MQSeries® Integrator for Sun Solaris

Installation Guide

Version 2.0.1

GC34-5842-01

MQSeries® Integrator for Sun Solaris

Installation Guide

Version 2.0.1

GC34-5842-01

Note!

Before using this information and the product it supports, be sure to read the general information under "Appendix F. Notices" on page 95.

Second edition (November 2000)

This edition applies to IBM MQSeries Integrator for Sun Solaris 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 overview of the concepts and capabilities of MQSeries Integrator 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 15 provides detailed installation information for MQSeries Integrator for Sun Solaris Version 2.0.1.

“Chapter 4. Getting Started with MQSeries Integrator” on page 25 explains how you complete product setup, using a simple installation to illustrate the tasks you need to complete. It also helps you to deploy your broker network and verify its operation using supplied verification programs.

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.

A glossary is also provided.

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.

What you need to know to understand this book

To understand this book, you need to be familiar with the system facilities of Sun Solaris and Windows NT. You also need to be familiar with the administration facilities of MQSeries for Sun Solaris V5.1 and MQSeries for Windows NT V5.1.

| You must also be familiar with the database product you are going to use to support the
| MQSeries Integrator for Sun Solaris Version 2.0.1 components.

The *MQSeries for Sun Solaris 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.

Terms used in this book

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

All new terms introduced in this book are defined in "Glossary of terms and abbreviations" on page 99. These terms are shown like *this* at their first use.

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 xiv).
- 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 Sun Solaris Version 2.0.1 Installation Guide, GC34-5842 (this book)
- IBM MQSeries Integrator for Windows NT Version 2.0.1 Installation Guide, GC34-5600
- 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 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-solaris-documentation` subdirectory under the root directory on the supplementary CD-ROM without installing them.
- On Windows NT, you can access the library after installation by selecting *Start->Programs->IBM MQSeries Integrator 2.0.1->Documentation*.

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 xiv.

- 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

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 Sun Solaris installation tasks you might need:

- IBM MQSeries for Sun Solaris V5.1 Quick Beginnings, GC33-1870.

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

  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.

MQSeries family publications

- 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”).

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 the following publication:

- 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.

- Down-load MQSeries SupportPacs.

MQSeries family publications

Summary of changes

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

Changes for this edition (GC34-5842-01)

The changes are summarized below:

- New migration information. See “Migration considerations” on page 11.
- Information about setting specific environment variables. See “Environment variables” on page 60.
- Details on defining the ODBC connections for Sun Solaris included. See “Creating and connecting to the databases” on page 31.
- Broker database support using Oracle 8. See “Appendix B. Setting up an Oracle8 MQSeries Integrator broker database” on page 63.
- For information about the components you must run on Windows NT, refer to the *MQSeries Integrator for Windows NT Version 2.0.1 Installation Guide*. This contains the details on planning and installation of MQSeries Integrator for Windows NT Version 2.0.1 that were previously in this book.
- See “Chapter 4. Getting Started with MQSeries Integrator” on page 25 for information about configuration and verification of all MQSeries Integrator components (on Sun Solaris and Windows NT).

Chapter 1. Installation overview

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

A broker domain consists of:

- One or more message brokers on Sun Solaris 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.
- Tthe two components that provide configuration and management support are the Configuration Manager and the Control Center on Windows NT.
  - 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 Sun Solaris. You can install and configure one or more message brokers on one or more Sun Solaris systems, subject to your licence agreement.

You can install and configure one User Name Server on Sun Solaris.

“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 15 tells you how to install the runtime support. You are recommended to install the runtime support before you install the configuration support on Windows NT, using the information presented in “Chapter 2. Planning for installation” on page 3, and “Chapter 3. Installing MQSeries Integrator” on page 15.

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 Sun Solaris, and have installed the Configuration Manager and Control Center on Windows NT, you can verify your installation.

"Chapter 4. Getting Started with MQSeries Integrator" on page 25 gives detailed step-by-step instructions for configuring your broker domain to verify your installation. It also 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

Successful installation of MQSeries Integrator involves three steps:

1. Planning and preparation:

   Careful planning of your installation will help you clarify your requirements and the actions needed to achieve the environment you want. All aspects of planning are covered in full in this chapter, and include the following:
   a. “System setup” on page 4.
   b. “Product components” on page 8.
   c. “Security considerations” on page 10.
   d. “Migration considerations” on page 11.
2. Installation:

   When you have decided which components you want to install, follow the guidance in “Chapter 3. Installing MQSeries Integrator” on page 15. The installation program checks for the prerequisite products, if any, required by the components you choose.
3. Configuration:

   When you have completed installation, you need to perform some initial configuration. These additional tasks will allow you to define and activate the resources in your installation. These steps are covered in “Chapter 4. Getting Started with MQSeries Integrator” on page 25.

   A simple configuration is used to illustrate the tasks needed and the results of the steps taken. A set of simple tests that verify that the installation has worked is also described.

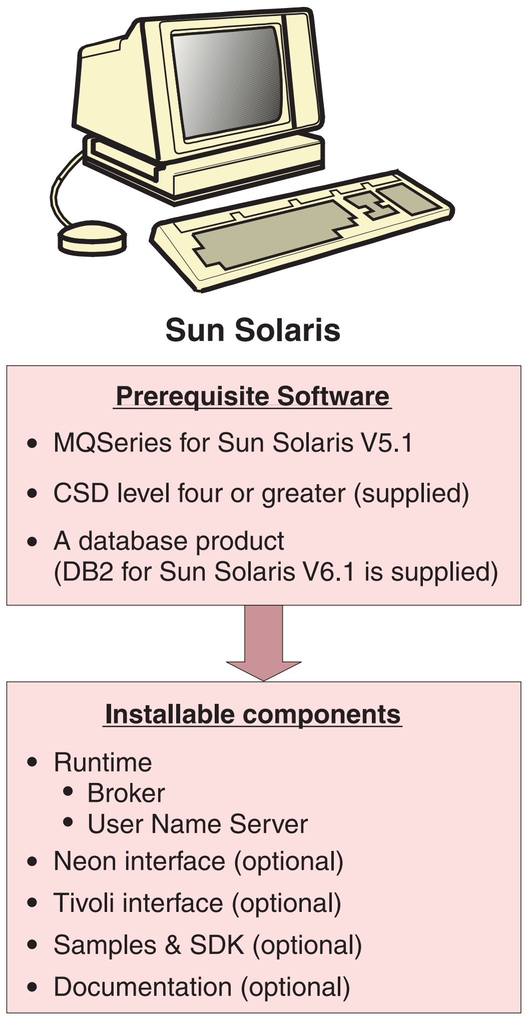

Sun Solaris
Prerequisite Software
• MQSeries for Sun Solaris V5.1
• CSD level four or greater (supplied)
• A database product (DB2 for Sun Solaris V6.1 is supplied)
Installable components
• Runtime
• Broker
• User Name Server
• Neon interface (optional)
• Tivoli interface (optional)
• Samples & SDK (optional)
• Documentation (optional)

Figure 1. Planning for installation

System setup

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

Hardware requirements

The hardware requirements for MQSeries Integrator for Sun Solaris are listed in the following sections.

General requirements

The general hardware requirements for MQSeries Integrator for Sun Solaris are:

- MQSeries servers:
  - Any Sun SPARC desktop or server system running version 2.7 of the operating system
  - Any Sun UltraSPARC desktop or server system
- Any communications hardware supporting NetBIOS, SNA LU 6.2, SPX, or TCP/IP.
- A minimum of 512 MB of RAM to support run-time operation of components.

Disk space required

The installation requirements depend on which 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.

Table 1 gives the component storage requirements in megabytes (MB).

Table 1. Component disk space requirements on Sun Solaris

Component	**MB**
Runtime (Broker and User Name Server)	150
Online documentation	15
Samples and SDK	1
NEON Interface	50
Tivoli Interface	3.5
TOTAL	250

Software requirements

Minimum supported levels are shown. Later compatible levels, if any, are supported unless otherwise stated.

Prerequisite software

The following products are prerequisites:

- Sun Solaris Version 2.7
- IBM MQSeries for Sun Solaris server Version 5.1.

  This must be at service level Corrective Service Diskette (CSD) 4.[1] The product and CSD are supplied in the MQSeries Integrator package. To find the current service level, use the following command:

  ```
  pkginfo | grep mqm
  ```

  You should look at `readme.txt` on the product CD-ROM to see the latest levels of software required.

1. If you are planning to establish an integrated broker network that includes both MQSeries Integrator and MQSeries Publish/Subscribe, see also “MQSeries Publish/Subscribe” on page 14.

The installation program checks that you have MQSeries for Sun Solaris Version 5.1 installed, and that it is at the correct service level.

If you do not have the MQSeries components you need, you are recommended to install these before you continue with MQSeries Integrator installation. You should check to see that these components are at the appropriate service level.

Note: Version 5.0 is not supported at any service level.

- A database product to support your broker or brokers, this can be one of the following:
  - IBM DB2 Universal Database for Sun Solaris Version 6.1 (Enterprise Edition, Connect Enterprise Edition, or Extended Enterprise Edition)
  - Oracle 8.1.5
  - Sybase 11.5 or 12

  If you already have one of the above installed, you can use it to support MQSeries Integrator.

  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.

  This database can be used with the broker component.

  If the installation program detects that you have a level of database prior to that indicated here, it highlights the need to upgrade your existing license and lists the supported levels. You can continue with installation, but you must upgrade your database before you can use MQSeries Integrator.

  DB2 requires FixPack 2 U469454 (this can be found on the MQSeries Integrator for Sun Solaris supplemental CD in the `supplement_sol/db2_for_solaris_v610_fixpack2` directory). DB2 also requires an additional 250MB of disk storage.

  Note: The supplied DB2 product has **restricted license terms and agreements**. You must only use this DB2 installation in association 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.

Optional products

The following products are options, not prerequisites.

- Connectivity

  The network protocols supported are SNA LU 6.2 and TCP.

  For SNA connectivity:

  - SunLink SNA Peer-to-Peer Version 9.1
  - If token ring is used: SunLink Token Ring Interface/ SBus 3.0.2 (requires patch 102463)
  - Sun TRI/P Adaptor V1.0
- Databases

  For a summary of the supported databases see "Database summary".

  The NEONRules and NEONFormatter subcomponents of the broker support message definitions created and maintained in a number of databases.

  Note: These databases are for message definitions created through the NEONFormatter only. The databases required for internal product use are listed earlier in this section.
- Application programming support

  The following software compilers are supported:
  - Sun WorkShop Compiler C Version 5.0
  - Sun WorkShop Compiler C++ Version 5.0
  - Micro Focus COBOL compiler Version 4.0 for UNIX

Database summary

The following is a list of supported databases:

- IBM DB2 Version 6.1 plus fixpack 2

  DB2 6.1 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.

  You must check the `readme.txt` file for your product to see if a Fixpack is required.
- Oracle 7.3.4 or 8.1.5

  Oracle 7.3.4 is not supported for use as a broker internal database.
- Sybase 11.5 or 12

  Sybase 12 is not supported by NEONRules and NEONFormats.

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 Sun Solaris 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 Sun Solaris 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 Sun Solaris has one primary component and four secondary components. A set of common files is installed with the primary component.

Runtime

When you install the Runtime component, the following two sub-components are always installed:

- Broker
- User Name Server

The broker

After installation, you can create one or more brokers on each system on which you have installed the Runtime component, subject to your license agreement (see “License information” on page 7 for details). You can configure and activate any number of brokers on these systems, subject to system resource constraints.

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.

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.

The database can be:
- IBM DB2 Version 6.1
- Oracle Version 8.1.5
- Sybase 11.5 or 12

“Database summary” on page 7 provides a summary of supported databases.

The User Name Server

The User Name Server requires an MQSeries queue manager to be assigned to it. The User Name Server does not require access to a database. You are recommended to configure one User Name Server within your broker domain.

Secondary components

There are four optional components that you can install in your broker domain if you choose. These are:
- Samples and Software Developers' Kit (SDK)
- Online documentation.
- NEON Interface
- Tivoli Interface

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 advantage of the full range of MQSeries Integrator function.

  – Sample programs

    Sample programs are supplied in C and Java. These programs are 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 49.

      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 in the `mqsi-docs` package, this is 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, which provides a cross book index and search facility, is also provided. You can access the documentation without installing: the publications are in the `mqsi_solaris_documentation` subdirectory in the root directory of the supplementary CD.

To read the documentation on Sun Solaris, install the Online documetation, start Acrobat reader and change to the /panels subdirectory then open the file bipabsol.pdf.

You can download a free copy of Acrobat Reader (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 xii.

NEON Interface

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

Tivoli Interface

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 group **mqbrkrs** on the system on which it is installed.

| The following table 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 (MQSeries non-trusted appl) (service user ID)	Member of **mqbrkrs**
Running broker (MQSeries trusted appl) (service user ID)	• Service user ID must be **mqm** • **mqm** must be a member of **mqbrkrs**
Clearing, joining, listing MQSeries publish/subscribe brokers	Member of **mqbrkrs**
Running publish/subscribe applications	Any user, subject to MQSeries Integrator topic and MQSeries queue access control

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

You can find more comprehensive information in the *MQSeries Integrator Administration Guide*.

| Migration considerations

| This section provides planning information for installation if you intend to migrate from,
| or coexist with, one of the following related products:
| • MQSeries Integrator Version 1
| • MQSeries Publish/Subscribe

| It also describes the action you must take prior to installation of MQSeries Integrator.

| For information about upgrading an existing Configuration Manager and Control Center
| on Windows NT, see the *MQSeries Integrator for Windows NT Version 2.0.1 Installation*
| *Guide*. This book can be found, in softcopy, on the MQSeries Integrator for Windows
| NT Version 2.0.1 CD.

MQSeries Integrator Version 1

If you have MQSeries Integrator Version 1.1 installed on your machine, and wish to continue using existing NEONRules and NEONFormats, you must:

1. Back up your existing NEONRules and NEONFormats
2. Uninstall your version of MQSeries Integrator Version 1.1
3. Install the `mqsi110` package, found on the main product CD

- Rules and formats
  1. If you are migrating from Version 1.1, backup existing Version 1.1 rules and formats using standard database backup facilities. This data can be used unchanged with Version 2.
  2. If you are migrating from MQSeries Integrator Version 1.0 to Version 2.0.1, your formats and rules are not compatible in their current form. If you wish to reuse your existing Version 1.0 rules and formats, you must migrate these to Version 1.1 format. Before you can migrate these formats, you must configure your databases. For details of the actions you need to take to complete this migration see “Appendix E. Using NEONFormatter and NEONRules nodes” on page 71. This step **must** be completed before you uninstall Version 1.0.
- Adding new rules and fomats

  Because the rules and formats created using the MQSeries Integrator Version 1.1 NEONRules and NEONFormatter user interface tools are fully compatible with MQSeries Integrator Version 2.0.1 components, you can continue to use these programs to add new rules and formats.

  If you prefer to use the MQSeries Integrator Version 2.0.1 facilities for defining rules and formats, you must use the Control Center. For a full description of these and other tasks supported by the Control Center, see the *MQSeries Integrator Using the Control Center* book.
- Access to rules and formats

  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 the 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 the 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.
- User exits

If you have set up user exits for the NEONFormatter, you must do the following if you want to reuse them with the NEONFormatter message processing node in MQSeries Integrator Version 2.0.1:

– If you are currently using user exits with MQSeries Integrator Version 1.1, your exits will work unchanged, but you must recompile and link them using the C++ Version 5 compiler.

– If you are currently using user exits with MQSeries Integrator Version 1.0, refer to the *MQSeries Integrator Version 1.1 Programming Reference for NEONFormatter* for instructions on how to use your user exits with MQSeries Integrator Version 1.1. They will then work with MQSeries Integrator Version 2.0.1.

- Logs and log records

  Log records generated by the NEONRules and NEONFormatter message processing nodes are written to the log file defined for the message flow in which the nodes appear.

  For details of how to specify a log file for a message flow, see the *MQSeries Integrator Using the Control Center* book.

  For details of the content of log records generated by the NEONRules and NEONFormatter nodes, see the *MQSeries Integrator Version 1.1 System Management Guide*.

- Backup files

  The configuration files established for Version 1.0 and Version 1.1 can both be reused unchanged with Version 2.0.1.

  Back up the following files:
  1. sqlsvses.cfg
  2. MQSIruleng.mpf
  3. MQSIputdata.mpf
  4. MQSIgetdata.mpf

  If you have user exit programs, you are recommended to back up these files too.

- Uninstall Version 1

  If you choose to uninstall Version 1.1, you must select the option that preserves your existing database entries. This ensures that all current rules and formats are still available for use with Version 2. Uninstallation of Version 1 is optional.

MQSeries Integrator Version 2.0.1 currently supports the following databases:
- IBM DB2 Universal Database Version 5.0 or later
- Microsoft SQL Server
- Oracle Version 7.3.4 or Version 8.1.5
- Sybase 11

NEONFormatter and NEONRules data continue to be supported by the same set of databases as Version 1.1. You are required to indicate the database you are using with your Version 1.1 setup during the installation process.

For further guidance of migration after installation, see “Appendix E. Using NEONFormatter and NEONRules nodes” on page 71, the *MQSeries Integrator Introduction and Planning* book.

MQSeries Publish/Subscribe

If you install MQSeries Integrator Version 2.0.1 with the intention of operating both MQSeries Integrator brokers and MQSeries Publish/Subscribe brokers connected in a heterogeneous broker network, you must ensure that you have the required level of MQSeries for your MQSeries Publish/Subscribe systems.

If you do not upgrade MQSeries to the specified levels on the systems on which you intend to run MQSeries Publish/Subscribe brokers, it is possible that some publications sent by MQSeries Integrator brokers will be wrongly put to the dead-letter queue (DLQ) by an MQSeries Publish/Subscribe neighbor broker.

You are not required to take any other specific action at installation time. Your current MQSeries Publish/Subscribe installation will not be affected by MQSeries Integrator Version 2.0.1 installation. However, you cannot create an MQSeries Integrator broker that uses a queue manager already in use by a MQSeries Publish/Subscribe broker.

For guidance on planning subsequent migration or integration of brokers, see *MQSeries Integrator Introduction and Planning*. 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 Sun Solaris.

It covers the following:
- "Delivery media".
- "Preparing for installation" on page 16.
- "Installation procedure" on page 19.
- "Setting up for LAN installation" on page 22.
- "What to do if something goes wrong during installation" on page 24.

Delivery media

The MQSeries Integrator for Sun Solaris Version 2.0.1 package includes the following:

- MQSeries Integrator for Sun Solaris Version 2.0.1.

  This CD includes the following:
  - MQSeries Integrator for Sun Solaris Version 2.0.1
  - MQSeries Integrator documentation for Sun Solaris Version 2.0.1
  - DB2 for Sun Solaris Version 6.1 (English only version)
  - NEON Rules and Formats runtime support

  Note to users

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

- MQSeries Integrator for Sun Solaris Version 2.0.1 supplemental CD.

  This CD includes the following:
  - MQSeries for Sun Solaris Version 5.1 CSD4.
  - MQSeries for Windows NT Version 5.1 CSD2.

    The CSDs are provided to enable you to upgrade existing installations of MQSeries for Sun Solaris or Windows NT Version 5.1.
  - IBM DB2 FixPack 2 U469454. This can be found in the following directory:

    `supplement_sol/db2_for_solaris_v610_fixpack2`
  - Additional product service, if required.
- MQSeries for Sun Solaris Version 5.1.

  If you are installing MQSeries for Sun Solaris Version 5.1, you must also install MQSeries for Sun Solaris Version 5.1 CSD4 (mqm-upd02). This can be found on the MQSeries Integrator for Sun Solaris supplemental CD. This product is provided in all available national languages.
- MQSeries Clients CD.

  Clients for all platforms in all available national languages are included.

Note: 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.
- MQSeries for Windows NT Version 5.1.

  If you are installing MQSeries for Windows NT Version 5.1, you must also install MQSeries for Windows NT Version 5.1 CSD4. This can be found on the MQSeries Integrator for Sun Solaris 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 5.

Preparing for installation

This section informs you of the steps you must take before you install and use MQSeries Integrator for Sun Solaris Version 2.0.1.

Before you start

Before starting to install MQSeries Integrator for Sun Solaris you:

- Must review `readme.txt`, which you can find in the root directory of the CD
- Must create a user group with the name `mqbrkrs`

  To do this, type the following command:

  ```
  groupadd mqbrkrs
  ```
- Must add **root** to the `mqbrkrs` group

  To do this, edit the `etc/group` file. Locate the line defining the `mqbrkrs` group and type `root` after the last colon. For example:

  ```
  mqbrkrs::42428:root
  ```
- Must create a user ID and add to `mqbrkrs` and `mqm`

  To do this, type the following command:

  ```
  useradd -G mqbrkrs,mqm -c "MQSeries Integrator user" <mqsi>
  ```

  Where `<mqsi>` is the user ID.

  Note that the `useradd` command creates a locked account by default. The account should now be unlocked to enable it to be used. To do this, type the following:

  ```
  Password <mqsi>
  ```

  You will be prompted to supply a new password for the user.

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.

- Must have the appropriate authority to access your MQSeries and database resources, see Table 2 on page 11 for more information.
- Must review the machine's Kernel configuration. See "Kernel Configuration" for the recommended settings.
- Are recommended to create and/or mount a `/var/mqsi` file system.

| After installation, the user ID `mqm`, in group `mqbrkrs`, owns all directories in `/var/mqsi`. All
| files and executables beneath `/opt/mqsi` are owned by user id `bin`, in group `bin`.

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 can also create another file system for product code. If, for example, you do not want to have the product code installed in the **`/opt/mqsi`** file system because it is too small (a minimum of 160MB is required), you can do one of two things:

1. Create a new file system and mount it as **`/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 **`/opt/mqsi`** to this new directory. For example:
   ```
   mkdir /bigdisk/mqsi
   ln -s /bigdisk/mqsi /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 common way of installing this product is direct from the CD. You can also set up for installation from a shared LAN drive. This is described in "Setting up for LAN installation" on page 22.

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

Kernel Configuration

| You are recommended to compare the values set in your Kernel configuration
| perameters to those listed in the following tables.

| To get the appropriate recommended values for your software configuration,

| 1. Check the recommended values for the following products:

- MQSeries Integrator
- MQSeries Version 5.1
- DB2 (if installed)
- Any other software that you are running, that provides a recommended value

2. Take the highest value for each parameter and compare each one to the corresponding value set in your Kernel configuration
3. If the highest recommended value is less than the current Kernel setting, you do not need to update this parameter
4. If the highest recommended value is greater than the current Kernel setting, replace the current value with the new higher value. To change a value edit the appropriate `set parameter = value` line in the /etc/system file. For further information on setting up the system, see the Sun Solaris System Administration documentation and documentation for other installed software products.

The following table shows the recommended Kernel parameter values on a Solaris 2.7 system.

```
set lwp_default_stksize = 0x4000
set rpcmod:svc_run_stksize = 0x4000
set shmsys:shminfo_shmmax = 4194304
set shmsys:shminfo_shmseg = 1024
set shmsys:shminfo_shmmni = 1024
set semsys:seminfo_semaem = 16384
set semsys:seminfo_semmni = 1024 (semmni < semmns)
set semsys:seminfo_semmap = 1026 (semmni +2)
set semsys:seminfo_semmns = 16384
set semsys:seminfo_semmsl = 125
set semsys:seminfo_semopm = 100
set semsys:seminfo_semmnu = 2048
set semsys:seminfo_semume = 256
set msgsys:msginfo_msgmap = 1026
```

Figure 2. Kernel parameter values - (Solaris 2.7 system)

The following table shows the Kernel parameter values recommended by DB2 for Sun Solaris on a 64 - 128MB machine.

```
set msgsys:msginfo_msgmax = 65535
set msgsys:msginfo_msgmnb = 65535
set msgsys:msginfo_msgmni = 128
set msgsys:msginfo_msgssz = 16
set msgsys:msginfo_msgtql = 256
set msgsys:msginfo_msgseg = 8192
set shmsys:shminfo_shmmax = 67108864
```

Figure 3. Kernel parameter values - recommended by DB2 for Sun Solaris

In addition, you need to increase the maximum number of concurrent open file descriptors on your system. You are recommended to set a value greater than 256.

To do this, you can use the `ulimit` command in the shell in which you are starting the broker.

Prerequisites

MQSeries Integrator for Sun Solaris 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 5.

Installation procedure

This section describes the installation of MQSeries Integrator for Sun Solaris Version 2.0.1.

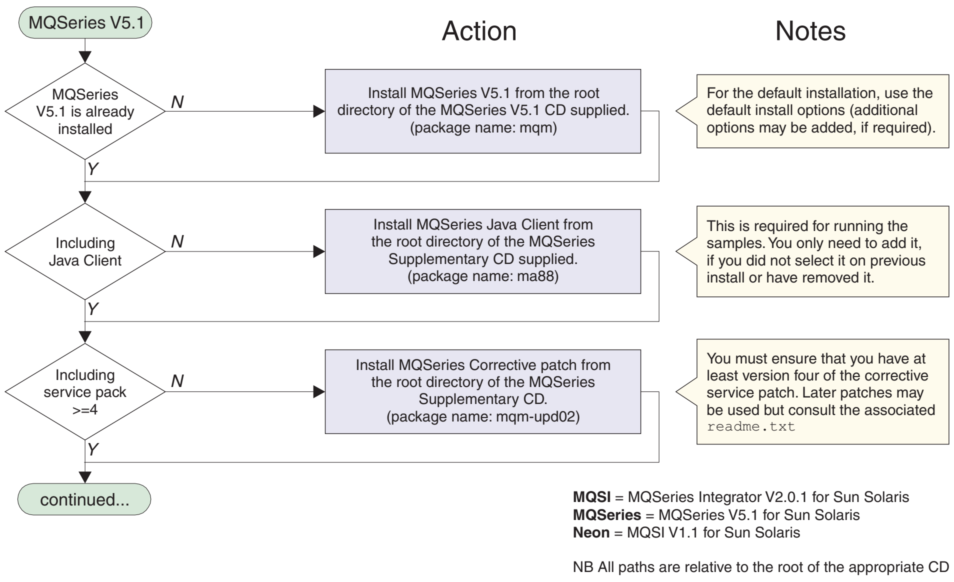

MQSeries V5.1
Action
Notes
MQSeries V5.1 is already installed
N
Y
Install MQSeries V5.1 from the root directory of the MQSeries V5.1 CD supplied. (package name: mqm)
For the default installation, use the default install options (additional options may be added, if required).
Including Java Client
N
Y
Install MQSeries Java Client from the root directory of the MQSeries Supplementary CD supplied. (package name: ma88)
This is required for running the samples. You only need to add it, if you did not select it on previous install or have removed it.
Including service pack >=4
N
Y
Install MQSeries Corrective patch from the root directory of the MQSeries Supplementary CD. (package name: mqm-upd02)
You must ensure that you have at least version four of the corrective service patch. Later patches may be used but consult the associated readme.txt
continued...
MQSI = MQSeries Integrator V2.0.1 for Sun Solaris
MQSeries = MQSeries V5.1 for Sun Solaris
Neon = MQSI V1.1 for Sun Solaris
NB All paths are relative to the root of the appropriate CD

Figure 4. Installation of MQSeries Integrator for Sun Solaris, part 1

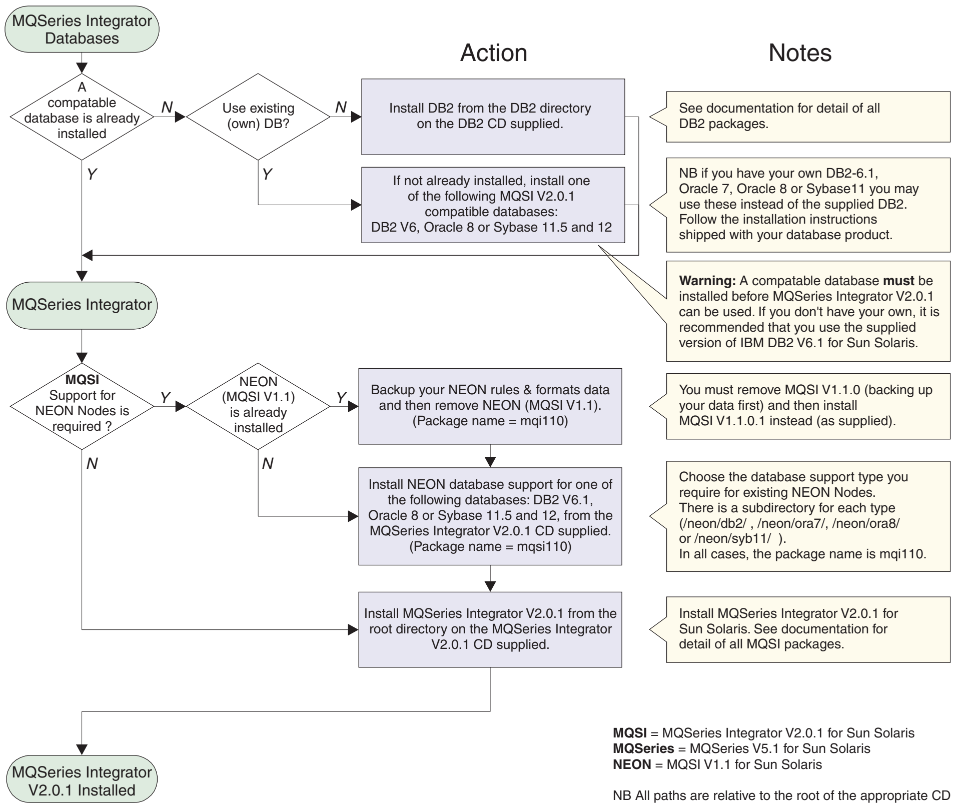

MQSeries Integrator Databases
Action
Notes
A compatable database is already installed
N
Use existing (own) DB?
N
Install DB2 from the DB2 directory on the DB2 CD supplied.
See documentation for detail of all DB2 packages.
Y
Y
If not already installed, install one of the following MQSI V2.0.1 compatible databases: DB2 V6, Oracle 8 or Sybase 11.5 and 12
NB if you have your own DB2-6.1, Oracle 7, Oracle 8 or Sybase11 you may use these instead of the supplied DB2. Follow the installation instructions shipped with your database product.
MQSeries Integrator
Warning: A compatable database must be installed before MQSeries Integrator V2.0.1 can be used. If you don't have your own, it is recommended that you use the supplied version of IBM DB2 V6.1 for Sun Solaris.
MQSI Support for NEON Nodes is required ?
Y
NEON (MQSI V1.1) is already installed
Y
Backup your NEON rules & formats data and then remove NEON (MQSI V1.1). (Package name = mqi110)
You must remove MQSI V1.1.0 (backing up your data first) and then install MQSI V1.1.0.1 instead (as supplied).
N
N
Install NEON database support for one of the following databases: DB2 V6.1, Oracle 8 or Sybase 11.5 and 12, from the MQSeries Integrator V2.0.1 CD supplied. (Package name = mqsi110)
Choose the database support type you require for existing NEON Nodes. There is a subdirectory for each type (/neon/db2/ , /neon/ora7/, /neon/ora8/ or /neon/syb11/). In all cases, the package name is mqi110.
Install MQSeries Integrator V2.0.1 from the root directory on the MQSeries Integrator V2.0.1 CD supplied.
Install MQSeries Integrator V2.0.1 for Sun Solaris. See documentation for detail of all MQSI packages.
MQSI = MQSeries Integrator V2.0.1 for Sun Solaris
MQSeries = MQSeries V5.1 for Sun Solaris
NEON = MQSI V1.1 for Sun Solaris
NB All paths are relative to the root of the appropriate CD
MQSeries Integrator V2.0.1 Installed

Figure 5. Installation of MQSeries Integrator for Sun Solaris, part 2

Installation of MQSeries Integrator for Sun Solaris

To install MQSeries Integrator for Sun Solarids, carry out the following procedure:

1. Check to see if the Volume Manager is running on your system by typing the following command:

   ```
   /usr/bin/ps -ef | /bin/grep vold
   ```

   If it is running, the CD is mounted on `/cdrom/mqsi_solaris` automatically. If it is not running, mount the CD by typing the following commands:

   ```
   mkdir -p /cdrom/mqsi_solaris
   mount -F hsfs -r /dev/dsk/cntndnsn /cdrom/mqsi_solaris
   ```

   Where `cntndnsn` is the name of your CD-ROM device.

   Alternatively, use the `volcheck` command to automount a CD-ROM device.
2. Use the Solaris **pkgadd** program, to install the software by carrying out the following procedure:
   a. To install, log in as root and execute:

      ```
      pkgadd -d <cdrom mount-point>
      ```

      Note: If you want to create a `logfile` which contains a transcript of everything that appears on the screen during installation, enter the following to the command:

      ```
      pkgadd -d /cdrom/mqsi_solaris/. 2>&1 | tee output_log_file
      ```
   b. Select one of the following packages:
      - mqsi MQSeries Integrator Version 2.0.1 for Sun Solaris (SPARC_SOL2.7) Version 2, Revision 1
      - mqsi-docs MQSeries Integrator 2.0.1 for Sun Solaris (SPARC_SOL2.7) Documentation Version 2, Revision 1
   c. Selecting the first package from the list above brings up the following:
      1) MQSI Runtime
      2) MQSI NEON Interface
      3) MQSI Tivoli Interface
      4) SDK & Samples

      Select the package you require.
   d. Press the Enter key.

During installation MQSeries Integrator for Sun Solaris checks that the following prerequisites have been installed:

- MQSeries for Sun Solaris Version 5.1
- MQSeries for Sun Solaris Version 5.1 CSD4

It also checks that the following have been created:

- `mqm` user
- `mqm` group

- `mqbrkrs` group

If any of the above are missing or not created the installation process warns you, and gives you the option to stop, but continues with the installation if you require.

Setting up for LAN installation

On Sun Solaris you can choose one of the following two methods to make the MQSeries Integrator installation files accessible on a LAN 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 folder on the LAN server to store the installation files. For example:
     ```
     mkdir /instmqsi
     ```
  2. Copy the entire CD to the new folder. For example:
     ```
     cp -rf /cdrom/mqsi_solaris/. /instmqsi
     ```
     This copies the complete contents of the CD to the specified location on the server.
  3. Give all licensed users access to the folder that now contains the CD-ROM image.
  4. Make the drive shareable:
     – Use the **share** command. For example, to give all users read-only access using NFS:
     ```
     share -F nfs -o ro -d "MQSeries Integrator Lan install" /instmqsi
     exportfs -a
     ```
  5. On the target machine, create a directory which will mount the shared drive. For example:
     ```
     mkdir /remotemqsiimage
     ```
  6. From a command prompt on the target machine:
     – Cconnect to the appropriate drive and folder using the **mount** command. For example:
     ```
     mount <machine name>:/instmqsi /remotemqsiimage
     ```
     Where <machine name> is the name of the target machine.
  7. Change to the installation directory:
     – In this example, `remotemqsiimage` type:
       a. `pkgadd -d <package_name>`.
          Where `<package_name>` is the name of the package to install. The two options are:
          - mqsi
          - mqsi-docs
       b. Press the Enter key

8. Follow the installation prompts.

Silent installation

MQSeries Integrator Version 2.0.1 supports silent installation, where no user interaction is expected. This method of installation is typically used where you have to install many identical copies of the same software across many machines. These machines can be remote to the installer.

Notes to users

1. This option should only be used by experienced administrators because no checks are performed on the system.
2. The output from the installation either appears on your terminal, or can be redirected to a file that you select.

The basic command is:

```
pkgadd -r RESPONSE -a ADMIN -d /cdrom/mqsi_solaris/. mqsi
```

where an example RESPONSE file is:

```
CLASSES=MQSI-Neon MQSI-Runtime MQSI-Tivoli SDK&Samples
```

and an example ADMIN file is:

```
mail=
instance=nocheck
partial=nocheck
runlevel=nocheck
idepend=nocheck
rdepend=nocheck
space=nocheck
setuid=nocheck
conflict=nocheck
action=nocheck
basedir=default
```

The RESPONSE file can be generated using the following command (from root):

```
pkgask -r RESPONSE -d /cdrom/mqsi_solaris/. mqsi
```

Or, for the documentation package:

```
pkgask -r RESPONSE -d /cdrom/mqsi_solaris/. mqsi-docs
```

The ADMIN file is the file /var/sadm/install/admin/default, altered so that no interaction or checking is done.

Note: Both the ADMIN and RESPONSE files must be in the current directory where the command is to be run.

Silent installation

An example script file is supplied on the product CD-ROM in the directory called `silent`. This script, “silent.sh”, enables administrators to install the product silently.

```
# Copyright IBM 2000
#
# This script installs MQSeries Integrator V2.0.1 with no
# checking of pre requisites, space etc and no user
# interaction.

pkgadd -r RESPONSE -a ADMIN -d /cd_rom/mqsi_solaris/. mqsi
```

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 xiv).
- 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.
- During installation a list of all the files installed is appended to `/var/sadm/install/contents`. Files that encountered problems during installation are shown with an exclamation mark (!) next to them.

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 you have, after checking the possible sources of error listed above, you must contact your IBM Support Center. See “Contacting your IBM Support Center” on page 67 for further information.

Chapter 4. Getting Started with MQSeries Integrator

This chapter helps you get started with MQSeries Integrator: it takes you through the set up of a very simple configuration, and shows how you can create a simple scenario to confirm that the product has been installed successfully.

For this configuration the Runtime component must be installed on your Sun Solaris machine, and the Configuration Manager and the Control Center components must be installed on your Windows NT machine. Refer to the *MQSeries Integrator for Windows NT Installation Guide* for information about Windows NT components.

The chapter has two sections:
- "Configuring a simple broker domain".
- "Verifying your installation" on page 46.

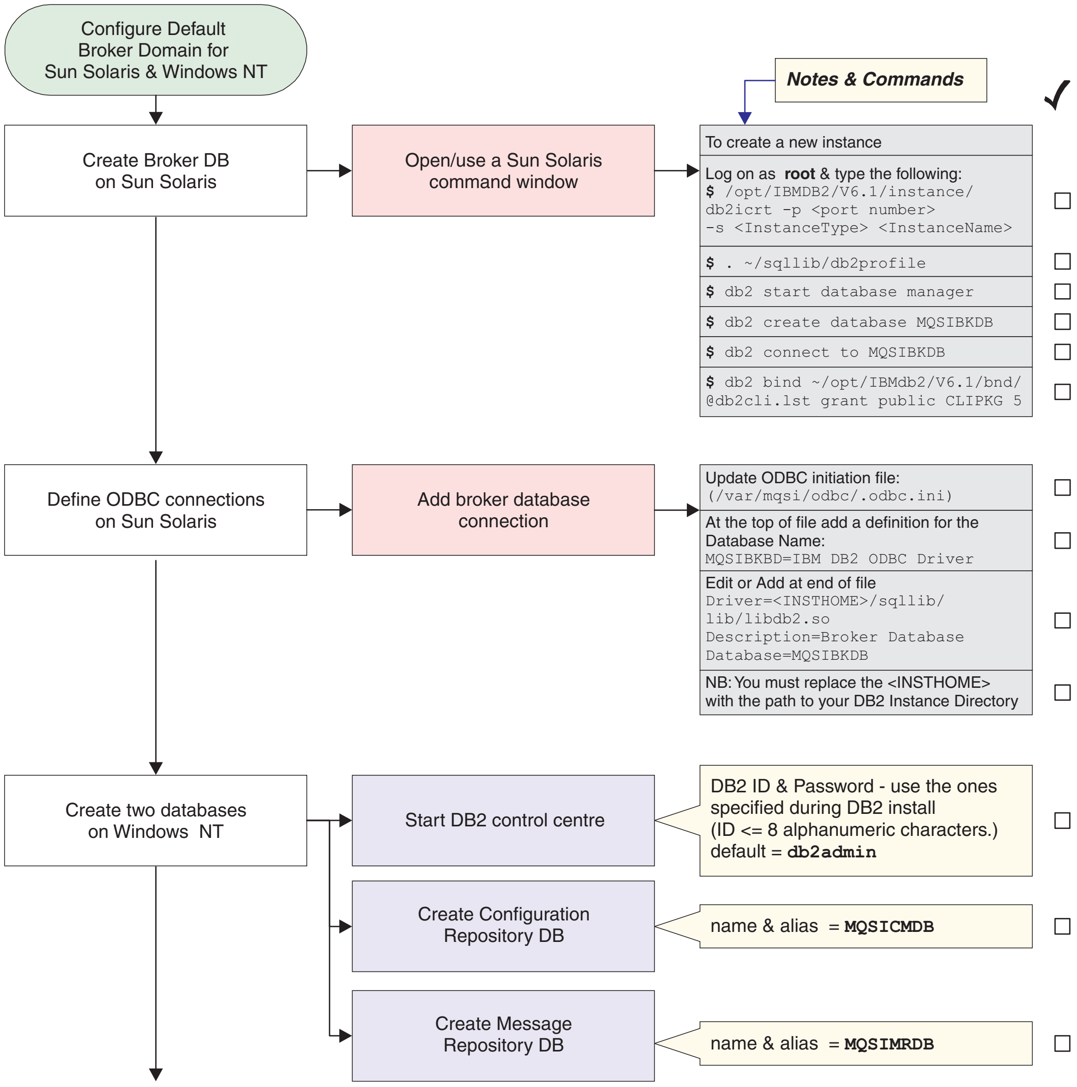
Configuring a simple broker domain

This section takes you through the steps you must complete, on each platform, to set up the minimum resources required in the broker domain.

There are five steps to take:
Step 1. "Designing a simple configuration" on page 29.
Step 2. "Creating and connecting to the databases" on page 31.
Step 3. "Setting up database authorizations" on page 35.
Step 4. "Configuring your broker domain" on page 37.
Step 5. "Starting your broker domain" on page 41.

Figure 6 on page 26 provides a schematic of these tasks: you might find it helpful to review this figure before you start the tasks detailed in this section, and to use it to check off the tasks as you complete them. You must complete the sub-tasks defined to complete each of these steps. The figure and the text that follows illustrate the tasks in a logical sequence and takes no account of the platform on which you must complete them. If you prefer, you can carry out all the tasks on Sun Solaris followed by all the tasks on Windows NT: the outcome is the same.

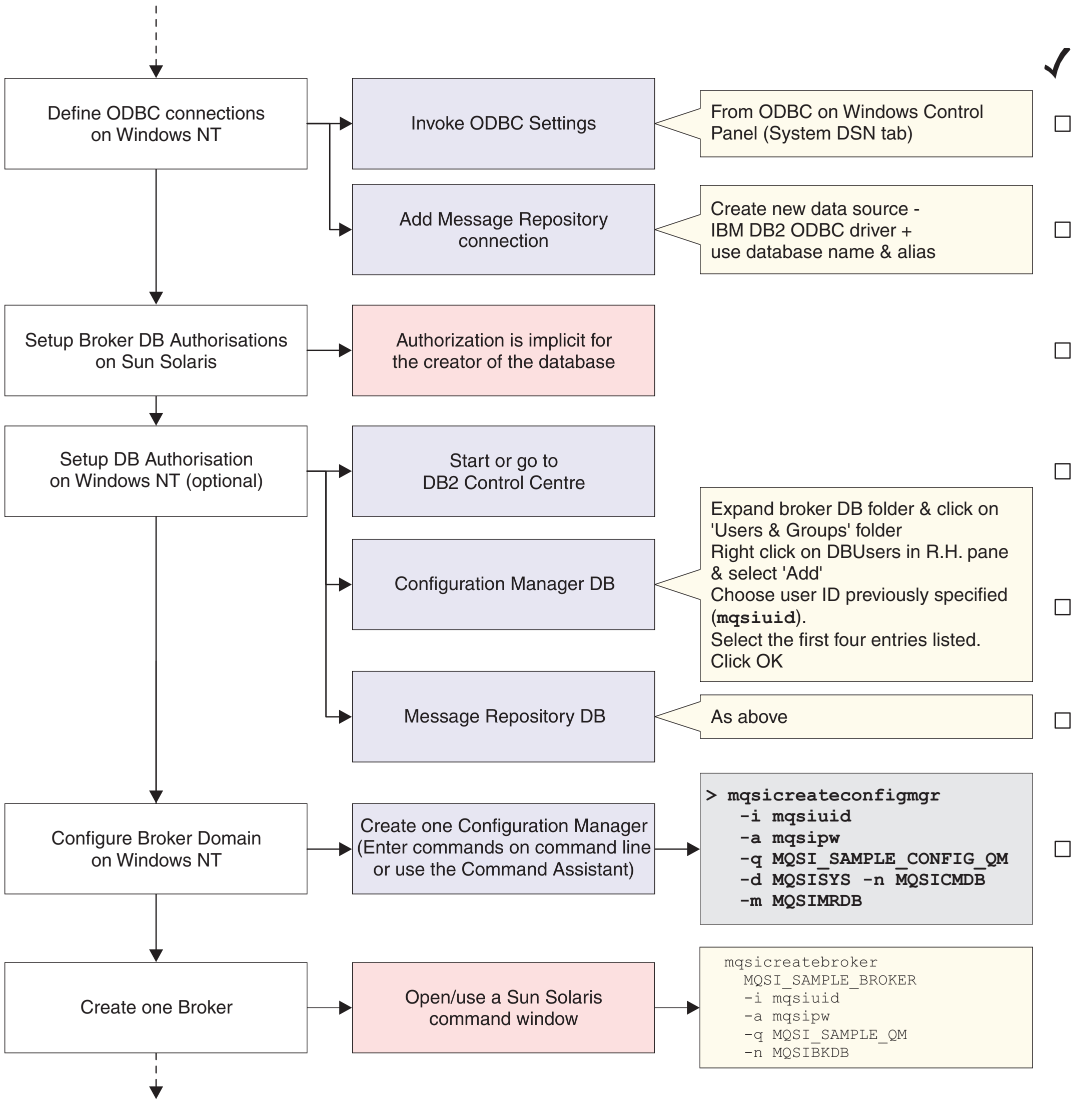
Configuring a broker domain



Configure Default Broker Domain for Sun Solaris & Windows NT
Create Broker DB on Sun Solaris
Open/use a Sun Solaris command window
Notes & Commands
✓
Define ODBC connections on Sun Solaris
Add broker database connection
Create two databases on Windows NT
Start DB2 control centre
Create Configuration Repository DB
Create Message Repository DB

Notes & Commands	✓
To create a new instance	
Log on as **root** & type the following: `$ /opt/IBMDB2/V6.1/instance/db2icrt -p <port number> -s <InstanceType> <InstanceName>`	☐
`$ . ~/sqllib/db2profile`	☐
`$ db2 start database manager`	☐
`$ db2 create database MQSIBKDB`	☐
`$ db2 connect to MQSIBKDB`	☐
`$ db2 bind ~/opt/IBMdb2/V6.1/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**	☐

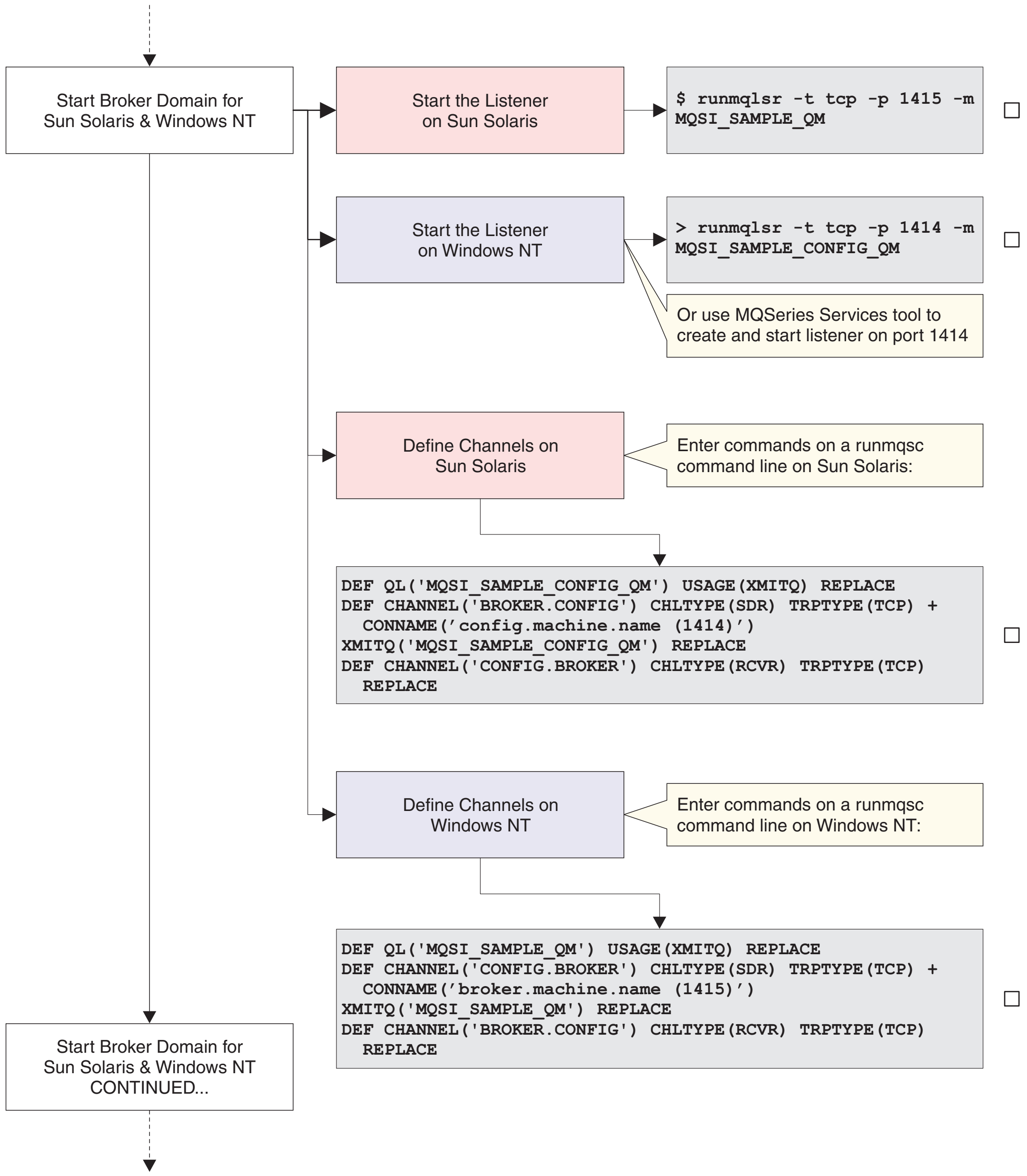
Figure 6. Configuring a simple broker domain


✓
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 Sun Solaris
Authorization is implicit for the creator of the database
Setup 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 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)
Create one Broker
Open/use a Sun Solaris command window

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

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

Figure 7. Configuring a simple broker domain continued


Start Broker Domain for Sun Solaris & Windows NT
Start the Listener on Sun Solaris
☐
Start the Listener on Windows NT
☐
Or use MQSeries Services tool to create and start listener on port 1414
Define Channels on Sun Solaris
Enter commands on a runmqsc command line on Sun Solaris:
☐
Define Channels on Windows NT
Enter commands on a runmqsc command line on Windows NT:
☐
Start Broker Domain for Sun Solaris & Windows NT CONTINUED...

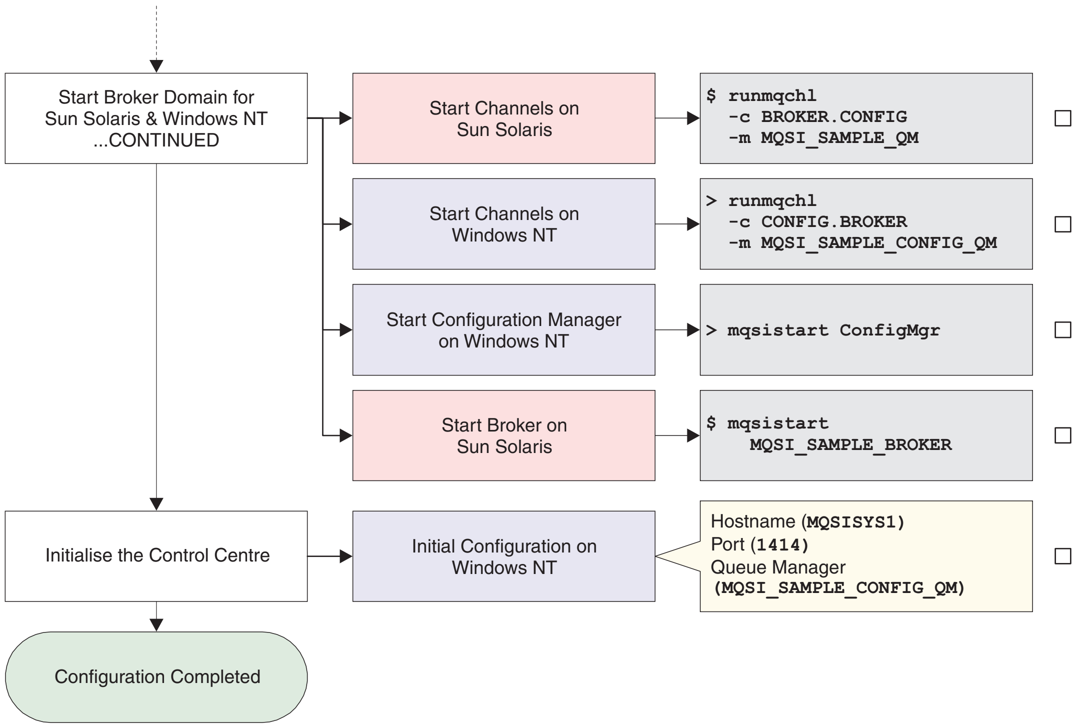
```
$ runmqlsr -t tcp -p 1415 -m
MQSI_SAMPLE_QM
```

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

```
DEF QL('MQSI_SAMPLE_CONFIG_QM') USAGE(XMITQ) REPLACE
DEF CHANNEL('BROKER.CONFIG') CHLTYPE(SDR) TRPTYPE(TCP) +
  CONNAME('config.machine.name (1414)')
XMITQ('MQSI_SAMPLE_CONFIG_QM') REPLACE
DEF CHANNEL('CONFIG.BROKER') CHLTYPE(RCVR) TRPTYPE(TCP)
  REPLACE
```

```
DEF QL('MQSI_SAMPLE_QM') USAGE(XMITQ) REPLACE
DEF CHANNEL('CONFIG.BROKER') CHLTYPE(SDR) TRPTYPE(TCP) +
  CONNAME('broker.machine.name (1415)')
XMITQ('MQSI_SAMPLE_QM') REPLACE
DEF CHANNEL('BROKER.CONFIG') CHLTYPE(RCVR) TRPTYPE(TCP)
  REPLACE
```

Figure 8. Configuring a simple broker domain continued


Start Broker Domain for Sun Solaris & Windows NT ...CONTINUED
Start Channels on Sun Solaris
Start Channels on Windows NT
Start Configuration Manager on Windows NT
Start Broker on Sun Solaris
Initialise the Control Centre
Initial Configuration on Windows NT
Hostname (MQSISYS1)
Port (1414)
Queue Manager (MQSI_SAMPLE_CONFIG_QM)
Configuration Completed

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

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

```
> mqsistart ConfigMgr
```

```
$ mqsistart
    MQSI_SAMPLE_BROKER
```

Figure 9. Configuring a simple broker domain continued

Most of these steps make use of commands supplied by MQSeries Integrator. Some of these commands can be invoked using the MQSeries Integrator Command Assistant; 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. You can choose which method you want to use to issue 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.

Designing a simple 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.

The assumptions made in this chapter are:

- You have installed the Runtime component on Sun Solaris.
- You have installed the Configuration Manager and the Control Center components on Windows NT.
- You have installed the product on systems with a TCP/IP hostnames of MQSISYS1 on Windows NT and MQSISYS2 on Sun Solaris. You **must** replace these names 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.
- 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 are logged on with the same user ID that you used to install MQSeries Integrator. If you are not, you must ensure that your current logon ID is a member of:
  - The `mqbrkrs` group on Sun Solaris
  - The **Administrator** group on Windows NT
- You have authorized the user IDs between Sun Solaris and Windows NT. You must ensure that:
  - The ServiceUserid that you will use on Windows NT (specified on the mqsicreateconfigmgr command) is defined on the machine where the Sun Solaris broker is running.
  - The ServiceUserid that you will use on Sun Solaris (specified on the mqsicreatebroker command) is defined on the machine where the Configuration Manager is running.

  In this example we use the service user ID `mqsiuid`. If you have chosen another service user ID, you must define this on both Sun Solaris and Windows NT. See the *MQSeries Integrator Administration Guide* for more information about setting up user IDs and groups.
- 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 once 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 Sun Solaris, 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 installation has been done as part of your installation procedure, and you have since restarted your system and verified your DB2 installation.

Once you have verified your installation, and understood and implemented the basic principles of operating your broker domain, you are very likely to 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*.

Creating and connecting to the databases

Note to users

On multi-way machines you need to 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:

On Sun Solaris

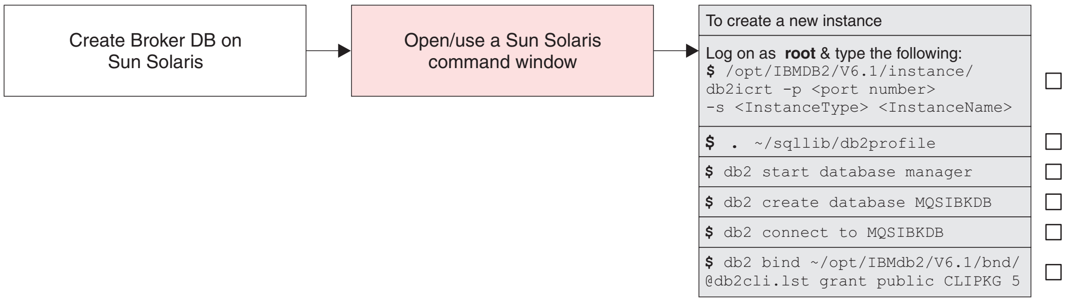

Create Broker DB on Sun Solaris
Open/use a Sun Solaris command window
To create a new instance
Log on as **root** & type the following:
$ /opt/IBMDB2/V6.1/instance/db2icrt -p <port number> -s <InstanceType> <InstanceName>
$. ~/sqllib/db2profile
$ db2 start database manager
$ db2 create database MQSIBKDB
$ db2 connect to MQSIBKDB
$ db2 bind ~/opt/IBMdb2/V6.1/bnd/@db2cli.lst grant public CLIPKG 5

Figure 10. Creating and connecting to the database on Sun Solaris

If you ran `db2setup`, a default instance will have 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.

1. Logon as root and type the following commands:

   ```
   /opt/IBMdb2/V6.1/instance/db2icrt
    -p <port number>
    -s <InstanceType> <InstanceName>
   ```

   Where `<port number>` is the number of the TCP port that the instance is to wait on for connections, `<InstanceType>` is one of `eee`,`ee`, or `client` (`ee` is the noraml MQSeries Integrator requirement), and `<InstanceName>` is the name of the new instance.

   Note: The user and the user's home directory should be local to the machine containing DB2 and not NIS or NFS mounted.

2. Logon as `<username>`, using the following command:

   ```
   . ˜/sqllib/db2profile
   db2 start database manager
   db2 create database MQSIBKDB
   db2 connect to MQSIBKDB
   db2 bind ˜/opt/IBMdb2/V6.1/bnd/@db2cli.lst grant public CLIPKG 5
   ```

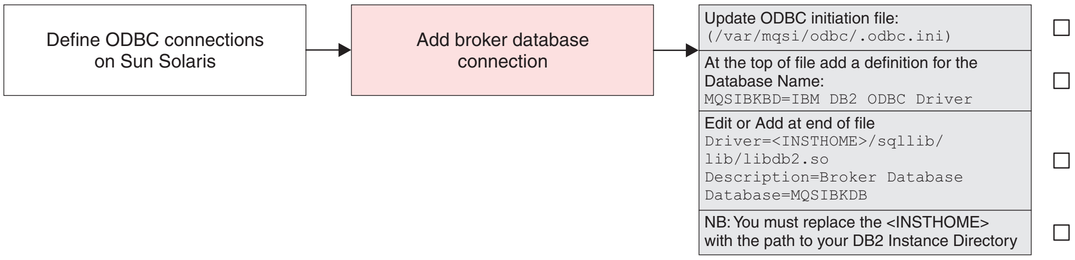

Define ODBC connections on Sun Solaris
Add broker database connection
Update ODBC initiation file: (/var/mqsi/odbc/.odbc.ini)
At the top of file add a definition for the Database Name: MQSIBKBD=IBM DB2 ODBC Driver
Edit or Add at end of file Driver=<INSTHOME>/sqllib/lib/libdb2.so Description=Broker Database Database=MQSIBKDB
NB: You must replace the <INSTHOME> with the path to your DB2 Instance Directory

Figure 11. Defining the ODBC connection on Sun Solaris

3. In order for MQSeries Integrator brokers to use the database you must update the ODBC initiation file (`/var/mqsi/odbc/.odbc.ini`) to contain definitions for the database.

   To do this edit the file and add the following lines.

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

     ```
     MQSIBKDB=IBM DB2 ODBC Driver
     ```

   - At the end of the file, edit or add the following lines:

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

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

     Note: Problems with ODBC can be traced by modifying the `/var/mqsi/odbc/.odbc.ini` file as follows:

     ```
     [ODBC]
     Trace=1
     TraceFile=/var/mqsi/odbc/odbctrace.out
     TraceDll=/opt/mqsi/merant/lib/odbctrac.so
     InstallDir=/opt/mqsi/merant
     ```

     The userid that MQSeries Integrator runs under must have write access to this file.

If you are using Oracle8 to support the broker, see “Appendix B. Setting up an Oracle8 MQSeries Integrator broker database” on page 63.

On Windows NT

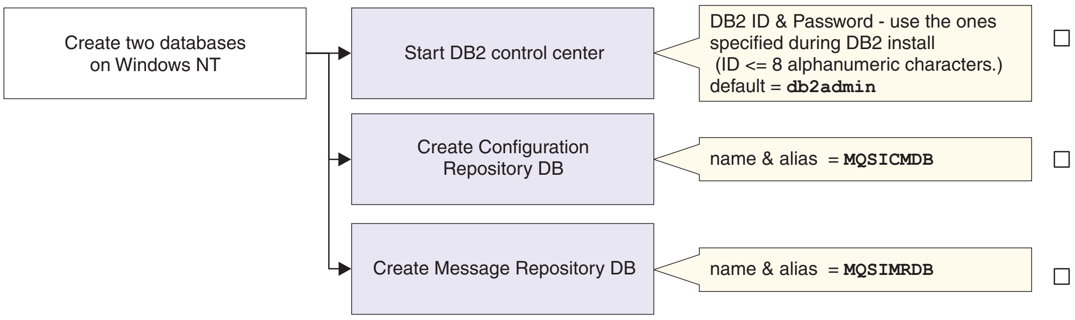

Create two databases on Windows NT
Start DB2 control center
DB2 ID & Password - use the ones specified during DB2 install (ID <= 8 alphanumeric characters.) default = db2admin
Create Configuration Repository DB
name & alias = MQSICMDB
Create Message Repository DB
name & alias = MQSIMRDB

Figure 12. Creating and connecting to the databases on Windows NT

Step 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 ID you specified when you installed DB2, the DB2 Administrator ID. 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.

Step 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.

For each database you create, you must enter a name and alias. You can let all remaining options take the default values.

a. 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.

b. 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 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.

Step 3. Define the ODBC connections.

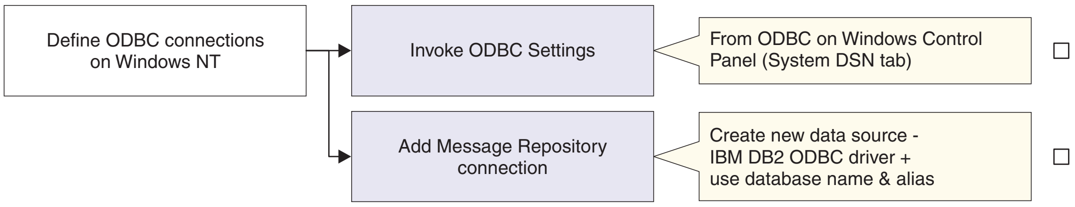

Define ODBC connections on Windows NT
Invoke ODBC Settings
From ODBC on Windows Control Panel (System DSN tab)
Add Message Repository connection
Create new data source - IBM DB2 ODBC driver + use database name & alias

Figure 13. Defining the ODBC connections on Windows NT

a. From the Windows NT Start menu, select *Start->Settings->Control Panel*.

b. Within the Control Panel, double-click the ODBC icon (this will be 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**.

Setting up database authorizations

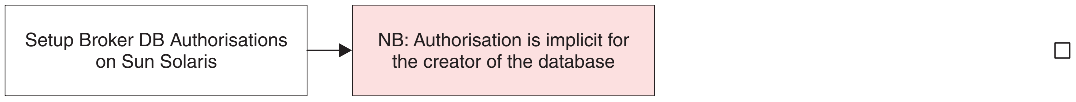

Setup Broker DB Authorisations on Sun Solaris
NB: Authorisation is implicit for the creator of the database

Figure 14. Setting up database authorizations on Sun Solaris

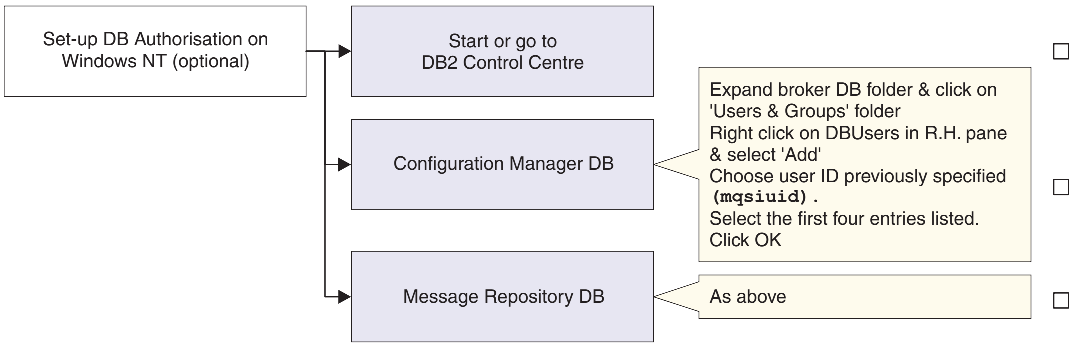

Set-up DB Authorisation on Windows NT (optional)
Start or go to DB2 Control Centre
Configuration Manager DB
Message Repository DB
Expand broker DB folder & click on 'Users & Groups' folder
Right click on DBUsers in R.H. pane & select 'Add'
Choose user ID previously specified (mqsiuid).
Select the first four entries listed.
Click OK
As above
☐
☐
☐

Figure 15. Setting up database authorizations on Windows NT

The next task is to authorize selected user IDs to access the database or databases you 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 Configuration Manager command. This option is not illustrated in this book. See the *MQSeries Integrator Administration Guide* for further information.

Step 1. Start the DB2 Control Center, if it is not already active. Log on with the DB2 administrator user ID you used in “Creating and connecting to the databases” on page 31.

Step 2. Complete the following tasks:

a. Expand the object tree until you find the database.

b. Expand the tree under this database and left-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 the database you 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.

Step 3. You can now close the DB2 Control Center.

Configuring your broker domain

Now you are ready to define the components that make up the simple configuration.

You must work through the following steps:

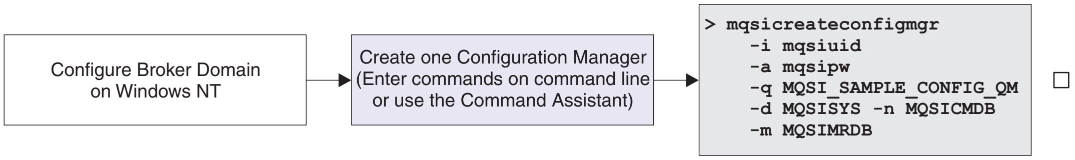
Step 1. Create one Configuration Manager on your Windows NT machine.
Step 2. Create one broker on your Sun Solaris machine.

On Windows NT, step 1 is illustrated using the Command Assistant. When you enter values in the entry fields, you will see a display of the command that is generated in the lower part of the screen.

The MQSeries Integrator commands are also provided for both platforms. On Sun Solaris, enter the commands at the command line. On Windows NT, enter the commands at a command prompt window. You are recommended to enter the commands from the `\bin` subdirectory of the directory in which you installed MQSeries Integrator for Windows NT (the home directory). If you accepted the default, the home directory is:

```
C:\Program Files\IBM MQSeries Integrator 2.0.1
```

Step 1: creating the Configuration Manager on Windows NT


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
```

Figure 16. Creating the Configuration Manager

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 screens 1 and 2. You can then review and check the full command, which will appear on screen 3. When you are happy that the command is correct, click **Finish**.

If you prefer, you can enter the command directly, for example:

```
mqsicreateconfigmgr -i mqsiuid -a mqsipw -q MQSI_SAMPLE_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_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.
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_CONFIG_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 62.
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.

Step 2: creating a broker on Sun Solaris

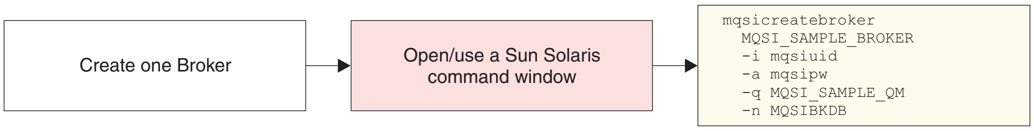


Figure 17. Creating a broker on Sun Solaris

To create a broker, enter the following command:

```
mqsicreatebroker MQSI_SAMPLE_BROKER -i mqsiuid -a mqsipw
-q MQSI_SAMPLE_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 these here and grant the user ID access to the database (described in "Setting up database authorizations" on page 35).

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_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. It is possible that the error has 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:

1. The broker MQSI_SAMPLE_BROKER has been created.
2. The queue manager MQSI_SAMPLE_QM has been created and started.
3. The MQSeries resources required by the broker have been defined These resources are detailed in "Default MQSeries resources" on page 62.
4. The required authorizations for the MQSeries resources have been set (the *setmqaut* messages seen in the responses to the command).

5. The database tables required by the broker have been set up in the database MQSIBKDB. These tables are listed in Table 5 on page 61.

Checking the components

You can check the existence of the MQSeries Integrator components you have created on each platform using the `mqsilist` command.

- On a Sun Solaris command line type:

  ```
  mqsilist
  ```

  This displays the broker component, with the queue manager which supports it. Now that the broker has been created, the command responds with:

  ```
  BIP8099I: MQSI_SAMPLE_BROKER - MQSI_SAMPLE_QM

  BIP8071I: Successful command completion.
  ```
- On a Windows NT command prompt type:

  ```
  mqsilist
  ```

  This displays the Configuration Manager component, with the queue manager which supports it. Now that the Configuration Manager has been created, the command responds with:

  ```
  BIP8099I: ConfigMgr - MQSI_SAMPLE_CONFIG_QM
  BIP8071I: Successful command completion.
  ```

Starting your broker domain

When you have created the MQSeries Integrator components, you can start to activate your broker domain. You must have Windows NT **Administrator** authority on your Windows NT machine to complete these steps.

You are recommended to complete the tasks described here in the following order:

Step 1. Start the MQSeries listener on each queue manager.
Step 2. Start the MQSeries sender channel on each queue manager.
Step 3. Start the Configuration Manager (Windows NT machine).
Step 4. Start your broker (Sun Solaris machine).
Step 5. Start the Control Center (Windows NT machine).

The MQSeries Integrator commands are used to illustrate these steps, with sufficient information provided to complete the task. For a full description of these commands, and possible errors, see the *MQSeries Integrator Administration Guide*. (You cannot use the MQSeries Integrator for Windows NT Command Assistant for these commands.) If errors are reported, you can also check the Application view of the Windows NT Event Viewer and the Sun Solaris syslog.

Step 1: starting the MQSeries listeners

Most of the resources you need to support this simple configuration have already been created and started for you when you invoked the create broker and create Configuration Manager commands. There is just one extra step you need to take to enable the Control Center to communicate with the Configuration Manager.

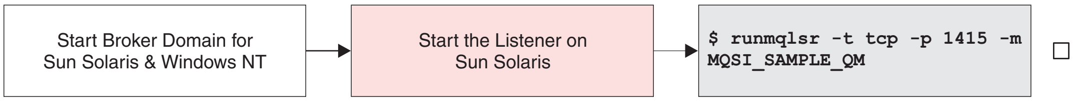

Start Broker Domain for Sun Solaris & Windows NT
Start the Listener on Sun Solaris
$ runmqlsr -t tcp -p 1415 -m MQSI_SAMPLE_QM

Figure 18. Starting the listener on Sun Solaris

- On a Sun Solaris command line type:

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

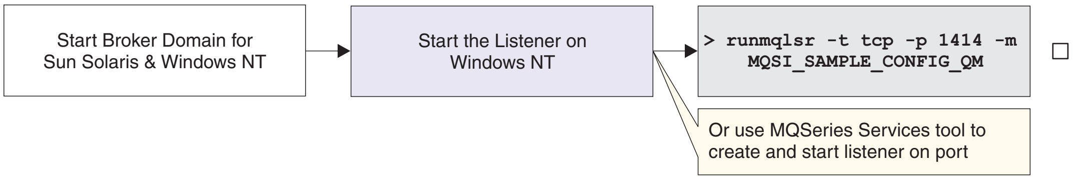

Start Broker Domain for Sun Solaris & Windows NT
Start the Listener on Windows NT
> runmqlsr -t tcp -p 1414 -m MQSI_SAMPLE_CONFIG_QM
Or use MQSeries Services tool to create and start listener on port

Figure 19. Starting the listener on Windows NT

- On Windows NT use 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:

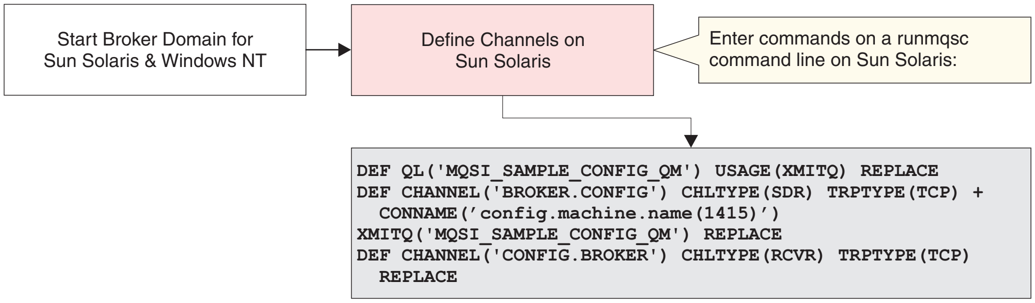
     ```
     runmqlsr -t tcp -p 1414 -m MQSI_SAMPLE_CONFIG_QM
     ```

     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. 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.

Step 2: defining the channels

To define the channels on each platform, use the following examples:

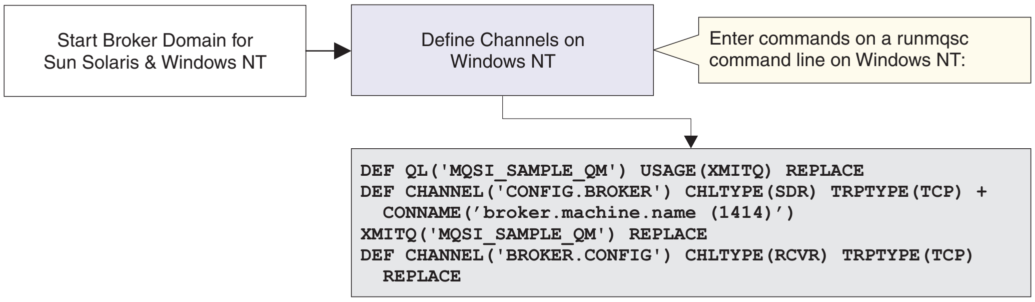

Start Broker Domain for Sun Solaris & Windows NT
Define Channels on Sun Solaris
Enter commands on a runmqsc command line on Sun Solaris:

```
DEF QL('MQSI_SAMPLE_CONFIG_QM') USAGE(XMITQ) REPLACE
DEF CHANNEL('BROKER.CONFIG') CHLTYPE(SDR) TRPTYPE(TCP) +
  CONNAME('config.machine.name(1415)')
XMITQ('MQSI_SAMPLE_CONFIG_QM') REPLACE
DEF CHANNEL('CONFIG.BROKER') CHLTYPE(RCVR) TRPTYPE(TCP)
  REPLACE
```

Figure 20. Defining the channels on Sun Solaris

The following example must be run on a runmqsc command line on the Solaris machine. The example sets up the required channels.

```
DEF QL('MQSI_SAMPLE_CONFIG_QM') USAGE(XMITQ) REPLACE
DEF CHANNEL('BROKER.CONFIG') CHLTYPE(SDR) TRPTYPE(TCP) +
CONNAME('config.machine.name (1415)')
XMITQ('MQSI_SAMPLE_CONFIG_QM') REPLACE
DEF CHANNEL('CONFIG.BROKER') CHLTYPE(RCVR) TRPTYPE(TCP) REPLACE
```


Start Broker Domain for Sun Solaris & Windows NT
Define Channels on Windows NT
Enter commands on a runmqsc command line on Windows NT:

```
DEF QL('MQSI_SAMPLE_QM') USAGE(XMITQ) REPLACE
DEF CHANNEL('CONFIG.BROKER') CHLTYPE(SDR) TRPTYPE(TCP) +
  CONNAME('broker.machine.name (1414)')
XMITQ('MQSI_SAMPLE_QM') REPLACE
DEF CHANNEL('BROKER.CONFIG') CHLTYPE(RCVR) TRPTYPE(TCP)
  REPLACE
```

Figure 21. Defining the channels on Windows NT

The following example must be run on a runmqsc command ine on the Windows NT machine. The example sets up the required channels.

```
DEF QL('MQSI_SAMPLE_QM') USAGE(XMITQ) REPLACE
DEF CHANNEL('CONFIG.BROKER') CHLTYPE(SDR) TRPTYPE(TCP)
CONNAME ('broker.machine.name (1414)')
XMITQ('MQSI_SAMPLE_QM') REPLACE
DEF CHANNEL('BROKER.CONFIG') CHLTYPE(RCVR) TRPTYPE(TCP) REPLACE
```

Step 3: starting the channels

Start each queue manager channel by typing the following commands:

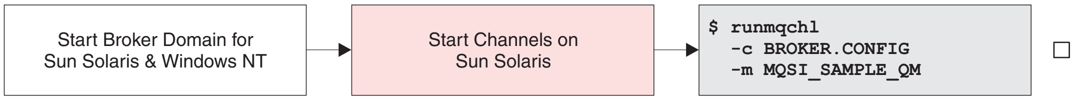

Start Broker Domain for Sun Solaris & Windows NT
Start Channels on Sun Solaris
$ runmqchl -c BROKER.CONFIG -m MQSI_SAMPLE_QM
☐

Figure 22. Starting the channels on Sun Solaris

- On Sun Solaris:

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

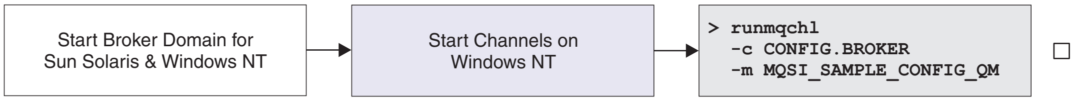

Start Broker Domain for Sun Solaris & Windows NT
Start Channels on Windows NT
> runmqchl -c CONFIG.BROKER -m MQSI_SAMPLE_CONFIG_QM
☐

Figure 23. Starting the channels on Windows NT

- On Windows NT:

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

Step 4: starting the Configuration Manager

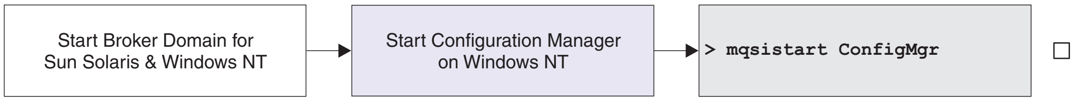

Start Broker Domain for Sun Solaris & Windows NT
Start Configuration Manager on Windows NT
> mqsistart ConfigMgr
☐

Figure 24. Starting the Configuration Manager

Start your Configuration Manager by entering the following command on the command line (you cannot do this using the Command Assistant):

```
mqsistart configmgr
```

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

The Configuration Manager cannot contact a broker until a reference to that broker exists in the configuration repository. You must create this reference to the broker using the Control Center. The steps required to do this are described in "Preparing for verification" on page 46.

Step 5: starting the broker

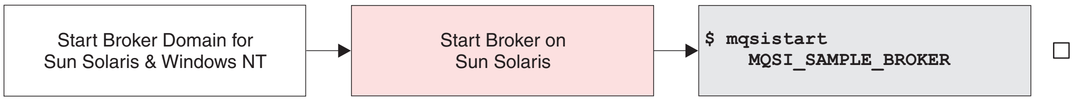

Start Broker Domain for Sun Solaris & Windows NT
Start Broker on Sun Solaris
$ mqsistart MQSI_SAMPLE_BROKER
☐

Figure 25. Starting the broker

Start your broker by entering the following command on the command line. If you are not using the sample broker name, substitute your broker name for MQSI_SAMPLE_BROKER in the command.

```
mqsistart MQSI_SAMPLE_BROKER
```

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

Step 6: starting the Control Center

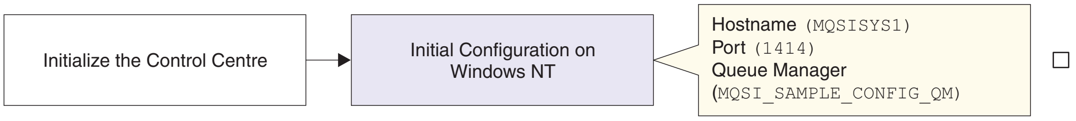

Initialize the Control Centre
Initial Configuration on Windows NT
Hostname (MQSISYS1)
Port (1414)
Queue Manager (MQSI_SAMPLE_CONFIG_QM)
☐

Figure 26. Initialize 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 “Verifying your installation” on page 46.

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 “Step 1: starting the MQSeries listeners” on page 41: (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 which groups your current user ID is a member of. 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.

Verifying your installation

You have now completed the configuration and activation tasks. This section 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 set up and operation:
- “Preparing for verification”.
- “Running the predefined verification applications” on page 49.
  - “Running the Results Service application” on page 50.
  - “Running the Scribble application” on page 51.
  - | “Running the Postcard application” on page 53.
- “Building and using a simple message flow” on page 55.

All the tasks illustrated here assume you have used the sample names and values when you completed the tasks in “Configuring a simple broker domain” on page 25. 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 will 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 Sun Solaris)

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 Results Service application
  - MQSI_SOCCER_PUBLICATION_QUEUE
  - MQSI_SOCCER_SUBSCRIPTION_QUEUE
- For the Scribble application
  - MQSI_SCRIBBLE_PUBLICATION_QUEUE
  - MQSI_SCRIBBLE_SUBSCRIPTION_QUEUE
- For the Postcard application
  - MQSI_POSTCARD_INPUT_QUEUE
  - MQSI_POSTCARD_OUTPUT_QUEUE

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

```
runmqsc MQSI_SAMPLE_QM < sample/mqsc
```

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.

Step 1. Stop the Control Center and the Configuration Manager (`mqsistop configmgr`).

Step 2. Import the message set required by the Postcard application into the message repository. The message set is defined in the file `PostcardMS.mrp`.

   a. Change to subdirectory `examples\postcard` in the MQSeries Integrator home directory. Enter the following command:

      ```
      mqsimrmimpexp -i MQSIMRDB mqsiuid mqsipw PostcardMS.mrp
      ```

      Note: The user ID and password used on this command must be the values that you specified for message repository access when you created the Configuration Manager (flags `-e` and `-r`). If you specified different values for these parameters, you must substitute the correct values here.

      For more details of this command, see the *MQSeries Integrator Administration Guide*.

Step 3. Restart the Configuration Manager (`mqsistart configmgr`). The Configuration Manager can now access the new message set and make it available.

Step 4. Restart the Control Center 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.

Step 5. 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 will 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*.

Step 6. 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.

Step 7. You must 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 in to 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.

Step 8. Now you must 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.

Step 9. 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.
```

Step 10. 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 “Configuring a simple broker domain” on page 25 (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` directory.

You can run this application on Sun Solaris by following these steps:

Step 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 `/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).

Step 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.

You can start the publisher application as follows:

a. Change to the `/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.

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

```
soccerGame Team1 Team2 MQSI_SAMPLE_QM

soccergame Arsenal Manchester_United MQSI_SAMPLE_QM
```

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 forwards it to the publication node. The publication can indicate:
- 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 kill that process[2] 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 therefore not released the queue which 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.

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

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 can run this application on Sun Solaris by following these steps:

Step 1. Start the publisher application.

a. Open a command window and change to the `/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.

Step 2. Start the subscriber application.

a. Change to the `/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.

Step 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 Sun Solaris)” on page 47 contains a single subscriber queue for this application, the default subscriber queue MQSI_SCRIBBLE_SUBSCRIPTION_QUEUE. If you want to start additional subscribers, you must 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.

| If you have a pre 1.8 version of Java on your machine, follow the steps below to invoke scribble:

1. Change to the `/opt/mqsi/jre/bin` directory
2. Enter the command `jre -classpath "$CLASSPATH" Scribble &`

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.

You can run this application by following these steps:

Step 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. You must 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.

Step 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.

Step 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.

Step 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.

Step 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 46. 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 47), 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 simple message flow

This verification scenario illustrates how you define a very simple 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 “Configuring your broker domain” on page 37. 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, sample queue managers, Control Center, listeners, sender channels, and Configuration Manager are running.

The following tasks are described:
- “Creating the MQSeries resources”.
- “Creating a simple message flow”.
- “Assigning the message flow to the broker” on page 56.
- “Testing the message flow” on page 57.

Creating the MQSeries resources

This simple verification tests needs two queues, one for input, the other for output. To create the queues locally on Sun Solaris, type the following from a command line:

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

Creating a simple 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!

Step 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.

Step 2. Select the *Message Flows* view.

Step 3. Right-click on the Message Flows root and select *Create->Message flow*.

Step 4. Enter the name MQSI_TEST. Click **Finish**. The new message flow appears in the tree view.

Step 5. Expand the IBMPrimitives tree to display the supplied nodes.

Step 6. Select the MQSI_TEST message flow in the left-hand pane. Drag and drop an MQInput node into the right-hand pane.

Step 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**.

Step 8. Drag and drop an MQOutput node into the right-hand pane.

Step 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**.

Step 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.

Step 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.

Step 1. Select the *Assignments* view.

Step 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**.

Step 3. Right-click the default execution group and select *Check Out*. This locks the execution group for you.

Step 4. Expand the *Message flows* tree in the center pane. This displays all the message flows available for assignment.

Step 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 only drop a message flow on an execution group (not on the broker itself).

Step 6. Check in the execution group by right-clicking the default execution group in 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.

Step 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*.

Step 2. In the *Assignments* view, right-click the broker name in the left-hand pane.

Step 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.

Step 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).

Step 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

To test your message flow, follow the steps below:

Step 1. Change to the directory containing the MQSeries sample programs

```
/usr/mqm/samp/bin
```

Step 2. Enter the following command:

```
amqsput MQSI_INQ MQSI_SAMPLE_QM
```

Step 3. Press **Enter**.

Step 4. Enter the text of your message.

Step 5. Press **Enter** twice to quit the program.

Step 6. Use either `amqsget` or `amqsbcg` to get the message you typed in. For example:

```
amqsget MQSI_OUTQ MQSI_SAMPLE_QM
```

Step 7. Your message is now displayed.

Step 8. Your test is complete. Congratulations!

Postcard

Appendix A. System changes after installation

This chapter describes 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 "Configuring a simple broker domain" on page 25.

- "Directory structure".
- "Environment variables" on page 60.
- "Database contents" on page 61.
- "Default MQSeries resources" on page 62.

The following are the default home directories:

- MQSeries Integrator

  `/opt/mqsi`
- DB2

  `/opt/IBMdb2`

  Written as `<db2_root>` wherever it appears in this chapter.

Directory structure

The following tables list the subdirectories created and populated within your home directories when you install the product. It also provides a brief description of the contents.

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. /opt/mqsi/ directory structure after installation

Directory Names	Contents
bin	Executable commands
classes	Java class files
CmdAsst	Command assistant files
docs	PDF files
icu	icu data files
include	Header files for samples
jre	Java Runtime Engine
lib	Shared libray files
lil	MQSI lil's
merant	Merant Drivers
messages	Description files for messages and exceptions
sample	Samples

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

Directory Names	Contents
template	Template files
tivready	Tivoli files

Table 4. /var/mqsi directory structure after installation

Directory	Contents
brokers	Brokers directory
log	MQSI logs directory
messages	Messages directory
odbc	ODBC files
registry	Registry files
users	Users directory

Environment variables

The following environment variables must 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. This can be found 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, a database table is created for your broker. It is labelled USERSPACE1.

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.

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
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

Database contents

You must add all this database table into your standard backup and recovery routines to ensure you can recover from system crashes and other emergencies.

This table is maintained by processes that are internal to MQSeries Integrator components. You must not access this table by any other means, nor change the access authority required by MQSeries Integrator. You have no requirement to know or understand the contents of this table.

Default MQSeries resources

When you create MQSeries Integrator components, some MQSeries resources are created for their use. Table 6 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 MQSeries Integrator broker database

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 an 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 setup using SQL*Net following Oracle documentation.
- Oracle operations are conducted within the correct Oracle environment (ORACLE_HOME and so on).

Naming

MQSeries Integrator uses the Data Source Name (DSN) defined in the ODBC set up. 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 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 provide 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 connects you to the SQL*Plus application and show 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 is 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 chapter gives details of the processes that allow you to uninstall any one or all of the MQSeries Integrator components on Sun Solaris.

Before you start

Before you start to uninstall MQSeries Integrator for Sun Solaris Version 2.0.1 you must:

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

Uninstalling

To uninstall on Sun Solaris, use the `pkgrm` command. For example, to remove MQSeries Integrator for Sun Solaris, type the following command:

```
pkgrm mqsi
```

To uninstall a CSD level, you must issue the command:

```
pkgrm mqsi-updnn
```

where `nn` is the update level.

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 Sun Solaris platform syslog or Windows NT Event log entries. 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 Sun Solaris platform syslog or Windows NT Event log entries and First Failure Support Technology(TM) (FFST(TM)) 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 Sun Solaris platform syslog or Windows NT Event log entries and First Failure Support Technology (FFST) output files. Check database product documentation for where to find these files.
- For Sun Solaris:
  - Version. You can find the version of Sun Solaris installed by using the `uname -a` command.
  - Sevice 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 xiv.

You can also download CSDs from this web site.

Applying maintenance to MQSeries Integrator Version 2.0.1

Before applying any maintenance, you must read the file `memo.ptf` and any `readme.txt` files in the root directory of the CD. If you need to apply a CSD:

1. Stop all brokers that are running
2. Change to the directory containing the CSD
3. Type the following command:
   ```
   pkgadd -d .
   ```
4. Select:
   ```
   mqsi-updnn
   ```

   Where `nn` is the level of update.

To see the current level of corrective service, enter the command:

```
pkginfo | grep mqsi
```

After installation of the CSD, you can restart your broker or brokers.

The current level is the highest value of `dnn` in the displayed values of `mqsi-updnn`.

To remove a CSD, enter the command:

```
pkgrm mqsi-updnn
```

Where `nn` is the number of the CSD to remove.

Note: CSDs are cumulative, therefore you do not need to apply CSD1 before you can apply CSD2. When you have installed a CSD, you are prevented from installing a previous CSD without first restoring the system using the backed-up files.

Restoring a previous 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.

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 xiv.

Appendix E. Using NEONFormatter and NEONRules nodes

You need to take some specific actions to use the NEONFormatter and NEONRules nodes after you have installed MQSeries Integrator Version 2.0.1 and its prerequisites (listed in “Software requirements” on page 5).

This appendix leads you through some of those actions and then directs you to specific NEONFormatter and NEONRules information in the MQSeries Integrator Version 1.1 documentation set for further details. The MQSeries Integrator Version 1.1 documentation is provided in the `\Book` subdirectory of the MQSeries Integrator Version 2.0.1 home directory. The default home directory is

```
\Program Files\IBM MQSeries Integrator 2.0.1
```

The actions you need to take are:
- Setting up the database environment
- Setting up tablespaces in the MQSeries Integrator database
- Configuring your database (DB2, Oracle7, and Oracle8)
- User-defined segments (Sybase and SQL Server 6.5)
- Setting up the server (Sybase and SQL Server 6.5)
- Installing the database schema (all databases)
- Editing the database connection file (sqlsvses.cfg)
- Migrating rules and formats (migrating from Version 1.0 to Version 2)

These actions are described in the sections that follow.

Setting up the database environment

The Version 1 NEON packages have been renamed from `mqsi110` to `mqsi40`. A pre-installation script checks to see if the old file is present and issues the following warning if the `mqsi110` file is found:

```
You appear to have an incompatible version of Neon's MQSeries Integrator V1.1
installed. If you are going to re use your rules and formats from this
product you must back up your data, remove 'mqi110' and install updated
package 'mqsi110' from the CD provided.
```

The following warning is issued if you have not installed the `mqsi110` package:

```
You do not have MQSeries Integrator V2.0.1 NEON support installed.
In order to use NEON files and formats with MQSeries Integrator V2.0.1
you must install MQSeries Integrator V2.0.1 NEON support, 'mqsi110',
from the CD provided.
```

Use the following information to check the environment for the database that you are using is set up correctly.

DB2

Verify the following:

- You can connect to an appropriately resourced DB2 database that stores MQSeries Integrator data, either directly or through a DB2 client.

- A DSN (DB2 instance) is defined, using either the ODBC Administrator tool or Client Configuration Assistant, to point to the DB2 database instance.
- The DB2 utility program `db2.exe` is in the execution path for the user doing the install.
- If you do not have DB2 for Windows NT, be sure database `home`, `lib`, and `bin` directories are in the PATH environment variable.

Oracle

Verify the following:

- You can connect to an appropriately resourced Oracle database that stores MQSeries Integrator data, either directly or through an Oracle client.
- You know the Oracle SYS account information.
- The ORACLE_HOME environment variable is set to the location of the database home.
- The PATH environment variable includes the product `bin` directory and the database `bin` directory.

Sybase

On the Windows NT workstation, verify that Sybase Client, Version 11.1.1 is installed. On the Sybase server, verify the following:

- You can connect to an appropriately resourced Sybase database that stores MQSeries Integrator data, either directly or through a Sybase client.
- You are a database owner or know the account information for the owner of the database.
- There is sufficient disk space for your calculated needs.
- The PATH environment variable includes the product `bin` directory and the database bin directory.
- The Sybase utility program `isql` is in the execution path for the user doing the install.

Collecting information

Before beginning the installation, ensure you know the drive letter of the CD-ROM drive from which you will run the installation and the information for the appropriate operating system in the following sections:

DB2
- database alias
- username
- password

Oracle
- SYS userid
- password for SYS userid
- service name

Sybase
- database name

- server name
- username
- password

Schema installation

To prepare the database for database schema installation for Oracle and DB2, refer to the Creating Tablespaces section below. For Sybase and SQL Server, refer to the User-Defined Segments section on page "User-Defined Segments (Sybase)" on page 85.

Creating tablespaces

The database must have MQSeries Integrator tablespaces created before you can install the database schema. You do not have to create tablespaces with Microsoft SQL Server or Sybase Adaptive Server.

Note: The size of your tablespaces depends on the numbers of Rules and Formats used at your site. You might want to place the tablespaces on different physical disks to balance I/O to avoid disk-access bottlenecks. You should separate data tablespaces and index segments by placing them on different disks and/or controllers. This optimizes index and data access parallelism.

DB2 tablespaces

For information on creating a DB2 database, refer to the DB2 installation documentation.

To create DB2 tablespaces:

1. Create the DB2 database for MQSeries Integrator
2. Create the following tablespaces in the database:
   - FORMATTER_DATA
   - FORMATTER_INDEX
   - RULES_DATA
   - RULES_INDEX
3. Grant DBADM privilege on the database to the user who will perform the installation.

Note: If you are using DMS tablespaces, use the Oracle Minimum Size guidelines (as shown in the table in step 2 below) to create minimum DB2 tablespaces.

Configuring DB2

There are three steps required in the configuration of DB2 before MQSeries Integrator can be installed. These are:

1. Create a new database to contain the rules and formats.
2. Configure a client connection to the new database.
3. Create tablespaces within the database for the MQSeries Integrator tables.

The first step must be done using the DB2 instance owning user (db2inst1). The other two steps can be done from a client machine (Windows NT). Once logged on as DB2 user on a UNIX machine, you must set up the .profile file to include the correct paths etc. To set this up, type the following:

```
 $ /home/<db2 instance name>/sqllib/db2profile
```

On Windows NT, there is a command line program that sets up the environment. This can be found in:

```
Start->Programs->DB2 for Windows NT->Command Window
```

There is also a program called Command Line Processor in the same directory that runs the DB2 command line required in steps 1 and 3 on a Windows NT machine.

1. To create a database, log on to the server as a DB2 database instance user and run the db2 command. At the command line, issue the following command:

   ```
   db2=>create database <database name>
   ```

   where <database name> is the name of the MQSeries Integrator database that is required. Use the quit command to exit the DB2 command line program:

   ```
   db2=>quit
   ```

2. To connect to a DB2 database, there must be a database alias set up. DB2 automatically creates an alias to the database on the server when it creates the database. This alias has the same name as that of the database. To connect to the database via a Windows NT client, an alias needs to be made on the client machine. If the server is already on the Windows NT machine, a new database alias must still be made because the default one does not have an ODBC driver associated with it. An ODBC driver is required to connect to the database using the MQSeries Integrator user interface. This can be done using the DB2 program called Client Configuration Assistant found in:

   ```
   Start->Programs->DB2 for Windows NT
   ```

   Run the program and do the following steps:

   a. Click the Add database button.
   b. A window called Add Database SmartGuide appears. Select the ’Manually configure a connection...’ option and click Next.
   c. If you are connecting to a local Windows NT DB2 database:
      1) Select Local for the protocol and click Next
      2) Make sure that the database is on the same drive as shown, then click on Next
   d. If you are trying to connect to a remote server:
      1) Select TCPIP for the protocol and click Next
      2) Type name of the host machine where the database instance is running and the port on which the instance is listening. The port number is in the /etc/services directory. Use the command:

         ```
         $ cat  /etc/services | grep db2
         ```

to display all the ports used by different DB2 instances.

e. Type the name of the database and click Next.

f. Change the alias if you want a different name than that of the database and click Next.

g. The last screen should have a tick in the register database as an ODBC source, and the system data source should be selected.

h. Click Done to create the service and ODBC driver. It asks whether to test the connection.

i. Click Test Connection.

j. Enter the db2admin as user and its associated password. You are then successfully connected.

3. Using a Windows NT machine with DB2 client installed, create a database alias as outlined above in step (1). Start the DB2 command line processor found in:

```
Start->Programs->DB2 for Windows NT
```

Issue the following commands:

```
db2=>connect to <data base alias>
  user <username> using <password>
```

Where <data base name> is the alias name made above, <username> is an authorized user, and <password> is the password.

```
db2=>create tablespace <tablespace name>
managed by system using
('<tablespace name>')
```

where <table space name> is the name of the tablespace to be made. For example, the following line shows how to create one of the tablespaces required by MQSeries Integrator:

```
db2=>create tablespace FORMATTER_INDEX managed by system using
('FORMATTER_INDEX')
```

Repeat this for all required tablespaces.

Use the quit command to exit the DB2 command line program:

```
db2=>quit
```

Now the database is ready for the installation of the Formatter and Rules tables (see Installing Database Schema in the *MQSeries Integrator Version 1.1 Installation and Configuration Guide*). After these are created, the MQSeries Integrator GUIs can connect to the database using the ODBC option in the DBMS section.

Now go to “Installing the database schema (all databases)” on page 87.

Oracle tablespaces

For information on creating an Oracle database, refer to the Oracle installation documentation.

To create Oracle tablespaces:

1. Create a dedicated Oracle instance where the MQSeries Integrator database resides. NEONET is the default Oracle instance name used in the inst_db.cmd file.
2. Create the following tablespaces in the Oracle database:

Table	Minimum size
TOOLS	1 MB
TEMP	10 MB
FORMATTER_DATA	20 MB
FORMATTER_INDEX	20 MB
RULES_DATA	20 MB
RULES_INDEX	20 MB

Configuring Oracle7

There are three steps in the configuration of Oracle7 before MQSeries Integrator can be installed. These are:

1. Create a new database to contain the rules and formats.
2. Configure a client connection to the new database.
3. Create tablespaces within the database for the MQSeries Integrator tables.

These three steps must be performed using the 'oracle' user account that belongs to the dba group.

Step 1: creating the database

Use one of the following methods:

1. Ask the DBA to create a new database with SID of MQSeries Integrator
2. On the UNIX machine on which the Oracle7 server is installed, run the orainst program:

   ```
   $ <ORACLE_HOME>/orainst/orainst
   ```

   and choose options to create a new database. This works only if the Oracle installer is installed. The orainst program is in the `$ORACLE_HOME/orainst` directory.
3. Use the default database that is created when Oracle is installed. This is usually called sid1 or sid.
4. Create a database manually. This is beyond the scope of this document, but is explained in detail in the Oracle7 online documentation.

For UNIX, after the database is created, add the following line to `/etc/oratab` if it does not already exist:

```
<SID of data base>:<ORACLE_HOME directory>:Y
```

If it already exists, ensure the end character is Y. If it is N, change it to Y. To start the database execute:

```
$ dbstart
```

To stop the database (and all other databases on the machine) execute:

```
$ dbshut
```

Step 2: configuring a client connection

The two configuration files that Oracle uses to connect a client to a server are tnsnames.ora and listener.ora. The tnsnames.ora file is found in the `<ORACLE_HOME>/network/admin` directory.

- On HP-UX and AIX® systems, tnsnames.ora file might have been placed in the /etc directory. This file is used to define service names. This definition links a service name to the host name of a server, the SID of a database on that server, and the port on which the listener on the server is listening for client connections.
- On the Windows NT version of Oracle7 client, is a utility called SQL Net Easy (`Start->Oracle for Windows NT`). This modifies the tnsnames.ora file found in the `<ORACLE_HOME>\network\admin` directory.

The best way to create the file on UNIX is to use SQL Net Easy on Windows NT, and then copy the tnsnames.ora file across to the `<ORACLE_HOME>/network/admin directory` on the UNIX machine using ftp.

Here is an example of a tnsnames.ora entry:

```
#
#  An example tnsnames.ora file
#

  <Service name> =
  (DESCRIPTION =
    (ADDRESS =
          (COMMUNITY = tcpcom.world)
          (PROTOCOL = TCP)
          (HOST = <Machine name>)
          (PORT = <Port number>)
    )
    (CONNECT_DATA =
      (SID = <SID of database>)
    )
  )

#
#  End of tnsnames.ora file
#
```

Where <Machine name> is replaced by the name of the server machine, <Port number> is the port on which the listener is listening for connections, and <SID of   database> is the SIDof the database that the service connects to. <Service name> can be any name, but must be unique within the tnsnames.ora file.

The listener.ora file contains the information required for the system to start listener processes that allow clients to connect to the database. This must contain the port address where the listener listens for client connections, the SID of the database, and the address of the Oracle home directory. This file should be created when a database is created by the installer; if not, the file must be configured manually. To find if any listeners are running, run the Oracle command:

```
$ <ORACLE_HOME>/bin/lsnrctl status
```

If there is no listener running for your database, add the following lines to the listener.ora file:

```
#
# Sample listener.ora file
#
# To start listener run: lsnrctl start mylsnr
# To stop listener run: lsnrctl stop mylsnr
# To check whether the listener is working run: lsnrctl status
mylsnr
#

mylsnr=
  (ADDRESS_LIST =
        (ADDRESS= (PROTOCOL= IPC)(KEY= <Service name>))
        (ADDRESS= (PROTOCOL= TCP)(Host= <Machine name>)
 (Port= <Port number>))
  )


SID_LIST_mylsnr=
 (SID_LIST=
  (SID_DESC=
   (SID_NAME=<SID of data base>)
   (ORACLE_HOME=<Oracle home directory>)
   (PRESPAWN_MAX=10)
  )
 )

STARTUP_WAIT_TIME_mylsnr = 0
CONNECT_TIMEOUT_mylsnr = 10
TRACE_LEVEL_mylsn = OFF

#
#  End of listener.ora file
#
```

Where <SID of database>, <Machine name>, and <Port number> correspond to the same entries as in tnsnames.ora. The word mylsnr can be changed to any word except LISTENER. The listener can now be started using the command:

```
<ORACLE_HOME>/bin/lsnrctl start mylsnr
```

and stopped with the command:

```
<ORACLE_HOME>/bin/lsnrctl stop mylsnr
```

There should now be a tnsnames.ora file on the client machine and a listener running on the server machine. Copy the tnsnames.ora file to the server so that you can connect to the database while on the server. To test the connection, run the svrmgrm utility (found in the <ORACLE_HOME>/bin directory). Enter the value system as the user name and its password and the service name created above. If it says:

```
ORA-12154: TNS: could not resolve service name
```

try the following:
1. Verify the tnsnames.ora is in the correct directory
2. Verify the correct service name is in the tnsnames.ora file
3. Verify there are no missing brackets or other typing errors

If the service name is resolved, but when the client attempts to connect to the database, the following error is returned:

```
ORA-12203: TNS: unable to connect to destination
```

try the following:
1. Verify the database is running (see section one above)
2. Verify the listener is running on the same port as the tnsnames.ora service and that the listener has the correct SID entry (run: lsnrctl status mylsnr)

If the connection is successful, the database is now configured for client connections from any machine that has the tnsnames.ora entry given above. This should be on the systems that MQSeries Integrator server and the MQSeries Integrator GUI client are installed on.

Step 3: creating tablespaces

To create tablespaces on Windows NT, use either the sqlplus plus33 command line or the svrmgrm utility. On Windows NT Oracle 7.3.3 and later, the utility Storage Manager can be used instead of svrmgrm.

For sqlplus, issue the following commands:

```
$ <ORACLE_HOME>/bin/sqlplus
 Enter user-name:sys
 Enter password:<sys_password>
 SQL>connect sys/<sys_password>@<Service name>
 SQL>create tablespace FORMATTER_DATA datafile 'FORMATTER_DATA.one'
     size 20M;
 SQL>create tablespace FORMATTER_INDEX datafile 'FORMATTER_INDEX.one'
     size 20M;
 SQL>create tablespace RULES_DATA datafile 'RULES_DATA.one'
     size 20M;
 SQL>create tablespace RULES_INDEX datafile 'RULES_INDEX.one'
     size 20M;
 SQL>create tablespace TOOLS datafile 'TOOLS.one'
     size 10M;
 SQL>alter tablespace TEMP add datafile 'TEMP_INCREASE_SIZE.one'
     size 1M;
 SQL>quit
```

Oracle tablespaces

For the svrmgrm utility, do the following:

```
$ <ORACLE_HOME>/bin/svrmgrm
```

1. Enter the sys user, sys password, service name and select connect as NORMAL
2. Select tablespace folder
3. Click on the tablespace menu and select Create Tablespace
4. In the name box add the tablespace name, for example:
   ```
   FORMATTER_DATA
   ```
5. Click New next to the Datafiles box
6. Enter a name for the data file, for example:
   ```
   FORMATTER_DATA.one
   ```
7. Put 20 in new file size and select MB from the side menu
8. Click OK to create the file
9. Finally click Create to make the tablespace

Repeat steps 1 to 9 for FORMATTER_INDEX, RULES_DATA, and RULES_INDEX.

Repeat steps 5 to 9 for a data file TEMP using a filesize of 10 MB.

Repeat steps 5 to 9 for a data file TOOLS using a filesize of 1 MB.

The database is now ready for the MQSeries Integrator install script called inst_db.sh (inst_db.bat on Windows NT). The use of this is described in the *MQSeries Integrator Version 1.1 Installation and Configuration Guide*.

Now go to “Installing the database schema (all databases)” on page 87.

Configuring Oracle8

There are three steps in the configuration of Oracle8 before MQSeries Integrator can be installed. These are:
1. Create a new database to contain the rules and formats.
2. Configure a client connection to the new database.
3. Create tablespaces within the database for the MQSeries Integrator tables.

These three steps must be performed using the ’oracle’ user account that belongs to the dba group.

Step 1: creating the database

Use one of the following methods:
1. Ask the DBA to create a new database with SID of MQSI
2. On the UNIX machine on which the Oracle8 server is installed, run the orainst program:
   ```
   $ <ORACLE_HOME>/orainst/orainst
   ```

   and choose options to create a new database. This works only if the Oracle installer is installed. The orainst program is in the `$ORACLE_HOME/orainst` directory.

3. Use the default database that is created when Oracle is installed. This is usually called sid1, (ORCL on Windows NT).
4. On Windows NT, run the Oracle Database Assistant, found in:
   ```
   Start->Programs->Oracle for Windows NT
   ```
5. Create a database manually. This is beyond the scope of this document, but is explained in detail in the Oracle8 online documentation.

For UNIX, after the database is created, add the following line to `/etc/oratab` if it does not already exist:

```
<SID of data base>:<ORACLE_HOME directory>:Y
```

If it already exists, ensure the end character is Y. If it is N, change it to Y. To start the database execute:

```
$ dbstart
```

To stop the database (and all other databases on the machine) execute:

```
$ dbshut
```

Step 2: configuring a client connection

The two configuration files that Oracle uses to connect a client to a server are tnsnames.ora and listener.ora.

- On UNIX the tnsnames.ora file is found in the <ORACLE_HOME>/network/admin directory, and on Windows NT the tnsnames.ora is found in the <ORACLE_HOME>/net80/admin directory. On HP-UX and AIX systems, tnsnames.ora might have been placed in the /etc directory. This file is used to define service names. This definition links a service name to the host name of a server, the SID of a database on that server, and the port on which the listener on the server is listening for client connections.
- On the Windows NT version of Oracle8 client, there is a utility program called Oracle Net8 Assistant (Start->Oracle for Windows NT). This modifies the tnsnames.ora file that can be found in the `<ORACLE_HOME>\network\admin` directory.

The best way to create the file on UNIX is to use Oracle Net8 Assistant on Windows NT, and then copy the tnsnames.ora file across to the `<ORACLE_HOME>/network/admin` directory on the UNIX machine using ftp.

Here is an example of a tnsnames.ora entry:

```
#
#  An example tnsnames.ora file
#

  <Service name> =
  (DESCRIPTION =
    (ADDRESS =
          (COMMUNITY = tcpcom.world)
          (PROTOCOL = TCP)
          (HOST = <Machine name>)
          (PORT = <Port number>)
```

```
    )
    (CONNECT_DATA =
      (SID = <SID of database>)
    )
  )

#
#  End of tnsnames.ora file
#
```

where <Machine name> should be replaced by the name of the server machine, <Port number> is the port on which the listener is listening for connections and <SID of database> is the SID of the database that the service connects to. <Service name> can be any name but must be unique within the tnsnames.ora file.

The listener.ora file contains the information required for the system to start listener processes that allow clients to connect to the database. This must contain the port address where the listener listens for client connections, the SID of the database, and the address of the Oracle home directory. This file should be created when a database is created by the installer; if not, the file must be configured manually. To find if any listeners are running run the Oracle command:

```
<ORACLE_HOME>/bin/lsnrctl status
```

If there is no listener running for your database, add the following lines to the listener.ora file:

```
#
# Sample listener.ora file
#
# To start listener run: lsnrctl start mylsnr
# To stop listener run:  lsnrctl stop mylsnr
# To check whether the listener is working run: lsnrctl status
mylsnr
#

mylsnr=
  (ADDRESS_LIST =
        (ADDRESS= (PROTOCOL= IPC)(KEY= <Service name>))
        (ADDRESS= (PROTOCOL= TCP)(Host= <Machine name>)
 (Port= <Port number>))
  )


SID_LIST_mylsnr=
 (SID_LIST=
  (SID_DESC=
   (SID_NAME=<SID of data base>)
   (ORACLE_HOME=<Oracle home directory>)
   (PRESPAWN_MAX=10)
  )
 )

STARTUP_WAIT_TIME_mylsnr = 0
CONNECT_TIMEOUT_mylsnr = 10
```

```
TRACE_LEVEL_mylsn = OFF

#
#  End of listener.ora file
#
```

Where <SID of database>, <Machine name>, and <Port number> correspond to the same entries as in tnsnames.ora. The word mylsnr can be changed to any word except LISTENER. The listener can now be started using the command:

```
<ORACLE_HOME>/bin/lsnrctl start mylsnr
```

and stopped with the command:

```
<ORACLE_HOME>/bin/lsnrctl stop mylsnr
```

There should now be a tnsnames.ora file on the client machine and a listener running on the server machine. Also copy the tnsnames.ora file to the server so that you can connect to the database while on the server. To test the connection, run the sqlplus utility (found in the <ORACLE_HOME>/bin directory). Enter the value system as user name followed by its password. At the command prompt, type:

```
sqlplus>connect sys/<sys password>@<service name>
```

If the following message is returned:

```
ORA-12154: TNS: could not resolve service name
```

try the following:
1. Verify the tnsnames.ora is in the correct directory
2. Verify the correct service name is in the tnsnames.ora file
3. Verify There are no missing brackets or other typing errors

If the service name is resolved but when the client attempts to connect to the database, the following error is returned:

```
ORA-12203: TNS: unable to connect to destination
```

try the following:
1. Verify the database is running (see step one above)
2. Verify the listener is running on the same port as the tnsnames.ora service and that the listener has the correct SID entry (run: lsnrctl status mylsnr)

If the connection is successful, the database is now configured for client connections from any machine that has the tnsnames.ora entry given above. This should be on the systems that MQSeries Integrator server and the MQSeries Integrator GUI client are installed on.

Step 3: creating table spaces

To create tablespaces on Windows NT, use either the sqlplus plus80 command line or the Oracle Storage Manager utility.

Oracle tablespaces

For the sqlplus, issue the following commands:

```
$ <ORACLE_HOME>/bin/sqlplus
 Enter user-name:sys
 Enter password:<sys_password>
 SQL>connect sys/<sys_password>@<Service name>
 SQL>create tablespace FORMATTER_DATA datafile 'FORMATTER_DATA.one'
     size 20M;
 SQL>create tablespace FORMATTER_INDEX datafile 'FORMATTER_INDEX.one'
     size 20M;
 SQL>create tablespace RULES_DATA datafile 'RULES_DATA.one'
     size 20M;
 SQL>create tablespace RULES_INDEX datafile 'RULES_INDEX.one'
     size 20M;
 SQL>create tablespace TEMP datafile 'TEMP.one'
     size 10M;
 SQL>create tablespace TOOLS datafile 'TOOLS.one'
     size 1M;
 SQL>quit
```

For the Oracle Storage Manager, do the following:

1. Enter the sys user, sys password, service name and select connect as NORMAL
2. Select tablespace folder
3. Click on the tablespace menu and select Create Tablespace
4. In the name box add the tablespace name, for example:
   ```
   FORMATTER_DATA
   ```
5. Click Add next to the Datafiles box
6. Enter a name for the data file, for example:
   ```
   FORMATTER_DATA.one
   ```
7. Put 20 in new file size and select MB from the side menu
8. Click OK to create the file
9. Finally click Create to make the tablespace

Repeat steps 1 to 9 for FORMATTER_INDEX, RULES_DATA, and RULES_INDEX.

Repeat steps 1 to 9 for a data file TEMP using a filesize of 10 MB.

Repeat steps 1 to 9 for a data file TOOLS using a filesize of 1 MB.

The database is now ready for the MQSeries Integrator install script called inst_db.sh (inst_db.bat on Windows NT). The use of this is described in the *MQSeries Integrator Version 1.1 Installation and Configuration Guide*.

Now go to “Installing the database schema (all databases)” on page 87.

User-Defined Segments (Sybase)

Sybase databases must have user-defined segments created before you can install the database schema. User-defined segments provide a mapping from the database tables and indexes to the underlying disk space on which the database resides.

You might want to place the user-defined segments on different physical disks to balance I/O and avoid disk-access bottlenecks. You should separate data and index segments by placing them on different disks, or controllers, or both. This optimizes index and data access parallelism.

Create the following user-defined segments in the database that will contain MQSeries Integrator tables and stored procedures:
- FORMATTER_DATA
- FORMATTER_INDEX
- RULES_DATA
- RULES_INDEX

The commands necessary to perform this are in the following vendor documentation: *SQL Server Reference Manual* for Sybase

The commands are:
- sp_addsegment
- sp_dropsegment
- sp_extendsegment

Setting up the server

Use the following information to set up your server.

Sybase

There are six pieces of information you need to access your Sybase Server database from MQSeries Integrator. These are:
1. User ID
2. User password
3. Server name
4. Server communication protocol (for example, TCP)
5. Server address
6. Database name

The first step is to create a database for MQSeries Integrator to use. This can be done using the isql command line utility, either on the machine on which Sybase Server is running, or remotely via a Sybase Client.

To start isql, type the following command:

```
isql -U<username> -P<password> -S<servername>
```

The default servername is the name of the machine on which the server was installed. The default System Administrator ID is username ″sa″ with no password (enter ″-P″ on its own, with no subsequent password value). You are recommended to set a password, both for security reasons, and because the MQSeries Integrator Rules Engine does not accept a null password value. This can be done with the following commands:

```
1. sp_password <old password>,<new password>
2. go
```

Where <old password> is entered as ″NULL″ if no password is currently set.

You can either use this System Administrator ID to create the database and access it from MQSeries Integrator, or you can create other usernames. MQSeries Integrator requires that the username it logs on to the Sybase Server with must be the owner of the database it is intended to access. By default, the owner of a database is the user that created it.

If you encounter problems starting isql, the Sybase Server might not be running. If you suspect that it is not, change to the /install directory off the Sybase root directory, and execute the

```
"RUN_<servername>"
```

file, where <servername> is the name of your server.

Now create the database using the following isql commands:

```
1. create database <database name> on <device name> = <size in MB>
2. go
```

Unless you have a particular preference, enter ″master″ for <device name>. Ten MB should be a sufficient size for most applications of MQSeries Integrator, but if you require more precise calculation details, see the *MQSeries Integrator Version 1.1 Installation and Configuration Guide*.

For the MQSeries Integrator inst_db.sh script to run, you must create the user-defined segments described in the *MQSeries Integrator Version 1.1 Installation and Configuration Guide*. To do this, use the sp_addsegment command:

```
1. sp_addsegment <segment name>,<database name>,<device name>
2. go
```

Repeat this for all segments. Again, unless you have a preference, use ″master″ for the device.

The database can be accessed directly from the machine on which the Sybase Server resides, but unless you are performing a Windows NT-only installation, you will need to connect to it remotely with the MQSeries Integrator user interfaces from a Windows NT machine.

First run the dscp command line utility on the Server machine, and enter the following commands:

```
>> open interfaces
>> read <servername>
```

This displays information about the server specified by <servername>, including the communications protocol, address, and, if appropriate, port number on which it is listening. You must know this information to configure the client.

On the Windows NT machine, run the Sybase Dsedit utility from the Start menu. Open the InterfacesDriver directory service and add a Server Object with the name and communication protocol/address of your server. Remember, if appropriate (for example, with TCP), to include the port number on which the server is listening - <address>,<port> is the correct format for entering this in the Network Address field. Exit the Dsedit utility.

You now have all the information that MQSeries Integrator requires to access its Sybase Server database. Ensure that the Sybase Server is running and continue with MQSeries Integrator installation.

Installing the database schema (all databases)

The inst_db script creates the necessary tables and stored procedures in the MQSeries Integrator database. The script sends the commands from the files in the install.sql directory.

To install the database schema:

1. Change to the **/opt/mqi110/install.sql** directory
2. To build the MQSeries Integrator schema, type one of the following:
   - **DB2**
     - Execute the `inst_db.sh` script using the following syntax:
       ```
       inst_db.sh <username> <password> <database>
       ```
   - **Oracle**
     - You must be SYS user to run `inst_db.sh`
       ```
       inst_db.sh SYS <SYS password> <servicename>
       ```
   - **Sybase**
     - `inst_db.sh <userid> <password> <servername> <dbinstance>`
3. While the script runs, look for errors.
4. When the script completes the instantiation, a verification message appears.
5. For installation details and to see if there are errors, refer to the `inst_db.log` file. You must always check the log, even if the verification message says the instantiation completed successfully.

   Note: In the inst_db.log file, the error ″table or view does not exist″ does not indicate a problem with database instantiation. The database is successfully instantiated if this is the only error you receive.

   In addition to the `inst_db.log` file, an **NNFIE.log** may appear. This log only appears if there are failure messages; it will not appear otherwise.

|

Editing the database connection file

Some MQSeries Integrator executables connect to the database using the database connection file sqlsvses.cfg. This file contains entries for DBMS sessions that detail the server name, user id, password, and database name that a particular session uses. Executables search the sqlsvses.cfg file for a given session name and attempt to connect to the MQSeries Integrator database (for example, msgtest searches for new_format_demo).

A sample sqlsvses.cfg file that is commented out is provided in the bin directory. Uncomment the section that applies to your DBMS type. You must edit the sample file with your site-specific information. This file enables certain NEONRules and NEONFormatter executables to connect to the database. For more information, refer to the *MQSeries Integrator Version 1.1 System Management Guide*.

To edit the database connection file:
1. Change to the `bin` directory
2. In the `bin` directory, locate the text file sqlsvses.cfg
3. In the sqlsvses.cfg file, edit the following:

DB2

```
<sessionname>:<dbalias>:<username>:<password>:
```

For example:

```
new_format_demo:dodge:neonuser:neonpwd:
rules:dodge:neonuser:neonpwd:
```

Oracle

```
<sessionname>:<servicename>:<username>:<password>:
```

For example:

```
new_format_demo:dodge::neonuser:neonpwd:
rules:dodge:neonuser:neonpwd:
nnfie:dodge:neonuser:neonpwd:
nnrmie:dodge:neonuser:neonpwd:
```

The default for both the Oracle username and password is NEONET.

Migrating rules and formats

This section explains how to migrate the data from an MQSeries Integrator Version 1.0 database to an MQSeries Integrator Version 2.0.1 compatible database. The NNRie and NNFie utilities are used to migrate rules and formats.

Notes:

1. This migration procedure assumes that you created rules and formats in MQSeries Integrator Version 1.0.

2. This procedure also assumes that the database to which you are migrating rules and formats is a clean database. If you are migrating to a database that already contains data, data conflicts might occur.
3. This procedure is only required for rules and formats from Version 1.0. Rules and formats from Version 1.1 do not need to be migrated.
4. In this appendix, reference is made to executable files having a suffix of `40`. This is the case, only, if the Version 1.0 code has been removed from your machine.

NNRie overview

Use the NEONRules Import/Export Utility, NNRie, to export existing rules from an MQSeries Integrator database and import them to an MQSeries Integrator Version 2.0.1 database.

Using NNRie, you can:

- Export a single rule identified by its corresponding application name, message type, and rule name.
- Export a single subscription identified by its corresponding application name, message type, and subscription name.
- Export entire rule sets, rules, and subscriptions identified by corresponding application group and message type names.
- Export all message types and their rule sets identified by the message types application group name.
- Export all application groups and their associated message types and rules.

This program creates an export file that can be interchanged between platforms. All application groups and their associated message types and rules should be exported. The exported file can then be imported to the Version 2 database using NNRie Version 2.

NNFie overview

The NEONFormatter Import/Export Utility, NNFie, is used to export existing formats from an MQSeries Integrator Version 1 database and to import the formats to an MQSeries Integrator database.

Using NNFie, you can:

- Export existing formats into a file that can be interchanged between platforms.
- Import the file into the database. NNFie can import a file created by exporting from an MQSeries Integrator formats database.

Additional information regarding NNFie is available in the *MQSeries Integrator Version 1.1 System Management Guide*.

Migrating your rules and formats

When you migrate rules and formats to a Version 2 database, you are recommended to use two separate databases. You are not recommended to rebuild the schema after export and importing into the same database.

Migrating rules and formats

Verify the following before migrating rules and formats:

- There is enough disk space to hold the output file. This file can be re-directed anywhere the system supports.
- The target database (MQSeries Integrator Version 2.0.1 database) has been instantiated.

The following steps are required to migrate your rules and formats from a Version 1.0 database to a Version 2 database.

1. Set up your environment
2. Check the consistency of your Version 1.0 database using the Consistency Checker version for the database from which you are exporting
3. Export your data from the Version 1.0 database
4. Modify your environment for import
5. Import your formats into the Version 2 database
6. Check the consistency of your database using the Consistency Checker

Setting up the environment

The sqlsvses.cfg file

Create the sqlsvses.cfg file. This file is used by the import and export utilities to create a session with the database server. Place this file in a the bin directory where the NNFie and NNRie utilities are located.

Note: One sqlsvses.cfg can be created and used for both NNFie, which migrates formats, and NNRie, which migrates rules.

Oracle

The sqlsvses.cfg file must contain:

```
nnfie:<dbms instance>:<user>:<password>:
nnrmie:<dbms instance>:<user>:<password>:
```

Sybase

The sqlsvses.cfg file must contain:

```
nnfie:<dbms instance>:<user>:<password>: <database>
nnrmie:<dbms instance>:<user>:<password>: <database>
```

DB2

The sqlsvses.cfg file must contain:

```
nnfie:<database name or alias>:<user>:<password>:
nnrmie:<database name or alias>:<user>:<password>:
```

Environment variables

Oracle: Set the following:

```
ORACLE_SID=<servername>
ORACLE_HOME=<your Oracle database directory>
```

Sybase: Set the following:

```
SYBASE=<your sybase database directory>
```

DB2: Set the following:

```
DB2INSTANCE=<your DB2 instance name>
```

Checking consistency

For NEONRules and NEONFormatter, run the Consistency Checker that matches the version of the database from which you are exporting. The following examples use the suffix 110 to match a Version 1.10 or Version 2.0.1 database. Use the suffix 40 to match a Version 1.0 database. You might want to direct the output to a file, as shown in the examples below, rather than standard out because the output from the scripts could be substantial.

Oracle

- **UNIX**

```
formatcc110.ksh <user> <password> <instance> >formatcc110.log
rulecc110.ksh <user> <password> <instance> >rulecc110.log
```

- **Windows NT**

```
formatcc110.cmd <user> <password> <instance> >formatcc110.log
rulecc110.cmd <user> <password> <instance> >rulecc110.log
```

Sybase

- **UNIX**

```
formatcc110.ksh <user> <password> <dbms instance> <database>>formatcc110.log
rulecc110.ksh <user> <password> <dbms instance> <database> >rulecc110.log
```

- **Windows NT**

```
formatcc110.cmd <user> <password> <dbms instance> <database> >rulecc110.log
rulecc110.cmd <user> <password> <dbms instance> <database> >rulecc110.log
```

DB2

- **UNIX**

```
formatcc110.ksh <user> <password> <database name or alias> >formatcc110.log
rulecc110.ksh <user> <password> <database name or alias> >rulecc110.log
```

- **Windows NT**

```
formatcc110.cmd <user> <password> <database name or alias> >formatcc110.log
rulecc110.cmd <user> <password> <database name or alias> >rulecc110.log
```

Exporting rules and formats

To export formats from your 1.0 database, run NNFie and to export rules, run NNRie, using the version that matches the version of the database from which you are exporting.

The data is exported to files named NEONet.fie and NEONet.rie.

DBMS platforms, MQSeries Integrator Version 1.0 database:.

```
NNFie10 -e NEONet.fie -s nnfie
NNRie10 -e NEONet.rie -s nnrmie
```

Modifying your environment for import

For all DBMS platforms, modify your sqlsvses.cfg file to change the instance, username, password, and database (for Sybase and SQL Server) parameters to reflect the MQSeries Integrator Version 2.0.1 database into which you are importing rules and formats.

Importing rules and formats into your Version 2 database

To import formats run NNFie Version 2 and to import rules, run NNRie Version 2. Make sure you have instantiated the Version 2 database before you attempt to import rules and formats.

- **UNIX**

  ```
  NNFie -i NEONet.fie -s nnfie
  NNRie -i NEONet.rie -s nnrmie
  ```

- **Windows NT**

  ```
  NNFie.cmd -i NEONet.fie -s nnfie
  NNRie.exe -i NEONet.rie -s nnrmie
  ```

Using NEONRules and NEONFormatter

When you have completed these actions, follow this checklist to fully use the functions of the NEONRules and NEONFormatter nodes.

- Test your installation by starting the NEONFormatter user interface. To do this, select *Start->Programs->IBM MQSeries Integrator 2.0.1->MQSIV1.1>NEONFormatter*.

  Follow the information that is described in Chapter 3 ″Formatter″ in the *MQSeries Integrator Version 1.1 User's Guide*. to define the function you require.
- You can then use the NEONRules user interface. To do this, click *Start->Programs->IBM MQSeries Integrator 2.0.1->MQSIV1->NEONRules*.

  Follow the information that is described in Chapter 4 ″Rules″ in the *MQSeries Integrator Version 1.1 User's Guide* to define the function you require.
- You can then use the NEON Visual Tester to verify your NEONFormatter and NEONRules definitions.

  Follow the information that is described in Chapter 5 ″Visual Tester″ in the *MQSeries Integrator Version 1.1 User's Guide*. (You can ignore the information about recaching Rules/Formatter in the section called ″General Options″.)
- Use the NEONFormatter functions to parse separate input messages into individual fields and to transform input messages into an output message with a different format.

  Follow the information that is described in Chapter 3 ″Formatter″ in the *MQSeries Integrator Version 1.1 System Management Guide*. (You can ignore the information in the section called ″Shared Libraries/DLLs″.)

- Use the NEONRules functions to evaluate messages, based on your chosen criteria.

  Follow the information that is described in Chapter 4 ″Rules″ in the *MQSeries Integrator Version 1.1 System Management Guide*. (You can ignore the information in the section called ″Shared Libraries/DLLs″.)

Appendix F. 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.

Trademarks

Glossary of terms and abbreviations

This glossary defines MQSeries® Integrator terms and abbreviations used in this book. If you do not find the term you are looking for, see the index or the *IBM® Dictionary of Computing*, New York: McGraw-Hill, 1994.

This glossary includes terms and definitions from the *American National Dictionary for Information Systems*, ANSI X3.172-1990, copyright 1990 by the American National Standards Institute. Copies may be ordered from the American National Standards Institute, 11 West 42 Street, New York, New York 10036. Definitions are identified by the symbol (A) after the definition.

A

Access Control List (ACL). The list of principals that have explicit permissions (to publish, to subscribe to, and to request persistent delivery of a publication message) against a topic in the topic tree. The ACLs define the implementation of topic-based security.

ACL. Access Control List.

AMI. Application Messaging Interface.

Application Messaging Interface (AMI). The programming interface provided by MQSeries that defines a high level interface to message queuing services. See also *MQI* and *JMS*.

B

blob. Binary Large OBject. A block of bytes of data (for example, the body of a message) that has no discernible meaning, but is treated as one solid entity that cannot be interpreted. Also written as BLOB.

broker. See *message broker*.

broker domain. A collection of brokers that share a common configuration, together with the single Configuration Manager that controls them.

C

callback function. See *implementation function*.

category. An optional grouping of messages that are related in some way. For example, messages that relate to a particular application.

check in. The Control Center action that stores a new or updated resource in the configuration or message respository.

check out. The Control Center action that extracts and locks a resource from the configuration or message respository for local modification by a user. Resources from the two repositories can only be worked on when they are checked out by an authorized user, but can be viewed (read only) without being checked out.

collective. A hyperconnected (totally connected) set of brokers forming part of a multi-broker network for publish/subscribe applications.

configuration. In the broker domain, the brokers, execution groups, message flows and message sets assigned to them, topics and access control specifications.

Configuration Manager. A component of MQSeries Integrator that acts as the interface between the configuration repository and an executing set of brokers. It provides brokers with their initial configuration, and updates them with any subsequent changes. It maintains the broker domain configuration.

configuration repository. Persistent storage for broker configuration and topology definition.

connector. See *message processing node connector*.

content-based filter. An expression that is applied to the content of a message to determine how the message is to be processed.

context tag. A tag that is applied to an element within a message to enable that element to be treated differently in different contexts. For example, an element could be mandatory in one context and optional in another.

Control Center. The graphical interface that provides facilities for defining, configuring, deploying, and monitoring resources of the MQSeries Integrator network.

D

datagram. The simplest form of message that MQSeries supports. Also known as *send-and-forget*. This type of message does not require a reply. Compare with *request/reply*.

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 Integrator.

exception list. A list of exceptions that have been generated during the processing of a message, with supporting information.

execution group. A named grouping of message flows that have been assigned to a broker. The broker is guaranteed to enforce some degree of isolation between message flows in distinct execution groups by ensuring that they execute in separate address spaces, or as unique processes.

Extensible Markup Language (XML). A W3C standard for the representation of data.

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 Integrator writes records on the local system. On Windows NT, this is the Event log. On UNIX® systems, this is the syslog. See also *system log*. Note that MQSeries records many events in the log that are not errors, but information about events that occur during operation, for example, successful deployment of a configuration.

M

message broker. A set of execution processes hosting one or more message flows.

messages. Entities exchanged between a broker and its clients.

message dictionary. A repository for (predefined) message type specifications.

message domain. The source of a message definition. For example, a domain of MRM identifies messages defined using the Control Center, a domain of NEON identifies messages created using the NEON user interfaces.

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 Integrator messages (BIPxxx) that refers to the local error logs to which records are written on the local system. On Windows NT, this is the Event log. On UNIX systems, this is the syslog. See also *local error log*.

T

terminal. The point at which one node in a message flow is connected to another node. Terminals enable you to control the route that a message takes, depending whether the operation performed by a node on that message is successful.

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

topic based subscription. A subscription specified by a subscribing application that includes a topic for filtering of publications.

topic security. The use of ACLs applied to one or more topics to control subscriber access to published messages.

topology. In the broker domain, the brokers, collectives, and connections between them.

transform. A defined way in which a message of one format is converted into one or more messages of another format.

U

Uniform Resource Identifier. The generic set of all names and addresses that refer to World Wide Web resources.

Uniform Resource Locator. A specific form of URI that identifies the address of an item on the World Wide Web. It includes the protocol followed by the fully qualified domain name (sometimes called the host name) and the request. The Web server typically maps the request portion of the URL to a path and file name. Also known as Universal Resource Locator.

URI. Uniform Resource Identifier

URL. Uniform Resource Locator

User Name Server. The MQSeries Integrator component that interfaces with operating system facilities to determine valid users and groups.

utility function. Function provided by MQSeries Integrator for the benefit of third-party developers writing plug-in nodes or parsers.

W

warehouse. A persistent, historical datastore for events (or messages). The **Warehouse** node within a message flow supports the recording of information in a database for subsequent retrieval and processing by other applications.

wildcard. A character that can be specified in subscriptions to match a range of topics. See also *multilevel wildcard* and *single-level wildcard*.

wire format. This describes the physical representation of a message within the bit-stream.

W3C. World Wide Web Consortium. An international industry consortium set up to develop common protocols to promote evolution and interoperability of the World Wide Web.

X

XML. Extensible Markup Language.

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.

GC34-5842-01