels between the Configuration Manager on Windows NT and the broker on AIX.

On AIX:

Define the sender transmission queue and channel. Replace `config.machine.name` with the TCP/IP hostname of the Windows NT machine upon which the configuration manager resides.

```
runmqsc MQSI_SAMPLE_CONFIG_QM
define qlocal(MQSI_SAMPLE_CONFIG_QM) usage(XMITQ)
define channel(TO.MQSI_SAMPLE_CONFIG_QM) chltype(SDR) trptype(TCP)
 conname('config.machine.name(1414)') xmitq(MQSI_SAMPLE_CONFIG_QM)
define channel('CONFIG.BROKER') chltype(RCVR)
end
```

On Windows NT:

Define the sender transmission queue and channel. You can use the MQSeries Explorer program rather than the runmqsc command program to define the queue and channels. Replace `config.machine.name` with the TCP/IP hostname of the AIX machine upon which the broker resides.

```
runmqsc MQSI_SAMPLE_QM
define qlocal(MQSI_SAMPLE_QM) usage(XMITQ)
define channel(TO.MQSI_SAMPLE_QM) chltype(SDR) trptype(TCP)
 conname('broker.machine.name(1415)') xmitq(MQSI_SAMPLE_QM)
define channel('BROKER.CONFIG) chltype(RCVR)
end
```

Start the MQSeries Listeners

Start the MQSeries listener on AIX by issuing the command:

```
runmqlsr -t tcp -p 1415 -m MQSI_SAMPLE_QM
```

Start the MQSeries listener for the Configuration Manager on Windows NT by using one of the following two methods:

1. You are recommended to use MQSeries Services (*Start->Programs->IBM MQSeries->MQSeries Services*). Expand the left-hand pane and find and left-click the queue manager (MQSI_SAMPLE_CONFIG_QM) to

display its services in the right-hand pane. If the Listener is listed, right-click the Listener, and select *All Tasks->Start*. This starts the listener as a background task.

If the Listener is not listed, right-click the queue manager and select *New->Listener*. This creates a listener with default properties of transport type TCP and port 1414. When it has been created, right-click the Listener and select Start.

This starts the listener as a background task.

2. If you prefer, you can use the following command on the command line:

```
runmqlsr -t tcp -p 1414 -m MQSI_SAMPLE_CONFIG_QM
```

When you use this command the listener is started as a foreground task and is not displayed in the MQSeries Services window.

Note: If the default MQSeries port 1414 is not available (perhaps because it is already in use by another queue manager), you must assign a different port number that is suitable. The port value must be set in the Listener properties dialog (Parameters tab), or as the -p parameter on the runmqlsr command. You must update the channel definitions to reflect any changes in the port numbers used. If the port is already in use, the Control Center will not be able to contact the Configuration Manager. For example, if you have set up a default queue manager on this system, it probably already has a listener started on this port. You can check what listeners are already active using MQSeries Services.

Start the Channels

Start the sender channels on each queue manager using the following commands.

On the Broker Queue Manager on AIX:

```
runmqchl -c >BROKER.CONFIG> -m MQSI_SAMPLE_QM
```

On the Configuration Queue Manager on Windows NT:

```
runmqchl -C >CONFIG.BROKER> -m MQSI_SAMPLE_CONFIG_QM
```

Task 7: Start the Configuration Manager and the Broker

On AIX start the broker by typing:

```
mqsistart MQSI_SAMPLE_BROKER
```

You must check the syslog to ensure that the broker has initialized successfully. BIP2001I indicates it has initialized.

On Windows NT start the Configuration Manager by typing:

```
mqsistart configmgr
```

This command initiates the startup of the Configuration Manager's Windows NT Service and can only report on whether that service has started successfully. If it has, you must check the Application view of the Windows NT Event Viewer to ensure that the Configuration Manager has initialized successfully.

Task 8: Starting the Control Center

Start the Control Center by double-clicking the Control Center icon in the MQSeries Integrator program folder, or by using the Windows NT Start menu (*Start->Programs->IBM MQSeries Integrator 2.0.1-> Control Center*). Complete the following tasks to set up the environment you need to complete the simple verification described in "Chapter 5. Verifying your installation" on page 41.

This section gives only a minimum of information required to complete your initial broker domain setup. For further information about the Control Center, refer to *MQSeries Integrator Using the Control Center*.

Step 1. Complete the initial dialog presented by the Control Center, *Configuration Manager Connection*, to provide the information needed to connect your Control Center session to the Configuration Manager. The fields are:

a. Hostname. This is initially blank. Enter the network hostname of the system on which the Configuration Manager has been created. In the simple configuration defined in this chapter, the value you must enter here is MQSISYS1. If you are using a different host name, enter your value here.

b. Port. This is initially blank. Enter the number of the port on which the queue manager is listening. You set this up in "Start the MQSeries Listeners" on page 36:. The default is 1414).

c. Queue Manager name. This is initially blank. Enter the name of the queue manager (MQSI_SAMPLE_CONFIG_QM). This queue manager already has a definition for the server connection required by the Control Center (the channel SYSTEM.BKR.CONFIG of type SVRCONN), which was created when the Configuration Manager was created.

When you have completed these fields, click **OK**. The Control Center now contacts the Configuration Manager, which might take a few minutes. If the Control Center fails to make contact, the most likely reasons are:

- The Configuration Manager has not started successfully.
- The listener has not started successfully.
- The queue manager is not available.

- You are logged on to the local security domain, but this user ID is not a member of the MQSeries Integrator groups. Check the groups of which your current user ID is a member. Also, check that you are logged on to the same security domain as the one you were logged on to when you installed MQSeries Integrator.

Check for MQSeries or MQSeries Integrator entries in the Windows NT Event log (Application view) to track down the problem.

If you want to check, or change, these settings at a later time, click *File->Connection* to bring up the connection dialog.

Task 9: Connect the Broker to the Domain

By connecting the broker to the domain, you are adding the definitions required to the Configuration Manager so that it will be able to define flows to the broker. This relies on the Configuration Queue Manager being able to route messages to the Broker Queue Manager using the channels defined.

To add a broker to the domain:

1. Using the Control Center select the `Topology` panel. In the left-hand panel, click with the right mouse button on the `Topology` icon and select checkout. A key symbol should appear to the Topology icon.
2. In the right-hand pane, click with the right mouse button and select `Create->Broker`. This opens a dialog box in which you enter the name of your broker on AIX, ’MQSI_SAMPLE_BROKER’, and the name of the Broker Queue Manager, ’MQSI_SAMPLE_QM’. Select `Finish` when complete.

   When complete, an icon identifying the broker will be created in the right hand panel. However, at this point neither the Configuration Manager nor the Broker has been contacted.
3. Click with the right mouse button on the `Topology` icon in the right-hand pane and select `Check in`. When complete, the key symbol disappears from the Topology icon.

At this stage the Broker definition has been stored in the Configuration Manager database but nothing has been sent to the Broker. To verify the connection between the Configuration Manager and Broker you should run the default execution process against the Broker.

To deploy the empty default execution group to the Broker:

1. Select the `Assignments` tab. In the furthest right-hand pane you will see a definition for the broker, and within the Broker icon an empty box containing the word `default`

   This indicates that there is one execution group defined for the broker and that the execution group contains no definitions.

2. In the far-left panel, click with the right mouse button on the icon identifying your broker and select `Deploy->Complete Assignments Configuration`

   This attempts to send the default execution group definition to the broker. After a short delay, a dialog box is displayed, indicating that the deploy request has been submitted to the Configuration Manager.
3. To see the results of the deploy request you must select the `Log` panel.

   All deployed responses are written to this panel. To refresh the display, click the `Refresh` icon in the top left-hand corner of the panel.

   If the response is successful, the broker on your AIX machine will start two new processes. You will see `'DataFlowEngine'` and `bipservice MQSI_SAMPLE_BROKER` in response to `ps -ef`

   If the response was not successful you should use the messages generated, and errors written, to the Windows Event Log (Applications View) to determine the cause of the error.

You have now:

- Created a Configuration Manager on Windows NT
- Created a Broker on AIX
- Added the broker to the Broker Domain defined by the Configuration Manager

See “Chapter 5. Verifying your installation” on page 41 for information on creating message flows for you to deploy to the AIX broker, and test using the supplied sample applications.

Chapter 5. Verifying your installation

You have now completed the configuration and activation tasks. This chapter explains how to deploy your broker domain, and how to verify your installation. You can choose to run one or more of a set of verification programs that illustrate different aspects of setup and operation:

- “Preparing for verification”.
- “Running the predefined verification applications” on page 44.
  - “Running the Results Service application” on page 45.
  - “Running the Scribble application” on page 47.
  - “Running the Postcard application” on page 49.
- “Building and using a message flow” on page 51.

All the tasks illustrated here assume you have used the sample names and values when you completed the tasks in “Chapter 4. Configuring a broker domain” on page 21. If you have changed any of these names or values, make sure that you use your values in this section.

You complete most of the tasks involved in running these verification programs using the Control Center. This section gives the minimum information you need to complete these tasks. For further information, see *MQSeries Integrator Using the Control Center*.

Preparing for verification

Before you can run any of the verification programs, you must complete some preparation.

Creating the MQSeries resources on AIX

The verification applications require local queues on the broker’s queue manager. This step creates the MQSeries queues needed by the applications. The queues are:

- For the Soccer application:
  - MQSI_SOCCER_PUBLICATION_QUEUE
  - MQSI_SOCCER_SUBSCRIPTION_QUEUE
- For the Scribble application:
  - MQSI_SCRIBBLE_PUBLICATION_QUEUE
  - MQSI_SCRIBBLE_SUBSCRIPTION_QUEUE
- For the Postacrd application:
  - MQSI_POSTCARD_INPUT_QUEUE
  - MQSI_POSTCARD_OUTPUT_QUEUE

An MQSC command file is provided to define these resources. The file is in the `sample/mqsc` subdirectory under the MQSeries Integrator home directory. From the command line change to this directory and type the following:

```
runmqsc MQSI_SAMPLE_QM < samples.tst
```

Importing and deploying the MQSeries Integrator resources on Windows NT

You must now work with the MQSeries Integrator resources that are used by the applications.

1. Ensure the Control Center is active and select the *Topology* view. Check out the broker domain topology by selecting the topology, right-clicking, and selecting *Check Out*. This locks the topology and allows you to make changes to it.
2. The title bar currently shows that you have an empty workspace, by displaying *Untitled*. You must import the supplied workspace import file that defines the resources used by the verification programs.
   a. Select *File->Import*. The Import dialog is displayed. This allows you to select the type of resources you want to import, and the file that contains the resource definitions.
   b. The valid resource types to import are:
      - Message flows
      - Topics
      - Topology

      The file supplied by MQSeries Integrator contains message flow and topology definitions, so you must select these two types.
   c. Click **Browse** and locate the `\examples` subdirectory (in the MQSeries Integrator home directory). Select the sample workspace import file `SamplesWorkspaceForImport`, click **Open**, and then click **Import**.

      You see a dialog box asking you if you want to save. Select **No**.

      The definitions can take a few minutes to import. When import has finished, a message dialog is presented, confirming that the resources have been imported successfully. Click **OK** to dismiss the dialog. You now see the sample broker (MQSI_SAMPLE_BROKER) in the topology.
   d. Select the *Message Flows* view. Check that the import has created two new folders of message flows, *Verification message flows* and *IBM Default message flows*. The default message flows are described in *MQSeries Integrator Introduction and Planning*. The default message flow folders are in addition to the *IBMPrimitives* folder. If you expand the tree for the verification message flows you can see three new message flows, one for each of the verification programs. They are *ScribbleInversion*, *Soccer*, and *Postcard*.

Note: You will see the *Postcard* message flow in the view. This sample is platform-independent.

3. Check that the message flows have already been assigned to the broker's default execution group (this happened when you imported the workspace); they should be showing in the Domain hierarchy and Topology panels.
4. Now save the changes that you have made. Select *File*->*Check in*->*All (Save to Shared)*. This causes two things to happen:
   a. The contents of the configuration repository are updated with the new definitions and assignments and everything is checked into the repository.
   b. The new workspace is saved locally. Because this is a new workspace, you are asked for a name for this workspace. Enter a name, for example *SampleWorkspace*, and click **Save**. This name now appears in the title bar.
5. Now deploy your changes to the broker. When you deploy, the Configuration Manager sends information to the broker about the resources it needs to support the message flow services.
   a. Select the *Topology* view.
   b. Select *File*->*Deploy*->*Complete Configuration (all types)*->*Normal*, or right-click the Topology root and select *Deploy*.

      A message dialog confirms initiation. Select **OK** to dismiss the dialog.

6. Select the *Log* view and refresh the contents by clicking the green refresh icon. It can take a few minutes for all 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. If everything is successful, the log contents appear with text that is similar to the following:

   ```
   **********************************************************************
   
   This message is generated at 2000-10-23 10:58:12
   
   BIP2056I: Broker MQSI_SAMPLE_BROKER successfully processed the
   entire internal configuration message.
   
   An internal configuration message was processed to completion.
   
   No user action required.
   
   ***********************************************************************
   
   This message is generated at 2000-10-23 10:58:12
   
   BIP4040I: The Execution Group 'default' has processed a
   configuration message successfully.
   
   A configuration message has been processed successfully.
   Any configuration changes have been made and stored persistently.
   
   No user action required.
   
   ***********************************************************************
   
   This message is generated at 2000-10-23 10:58:14
   
   BIP4045I: Configuration changed successfully.
   
   The message broker received a configuration message and updated
   its configuration accordingly.
   This change concerned its publish-subscribe capability.
   
   No user action required.
   ```

7. View the deployed configuration graphically in the *Operations view* Refresh the view, and the topology view is displayed.

Running the predefined verification applications

This section describes how to run each of the three applications supplied with MQSeries Integrator. You can run any of these, in any order, immediately after installation or at any time in the future. If you choose to run these applications later, make sure you have your system set up in the same way as the system you configured in “Chapter 4. Configuring a broker domain” on page 21 (or make the appropriate adjustments as you follow these steps).

The verification applications also illustrate how MQSeries Integrator can be used to transform and route messages outside the programming logic of the participating applications, which can therefore run unaffected by updates to that transformation logic, or routing logic, or both.

Running the Results Service application

The Results Service application is written in the C programming language and demonstrates a number of basic publish/subscribe features. The application is a simple implementation of a soccer match results gathering service. It consists of one or more publisher applications, and one subscriber application. You can find the files that make up this application (source, header files, and executables) in the `sample\Soccer` subdirectory in the MQSeries Integrator home directory.

Run this application on AIX as follows:

1. Start the subscriber application **soccerResults**.

   You must start a single subscriber that subscribes to all soccer matches being played, and displays the results for them. The subscriber application functions as a results server. You must start soccerResults before you start any instances of the publisher application, so that the results server does not miss any publications.

   You can start the Results Service as follows:

   a. Change to the `/usr/opt/mqsi/sample/soccer/bin` directory.

   b. Enter the command

      ```
      soccerResults MQSI_SAMPLE_QM
      ```

   A message is displayed by the results server indicating that it has registered a subscription and started successfully, and you can now start the match simulator (publisher).

2. Start the publisher application, **soccerGame**.

   You can run one or more publishers. Each instance publishes event publications on a single soccer match. You must specify two soccer teams as input to soccerGame.

   Start the publisher application as follows:

   a. Open a command window and change to the `/usr/opt/mqsi/sample/soccer/bin` directory.

   b. Enter the command to start up a soccer game. You can use the "_" character to represent a space in the name of a team:

      ```
      soccerGame Team1 Team2 MQSI_SAMPLE_QM

      soccerGame Arsenal Manchester_United MQSI_SAMPLE_QM
      ```

      Note: Team names can contain only the characters 0-9, a-z, and A-Z.

How the Results Service works

The Results Service application uses messages that have a standard MQSeries header, an MQRFH2 header, and a string that specifies the playing teams and their scores.

The soccer simulator **soccerGame** publishes an event publication following this message template to the queue MQSI_SOCCER_PUBLICATION_QUEUE on the broker's queue manager. The MQInput node in the *Soccer* message flow has been set up so that it identifies this queue as its input queue.

The input node retrieves the publication from this queue and forwards it to the publication node. The publication can indicate that:

- A match has started.
- A goal has been scored.
- A match has ended.

The results server **soccerResults** subscribes to all these event publications arriving on queue MQSI_SOCCER_SUBSCRIPTION_QUEUE. It processes these messages and displays the information: the start of a new game, a score update, and the end of a game.

One important feature of the soccer simulator **soccerGame** is its ability to maintain a current state of all the matches being played (the multiple publishers). It achieves this by publishing a retained publication message to the broker with the latest score of each match every time the score changes. This means you can restart the results server after a failure, and the results server subscribes to all these retained publications to restore the current match state to the state it had the last time the results server was running.

If you want to see this use of retained publications, you can start several instances of the publisher application **soccerGame**. When these are running, and a couple of goals have been scored, change to the window running the results server application and prematurely stop that process[1] using Ctrl-C.

Wait about 30 seconds, then restart the results server **soccerResults**. You will see that the matches being played are restored to their last known score, and updated by any remaining match changes that occurred whilst the results service was stopped.

If you restart the results server too quickly, it might fail to open the subscriber queue with reason 2042 (MQRC_OBJECT_IN_USE). This is because the queue manager has not yet recognized that the application has failed, and has

[1] If you have created your broker to run as an MQSeries trusted application, you must not end this application in this way, because the queue will not be released. For more information about MQSeries Integrator and MQSeries trusted applications, see *MQSeries Integrator Introduction and Planning*.

therefore not released the queue that the application opened exclusively. You can retry the restart after a few seconds; once the queue is available it will succeed.

See the *MQSeries Integrator Programming Guide* for more details about the implementation of this application and the publish/subscribe techniques it uses.

Running the Scribble application

The Scribble application is written in Java and demonstrates a number of basic publish/subscribe and message transformation features. In contrast to the Results Service, which works with multiple publishers and one subscriber, Scribble works with one publisher and any number of subscribers.

The publisher publishes the current coordinates of the line being drawn in its window, and each subscriber receives the inverted coordinates and displays the resulting drawing in its window.

You must have:

- MQSeries Client for Java on the AIX machine.
- the efix as described in the `readme.txt`.
- environment variables correctly set in the profile. Refer to `readme.txt` for information on these.

Run this application on AIX as follows:

1. Start the publisher application.
   a. Change to the `/usr/opt/mqsi/sample/scribble/classes` subdirectory
   b. Run `java Scribble &`

   You now see a dialog that prompts you for the broker queue manager name. Enter MQSI_SAMPLE_QM and click **OK**. A confirmation dialog, *Scribble ready*, is displayed. Click **OK**. The publisher window is displayed.
2. Start the subscriber application.
   a. Change to the `/usr/opt/mqsi/sample/scribble/classes subdirectory`
   b. Run `java ScribbleListen &`.

   You now see a dialog that prompts you for the queue manager name. Enter MQSI_SAMPLE_QM. The dialog also allows you to enter a queue name. If you want to use the default subscriber queue, MQSI_SCRIBBLE_SUBSCRIPTION_QUEUE, you do not have to specify this. If you are using a different queue, you must define that queue and enter the name here. Click **OK**. The subscriber window is displayed.

3. Start dragging the mouse with either mouse button depressed to draw lines in the publisher window. These lines appear inverted in your subscriber window.

You can start multiple Scribble subscribers, but you must specify a different queue for each one. The definitions you completed in “Creating the MQSeries resources on AIX” on page 41 contain a single subscriber queue for this application, the default subscriber queue MQSI_SCRIBBLE_SUBSCRIPTION_QUEUE. If you want to start additional subscribers, define additional queues like the default one, and enter the queue name as well as the queue manager name at the dialog you see when you first start the subscriber.

How Scribble works

The Scribble application uses messages that have a standard MQSeries header, an MQRFH2 header, and a message body formatted in XML that specifies the drawing coordinates. When you drag the mouse across the publisher window with a mouse button depressed, it draws a line that the publisher records as a set of coordinates. It publishes each set of coordinates in an XML message to the publication queue MQSI_SCRIBBLE_PUBLICATION_QUEUE. The MQInput node in the *ScribbleInversion* message flow has been set up so that it identifies this queue as its input queue.

The input node retrieves the publication from this queue and the message flow inverts the drawing by manipulating the coordinates (transformation), and publishes the resulting drawing (routing) to each ScribbleSubscriber’s subscription queue (MQSI_SCRIBBLE_SUBSCRIPTION_QUEUE is the default).

The details of the transformation and routing performed by the *ScribbleInversion* message flow are:

- Receive the published message in the MQInput node.
- Filter on the publish / subscribe topic *scribble/coord* in the **FilterOnTopic** node. This node is of primitive type Filter.

  If the match is successful, the message is passed to the **InvertCoordinates** node, which is of primitive type Compute, for transformation.

  If no match is found, the message is sent directly to the publication node without inversion.

You can see the SQL code that implements the inversion of the coordinates in the message. Select the *Message Flows* view in the Control Center, select *ScribbleInversion* in the tree in the left pane, click the **InvertCoordinates** node with the right mouse button to display the node’s context menu, and select *Properties*.

Running the Postcard application

The Postcard application is based on the MQSeries for Windows NT Version 5.1 Postcard application, and has been extended to demonstrate the transformation capabilities of MQSeries Integrator Version 2.0.1. It is written in C with a Java end-user interface.

Postcard allows you to send a postcard to another nickname, either known to this instance, or to a different instance, of the application. You must run this application on the same system as the broker. You can find the files that make up this application (source, header files and executables) in the `examples\postcard` subdirectory under the MQSeries Integrator home directory.

Run this application by following these steps:

1. Start the first Postcard application.

   This first instance acts as the sending application. Select *Start->Programs->IBM MQSeries Integrator 2.0->Samples ->Postcard->Postcard*.

   You now see a dialog that prompts you for a nickname to use for sending or receiving messages. Enter an alphanumeric string of up to 24 characters. The dialog also asks you for the name of the broker queue manager. This is an optional field, but if you do not enter a queue manager name, the default queue manager is used. Enter MQSI_SAMPLE_QM to specify the correct queue manager for verification. Click **OK**. A Postcard window appears.
2. Start the second Postcard application.

   This second instance acts as the receiving application. You must enter a second nickname and the name of the broker queue manager. Click **OK**. A second Postcard window appears.
3. Fill in the postcard and send the message.

   In the sender (first) Postcard window, fill in the **To** field with the nickname of the receiving (second) postcard application. Fill in the remaining fields from the pulldown menus to build the content of the postcard (location, length of stay, and weather). Click the **Send** button. You see the message, marked *Sent*, in the box called “Postcards sent and received (transformed)” in the lower part of the Postcard window.
4. View the received postcard message.

   You will see the postcard arrive in the second (receiving) Postcard application. Select the received message in the list and click **View** to see the contents of the (received and transformed) message. The message has been transformed to include the country of the city from which the message was sent.
5. Return a postcard to the original sender.

From the second Postcard application, select a message from the "Postcards sent and received (transformed)" box and click **Reply**. This gives you a new postcard to fill in, with the first application's nickname already in the **To** field. Fill in the remaining fields and click **Send** to send the new postcard to the first application.

How Postcard works

The Postcard application sends messages to, and receives messages from, the *Postcard* message flow. You deployed this message flow to the broker in "Preparing for verification" on page 41. You also stored the message set *PostcardMS* in the message repository, and deployed it to the same broker. The message set is referenced by the message processing nodes within the flow. It contains one message, *PostcardMessage*, that defines these elements:

1. **Location** (of type STRING)
2. **Country** (STRING)
3. **MessageText** (STRING)
4. **Duration** (INTEGER)
5. **Recipient** (STRING)
6. **Sender** (STRING)
7. **GoodTime** (INTEGER)
8. **Weather** (STRING)

The elements of type STRING each have an associated element length that defines the maximum number of characters valid in this element.

The application program creates and interprets the messages based on a structure defined in the C header file `postcardstruc.h` in the `examples\postcard` subdirectory. This header is an identical representation of the message in the message set in the Control Center.

When a user sends a postcard, the application puts a message to the queue associated with the *Postcard* message flow (MQSI_POSTCARD_INPUT_QUEUE). The message flow provides the following message processing:

- The MQInput node retrieves messages from the input queue MQSI_POSTCARD_INPUT_QUEUE.
- The input node passes the message to node **AddCountry**, an instance of the IBMPrimitive Compute node. This node enhances the content of the message by adding the `Country` field, containing the country of the location you selected when you sent the postcard (for example, if you selected "Adelaide", it adds "Australia").
- The **AddCountry** node now passes the message to the MQOutput node, which puts it to the output queue (MQSI_POSTCARD_OUTPUT_QUEUE).

- The receiving postcard application retrieves the message from the output queue, reads and interprets the content, and displays the element content in the corresponding fields of the Postcard user interface window.

Although this application has a message set already defined (see “Importing and deploying the MQSeries Integrator resources on Windows NT” on page 42), you can create a message set based on a C header file (like `postcardstruc.h`) by importing that C structure into the message repository using the import function of the Control Center. This is explained in detail in *MQSeries Integrator Using the Control Center*.

Note: Do not try this after importing PostcardMS; this would create duplicate entries and cause problems with deployment.

Building and using a message flow

This verification scenario illustrates how you define a basic message flow, how you assign the resources to the broker, and deploy your changes. It uses MQSeries Explorer to send messages through the message flow you create. It does not use any defined message sets. It assumes you are using the sample broker that you created in “Task 5: Creating a broker on AIX” on page 34. If you want to use another broker, you must ensure that you create it (using **mqsicreatebroker**) and define it in the configuration repository (from the *Topology* view in the Control Center).

The tasks assume that the broker, Control Center, queue managers, listeners, channels, and configuration manager are running.

The following tasks are described:

- “Creating the MQSeries resources”.
- “Creating a message flow”.
- “Assigning the message flow to the broker” on page 52.
- “Testing the message flow” on page 53.

Creating the MQSeries resources

This verification test needs two queues, one for input, the other for output. This section shows how to use MQSC to perform these definitions.

```
runmqsc MQSI_SAMPLE_QM
define qlocal ('MQSI_INQ')
define qlocal ('MQSI_OUTQ')
```

Creating a message flow

You must now create the message flow that will process the messages you put to your input queue. The message flow is very simple: the processing it does is to retrieve the message from the input queue and put it to the output queue!

A simple message flow

1. Select *File->New Workspace* to create a new (untitled) workspace. If you have run the verification applications, and have your sample workspace already open, you can create your new message flow in this workspace if you prefer.
2. Select the *Message Flows* view.
3. Right-click on the Message Flows root and select *Create->Message flow*.
4. Enter the name MQSI_TEST. Click **Finish**. The new message flow appears in the tree view.
5. Expand the IBMPrimitives tree to display the supplied nodes.
6. Select the MQSI_TEST message flow in the left-hand pane. Drag and drop an MQInput node into the right-hand pane.
7. Right-click the MQInput node in the right-hand pane and select *Properties*. On the Basic tab, type the MQSeries input queue name of your input queue (MQSI_INQ). Click **OK**.
8. Drag and drop an MQOutput node into the right-hand pane.
9. Right-click the MQOutput node in the right-hand pane and select *Properties*. On the Basic tab, type in the queue manager name (MQSI_SAMPLE_QM) and the queue name (MQSI_OUTQ) for the output queue. Click **OK**.
10. Right-click the MQInput node and select *Connect->Out*. This gives you a connector attached to your mouse. Drag this to the MQOutput node and drop by left-clicking. The connector attaches itself to the input terminal.
11. You have now completed your first message flow. Select *File->Check in->All (Save to Shared)*. This checks in all the resources to the configuration repository and saves a local copy of the workspace file. If you created a new workspace for this new message flow, you will be prompted to give the workspace a name when you save it.

Assigning the message flow to the broker

Now you have created a message flow, you have to tell MQSeries Integrator where you want to run that message flow (that is, on which broker). To do this, you must assign the message flow to your broker.

1. Select the *Assignments* view.
2. Expand the broker name (MQSI_SAMPLE_BROKER) to display the broker's execution groups. The sample broker currently just has one execution group, the default one (called *default*) which is always created whenever you create a broker using **mqsicreatebroker**.
3. Right-click the default execution group and select *Check Out*. This locks the execution group for you.
4. Expand the *Message flows* tree in the center pane. This displays all the message flows available for assignment.

5. Find MQSI_TEST and drag and drop it on the default execution group in the right-hand pane, where you can see the graphic of the broker and the default execution group. You can drop a message flow only on an execution group (not on the broker itself).
6. Check in the execution group by right-clicking the default execution group in the left-hand pane and selecting *Check In*.

Deploying the message flow to the broker

Assignment makes the connection between a message flow and a broker, but it is only when you deploy the change that the Configuration Manager updates the broker with the configuration stored in the configuration repository.

1. Before you can deploy any changes, you must have checked in everything that you have updated. If you have followed the instructions in this section, all the relevant resources are checked in. However, if you are in any doubt, you can check everything in by selecting *File->Save to Shared*.
2. In the *Assignments* view, right-click the broker name in the left-hand pane.
3. Select *Deploy->Complete Assignments Configuration*. When the Configuration Manager receives this request from the Control Center, it sends messages to the broker to give it the updated information it needs to be able to support your new message flow.
4. Check the deploy by changing to the *Log* view and clicking the refresh button (the green icon above the log pane). Check for success messages. (There might be a slight time delay before the messages appear.)
5. View the deployed configuration graphically in the *Operations* view. Refresh this view and the broker, execution group and message flow are displayed with green lights, to show they are all active.

Testing the message flow

You can use MQSeries Explorer to test your message flow.

1. Change to the directory containing the MQSeries sample programs: `/usr/mqm/samp/bin`
2. Enter `amqsput MQSI_INQ_MQSI_SAMPLE_QM`
3. Press **enter**.
4. Enter the text of the message and press **enter** twice to quit the program.
5. Use

   ```
   amqsget MQSI_OUTQ MQSI_SAMPLE_QM
   ```

   or

   ```
   amqsbcg MQSI_INQ_MQSI_SAMPLE_QM
   ```

   to get the message you typed in.
6. Your test is complete.

A simple message flow

Appendix A. System changes after installation

This appendixdescribes the changes that installation and configuration have made on the systems you have set up in your broker domain. It assumes that you have followed the guidance and details of configuration given in “Chapter 4. Configuring a broker domain” on page 21.

- “Directory structure”.
- “Environment variables” on page 56.
- “Database contents” on page 57.
- “Default MQSeries resources” on page 58.

The product’s home directory, which defaults to

```
/usr/opt/mqsi
```

is written as <mqsi_root> wherever it appears in this chapter.

DB2’s home directory defaults to

```
/usr/lpp/db2_06_01
```

Directory structure

The following tables list the subdirectories created and populated within your home directory <mqsi_root> when you do a full installation. It also provides a brief description of the contents. If you have not done a **Full** install, your directory structure will be a subset of those shown.

All files are installed with default security: all users can access and execute these files. You can use standard operating system facilities to impose stricter security on these files, or a subset of files, if you choose.

Table 3. The /usr/opt/mqsi directory structure after installation

Directory Name	**Contents**
bin	Executable files
classes	Java class files
CmdAsst docs	Command Assistant files PDF files for V2.0.1 and V1.1
icu	data-conversion tables
include	Header and other files for samples
lib	Shared library files

Directory structure

Table 3. The /usr/opt/mqsi directory structure after installation (continued)

Directory Name	Contents
lil	Library files
merant	ODBC drivers
messages	Description files for messages and exceptions
sample	C, C++ and Java language samples at V1.1 and V2.0.1 level, MQSC samples
template	files used when creating a broker
tivready	files to support Tivoli Management Systems

Table 4. Additional directories in /var/mqsi

Directory Name	Contents
brokers	Broker configuration and data files
lib	Libraries
log	Log files
odbc	Sample .odbc.ini file
registry	MQSeries Integrator configuration files
users	Executables specific to users

Environment variables

The following environment variables will need to be set:

- CLASSPATH
- LC_MESSAGES
- LD_LIBRARY_PATH
- MQSI_PRELOAD
- MQSI_PARAMETERS_FILE
- NLSPATH
- ODBCINI
- PATH

To ensure you get the latest environment variables you must refer to the `readme.txt` on the main product CD.

A sample profile is shipped with this product, in the following directory:

```
/opt/mqsi/sample/profiles/profile.sol
```

The definitions of the rules and formats in the database identified in the `MQSIruleng.mpf` configuration file are needed by every broker in which you deploy a message flow that includes NEONRules or NEONFormatter message processing nodes.

These definitions are not distributed through your broker network in the same way as the formats and rules defined by MQSeries Integrator Version 2.0.1 Control Center. You must therefore ensure that the brokers that need to access these definitions can do so:

- Ensure that the system on which the broker is installed has client access to the system on which the database is installed.
- Make sure that the file identified in the `MQSI_PARAMETERS_FILE` environment variable contains the correct information to connect to the database. See the *MQSeries Integrator Version 1.1 System Management Guide* for more details about the contents of this file.

You also need to ensure that NEON is set to `/opt/mqi110/lib`

You are strongly recommended to set the broker identifier to be the same value as the broker service identifier to prevent compatibility problems.

Database contents

When you create MQSeries Integrator resources following installation, database tables are created for your broker. It is labelled USERSPACE1. This database can be individually created for each set of tables, or can be shared.

Table 5 shows the tables that are created by the **mqsicreatebroker** command in the broker database. The tables are created when you create the first broker. When you create further brokers specifying the same database, new rows are created for each broker. Every row created in the table includes the broker name; therefore each row is unique to a single broker.

You must add all these database tables into your standard backup and recovery routines to ensure you can recover from system failures and other emergencies.

Table 5. Database tables for brokers

Table name	Description
BACLENTRIES	ACL entries
BCLIENTUSER	Maps client identifiers to durable subscriptions
BGROUPNAME	Publish/subscribe principals: groups
BLOGICALTOPHYSNAME	Maps logical to physical names

Table 5. Database tables for brokers (continued)

Table name	Description
BMQPSTOPOLOGY	Publish/subscribe neighbor information
BNBRCONNECTIONS	Inter-broker neighbor connection information
BPHYSICALFILE	Physical file mapping
BPUBLISHERS	Registered publishers
BRETAINEDPUBS	Retained publications
BRMCONFIG	Broker configuration details
BROKERAA	Broker process details to support recovery
BROKERAAEG	Execution group details to support recovery
BROKERRESOURCES	Broker resources
BSUBSCRIPTIONS	Durable subscription information
BTOPOLOGY	Inter-broker neighbor information
BUSERCONTEXT	Maps client identifiers to context information
BUSERMEMBERSHIP	Publish/subscribe principals: membership
BUSERNAME	Publish/subscribe principals: users
BWFFRELATIONSHIP	Workfile details

These tables are maintained by processes that are internal to MQSeries Integrator components. You must not access these tables by any other means, nor change the access authority required by MQSeries Integrator.

Default MQSeries resources

When you create MQSeries Integrator components, some MQSeries resources are created for their use. Table 6 on page 59 lists all these MQSeries resources, and indicates the component associated with the queue manager on which they are created. For details of which resources are created by which create commands, see the command descriptions in the *MQSeries Integrator Administration Guide*.

All these resource names start with reserved characters “SYSTEM”. Therefore you should not find any conflict of names. There is one exception to this: if you have been using MQSeries Publish/Subscribe, it defines queue SYSTEM.BROKER.CONTROL.QUEUE, which MQSeries Integrator also uses. However, the use is compatible and you do not have to take any action to continue using this queue.

Table 6. MQSeries Integrator default objects

Resource name	Type	Queue manager	Description
SYSTEM.BROKER.ADMIN.QUEUE	queue	broker	Target for messages sent by the Configuration Manager and commands to modify the broker's configuration and operation.
SYSTEM.BROKER.CONTROL.QUEUE	queue	broker	Target for publish/subscribe control requests from applications. A queue of this exact name is used by the MQSeries Publish/Subscribe. You might therefore already have defined a queue of this name on the queue manager. You can continue to use this same queue as you migrate to MQSeries Integrator.
SYSTEM.BROKER.EXECUTIONGROUP.QUEUE	queue	broker	Target for messages to the broker.
SYSTEM.BROKER.EXECUTIONGROUP.REPLY	queue	broker	Target for response messages for the broker from the User Name Server.
SYSTEM.BROKER.INTERBROKER.QUEUE	queue	broker	Target for publications from neighbor brokers.
SYSTEM.BROKER.MODEL.QUEUE	queue	all	Model for dynamic response queues.
SYSTEM.BROKER.SECURITY.QUEUE	queue	User Name Server	Target for request messages to the User Name Server. queue is used by brokers, the Configuration Manager, and the command line tools.

Note: These resources are defined in addition to the MQSeries product default objects, which are defined when the MQSeries Messaging product is installed. You can find a full description of these default objects in *MQSeries System Administration*.

Appendix B. Setting up an Oracle8 broker database on MQSeries Integrator

This chapter describes how to set up an Oracle8 broker database on MQSeries Integrator and is intended for an Oracle DBA.

You are recommended to create a new database instance. Although it is possible to use any existing database instance, you are **not** recommended to do this.

These instructions assume that:

- The database instance will be created following Oracle documentation.
- Oracle communications will be set up using SQL*Net following Oracle documentation.
- Oracle operations are conducted within the correct Oracle environment. For example, ORACLE_HOME.

Naming

MQSeries Integrator uses the Data Source Name (DSN) defined in the ODBC setup. You do not have to use a particular named instance for use as a broker database.

Schema

MQSeries Integrator does not demand a particular schema or set of tablespaces for keeping broker information. When you create a broker, the tables are created in the database with an ownership defined by the user identifier specified on the command line (the user ID must already be known to the database). For example:

```
mqsicreatebroker BRK -i bid - a bpw -q QM -n BDB -u dbid -p dbpw
```

creates tables all owned by `dbid` (like DBID.BROKERAA). The tablespace used to hold these tables is the default tablespace for the Oracle user ID specified (the default is normally SYSTEM).

Sizing

When you create a broker, MQSeries Integrator does not generate much data to be stored in the database. Deployment of a complex flow to the broker can consume more database space but still not a large amount. If you create an instance specifically for use as a broker database, taking the default settings

Schema

defined by the ’dbassist’ tool should be sufficient for most applications. 50 MB is enough for a custom setup. If a tablespace is set up specifically for use by the brokers, this can be extended at a later date.

User Identifier

The Oracle user ID that you use to store broker information needs the privileges: connect, resource and create table. For example, the following is sufficient:

```
CREATE USER dbid IDENTIFIED BY dbpw;
GRANT CONNECT TO dbid;
GRANT RESOURCE TO dbid;
GRANT CREATE TABLE TO dbid;
```

and optionally:

```
ALTER USER dbid DEFAULT TABLESPACE brktbspc;
```

This user ID can be the same as the operating system ID that will be used to create the broker or it can be specific to the database. If it is specific to the database, use the ’-u’ and ’-p’ flags when you create the broker.

ODBC

1. When you have created and started the database, and SQL*Net has been configured (using listener.ora and tnsnames.ora) and started to check that the database is accessible through the SQL*Net interface. First check that the SQL*Net listener is running by using:

   ```
   lsnrctl status
   ```

   or checking the process list for tnslsnr using:

   ```
   ps -ef | grep tnslsnr
   ```

2. Oracle provides a utility tnsping to check that a configured DSN (Oracle Global Database Name) is accessible. For example:

   ```
   tnsping myDSN
   ```

   However, it is useful to check further and test the user ID using SQL*Plus. For example:

   ```
   sqlplus dbid/dbpw@myDSN
   ```

3. This should connect you to the SQL*Plus application and shows in a process list that the connection is not local (that is, the connection is using SQL*Net). Use:

   ```
   ps -ef | grep myDSN
   ```

One of the entries in the list should be something like:

```
myid 1234 5678 0    10:55:57 ? 0:00 oraclemyDSN (LOCAL=NO)
```

4. Update the MQSI ODBC description file /var/mqsi/odbc/.odbc.ini to add entries for your broker database. At the head of the file, in the [ODBC Data Sources] section, add an entry specifying the DSN that the broker will use. (This might be different from the DSN defined by SQL*Net if required.) For example:

```
[ODBC Data Sources]
 MYBRKDSN=MERANT 3.60 Oracle 8 driver
```

5.

   Create a stanza to define the driver path and Oracle DSN for your broker DSN, for example:

```
[MYBRKDSN]
 Driver=/opt/mqsi/merant/lib/UKor815.so
 Description=Oracle8
 ServerName=myDSN
 EnableDescribeParam=1
 OptimizePrepare=1
```

6.

   You should now be able to create a broker with a command like:

```
mqsicreatebroker BRK -i uid -a pwd -q QM -n MYBRKDSN -u dbid -p dbpw
```

Appendix C. Uninstalling MQSeries Integrator

This appendix gives details of the processes that allow you to uninstall any one or all of the MQSeries Integrator components on AIX.

Before you start

1. Log on as root.
2. Stop any brokers you have running.
3. Stop your User Name Server if you have one.

Uninstalling

To uninstall:

1. Type `smitty`
2. Select

   ```
   Software Installation and Maintenance
     Software Maintenance and Utilities
       Remove Installed Software
   ```

   or use the fastpath command `smitty remove`.
3. In the `Software name` field, enter the components you want to remove.

Notes:

1. Entering `mqsi` in the Software name field removes all of MQSeries Integrator.
2. Pressing `F4` in the Software name field allows you to select the components from a list.
3. The `/var/mqsi` directory and all the files in it are not removed when uninstalling. If you want to remove these files, you must do so manually.
4. The mqbrkrs group is left on the system.

Contacting your IBM Support Center

If you are unable to resolve problems that you find when you use MQSeries Integrator, or if you are directed to do so by an error message generated by MQSeries Integrator, you can request assistance from your IBM support center.

Before you contact them, use the checklist below to gather key information. Some items may not necessarily be relevant in every situation. But you should provide as much information as possible to enable the IBM support center to recreate your problem.

- For MQSeries Integrator:
  - CSDs applied.
  - E-fixes applied.
  - All current trace and error logs, including relevant AIX platform syslog entries or Windows NT Event log. User trace log files at debug level should be obtained for all relevant message flows and should preferably be formatted.
  - A list of the components installed. This should include details of the number of machines and their operating systems, the number of brokers and the machine on which they are running, and the existence and details of any User Name Servers.
  - The file obtained by exporting your workspace. This action is performed from the Control Center; see the *MQSeries Integrator Using the Control Center* manual for details of how to do this.
  - The files obtained by exporting all relevant message sets. This action is performed for each message set by using the `mqsimrmimpexp` command with the -e flag set.
  - A sample of the messages being used when the problem arose.
  - If relevant, the report file from the C or COBOL importer. This is located in the directory from which the file import was attempted.
- For MQSeries:
  - CSDs applied.
  - E-fixes applied.
  - All current trace and error logs, including relevant AIX platform syslog entries or Windows NT Event log and (FFST™) output files. You can find these files, which have the extension FDC, in the errors subdirectory within the MQSeries home directory.
  - Details of MQSeries client software, if appropriate.
- For each database you are using:
  - Product and release level (for example, DB2 6.1).
  - CSDs applied.
  - E-fixes applied.
  - All current trace and error logs, including relevant AIX platform syslog entries or Windows NT Event log and First Failure Support Technology (FFST) output files. Check database product documentation for where to find these files.
- For AIX:

  - Version. Use the `lslpp -l bos.rte` command.
  - Service level applied.
- For Windows NT:
  - Version.
  - Service Pack level.
  - The version of the system files `msvcrt.dll`, `msvcp60.dll`, `msvcirt.dll`, and `mfc42.dll`. You can find these files in the WINNT\SYSTEM32 directory. Use the Windows NT Explorer file properties to display the versions.
- Details of the operation you were performing, the results that occurred, and the results you were expecting.

Appendix D. Applying maintenance

Maintenance updates are supplied on CD in the form of a Program Temporary Fix (PTF), referred to as a Corrective Service Diskette (CSD). You can find the latest information about available CSDs on the Internet, at the address given in “MQSeries information available on the Internet” on page xiii.

You can also download CSDs from this Web site.

Applying maintenance to MQSeries Integrator for AIX

Follow these instructions if you need to apply maintenance to your installation of MQSeries Integrator.

Applying maintenance

1. Read the file memo.ptf in the root directory of the CD, and any readme.txt files also in the root directory. These files might contain additional information about how you must install this maintenance.
2. Ensure that you are logged on as root
3. Ensure that any running components are stopped (that is the broker and the UserNameServer)
4.
   a. Type smitty
   b. Select

      ```
      Software Installation and Maintenance
        Install and Update Software
          Update Installed Software to Latest Level (Update All)
      ```

      or use the fastpath command smitty update_all
   c. Either, enter the directory where the update images are, or press PF4 and select the appropriate device (for example, if installing from CD-ROM)
   d. Decide whether or not you want to commit the update by selecting the appropriate response in the COMMIT software updates field.

      If you answer Yes you will not be able to restore the previous level of the product without uninstalling it. Answering No will backup the replaced files so that you can reject the update and return to the original level of the product, (see “Committing updates” on page 70).
   e. Press the Enter key to perform the update

Restoring a previous service level

You are able to restore a previous service level if you want to, or if you are instructed to do so by your IBM support center. However, this is only possible if you accepted the default action to backup replaced files when you applied the current service level.

Because CSDs are cumulative, you must uninstall the most recent CSD you applied before attempting to uninstall any previous CSDs. For example, if you have installed CSD1, CSD2, and CSD3 on your machine and you want to revert to the CSD1 level of code, you must first uninstall CSD3 and then uninstall CSD2.

Carry out the following procedure:

1. Type `smitty`
2. Select

   ```
   Software Maintenance and Installation
     Software Maintenance and Utilities
       Reject Applied Software Updates (Use Previous Version)
   ```

   or use the fastpath command `smitty reject`
3. In the `SOFTWARE name` field type `mqsi`
4. Press the `Enter` key to remove the update

Committing updates

If you no longer need the ability to restore the product to a previous level, or want to save disk space, you can commit the update. This removes the files you backed up when the update was applied.

To do this on AIX:

1. Type `smitty`
2. Select

   ```
   Software Maintenance and Installation
     Install and Update Software
       Commit Applied Software Updates (Remove Saved Files)
   ```

   or use the fastpath command `smitty commit`
3. In the `SOFTWARE name` field type `mqsi`
4. Press the `Enter` key to commit the update

Checking the service level

After initial installation, the MQSeries Integrator Service level indicates that no service has been applied.

Checking the service level

After one or more updates, the service level is updated to show the CSD most recently applied. The service level is expressed in terms of the PTF number for a particular CSD.

To check the service level on AIX:

1. Type `smitty`
2. Select

   ```
   Software Maintenance and Installation
     Software Maintenance and Utilities
       List Software and Related Information
         List Installed Software and Related Information
           List Installed Software
   ```

   or use the fastpath command `smitty list_installed_sw`
3. In the `SOFTWARE Name` field, enter `mqsi\*`
4. Press the `Enter` key. You will see the level of each MQSI fileset displayed (for example 2.0.1.0, where the last digit is the service level)

Applying maintenance to IBM DB2 Universal Database

If DB2 was installed on this system by the MQSeries Integrator installation program, it is installed with no service applied.

You can also obtain information about the current status of maintenance of this product, and download fix packs for DB2 from the Web site identified in “DB2 publications” on page xiii.

Appendix E. 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
DB2 Universal Database
First Failure Support Technology
MQSeries
POWERserver
SupportPac
DB2
FFST
IBM
POWERparallel
RS/6000
VisualAge

Tivoli is a trademark of Tivoli Systems Inc. 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, Windows NT, and the Windows logo 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, and service names may be trademarks or service marks of others.

Glossary of terms and abbreviations

This glossary defines MQSeries Integration Builder 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 Integration Builder 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 Integration Builder 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*.

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. 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 Integration Builder .

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.

F

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. 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 Integration Builder 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 source of a message definition. The following domains are recognized:

- MRM, which identifies messages defined using the Control Center
- XML, which identifies messages that are self-defining
- BLOB, which identifies messages that are undefined

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 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.

MQI. Message queue interface.

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.

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.

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

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.

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.

system log. A generic term used in the MQSeries Integration Builder 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 Integration Builder component that interfaces with operating system facilities to determine valid users and groups.

utility function. Function provided by MQSeries Integration Builder 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.

Index

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 44–1962–870229
  - From within the U.K., use 01962–870229
- 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.