WebSphere® MQ Integrator Enabler

Installation and Setup Guide

NOTE:

Before using this information and the product it supports, read the information in "Appendix C, *Notices*" on page 76.

Fifth Edition (June 2002)

Printed in USA.

Contents

Figures

About this book

This publication:

- Introduces WebSphere MQ Integrator Enabler (WMQI Enabler).
- Identifies the prerequisite hardware and software applications used by WMQI Enabler.
- Explains how to install WMQI Enabler.
- Describes how to customize WMQI Enabler.
- Shows how to test the installation of WMQI Enabler.

Who should read this book

This publication is intended for installation specialists, configuration specialists, and planners.

Terminology used in this book

All new terms introduced in this book are defined in the *Glossary*.

This book uses the following shortened names:

- MQSeries®: a general term for IBM MQSeries messaging products.
- DB2®: a general term to encompass IBM DB2 Universal Database® Enterprise Edition, Connect Enterprise Edition, and Extended Enterprise Edition.

Prerequisite and related information

No prerequisite reading is required although a knowledge of the Microsoft Windows 2000®, AIX®, or SOLARIS® operating systems and working knowledge of IBM's DB2 7.2, MQSeries 5.2, WebSphere MQ Integrator 2.1, and MQSeries Workflow 3.3.2 products are necessary. Review the following publications for additional information.

- *Development Guide*
- *Application Integration Guide*

How to get additional information

Visit the following Web site at:

http://www.ibm.com/software/mqseries/support/

By following this link you can find:

- The latest information about MQSeries family of products.
- Download SupportPacks.
- Access FAQs.
- Access MQSeries family publications library.

How to send your comments

Your feedback is important in helping to provide the most accurate and high-quality information. If you have any comments or suggestions about this book or any other *WebSphere MQ Integrator Enabler* documentation:

Your feedback is important in helping to provide the most accurate and high-quality information. If you have any comments or suggestions about this book or any other *WebSphere MQ Integrator Enabler* documentation:

- By mail, to this address:

  User Technologies Department (MP095)
  IBM United Kingdom Laboratories
  Hursley Park
  WINCHESTER,
  Hampshire
  SO21 2JN
  United Kingdom

- By fax:

  - From outside the U.K., after your international access code use 44-1962-816151
  - From within the U.K., use 01962-816151

- Electronically, use the appropriate network ID:

  - IBM Mail Exchange: GBIBM2Q9 at IBMMAIL
  - IBMLink: HURSLEY(IDRCF)
  - Internet: idrcf@hursley.ibm.com

Whichever method you use, ensure that you include:

- The publication title and order number
- The topic to which your comment applies

Your name and address / telephone number / fax number / network ID

Chapter 1
Introduction

WebSphere MQ Integrator Enabler (WMQI Enabler) is designed to integrate various disparate applications using a hub-and-spoke architecture as shown in the figure below:

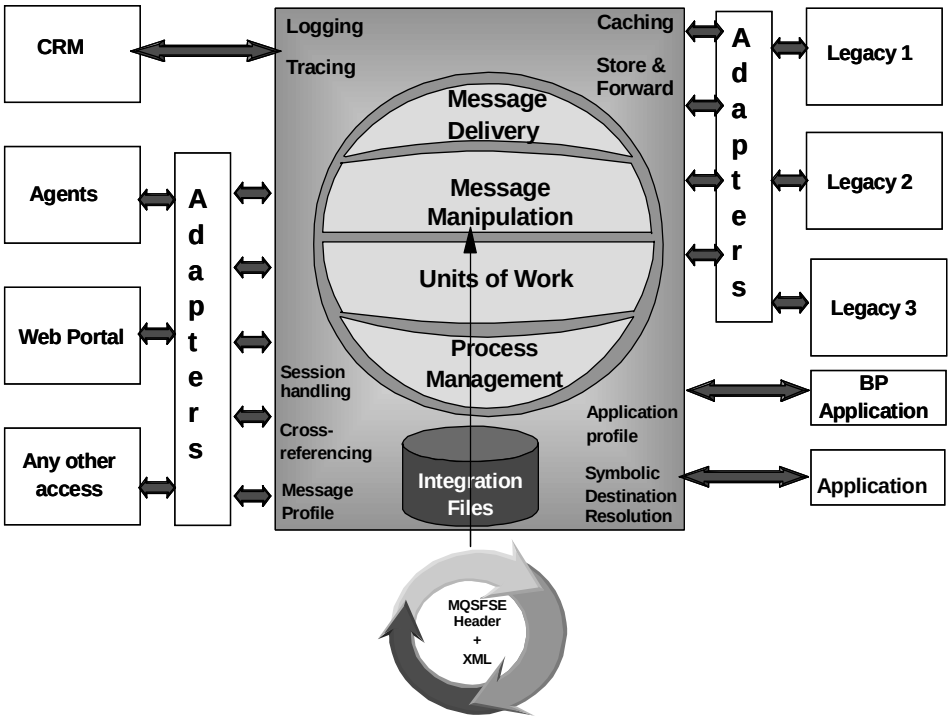

Figure 1: Intended WMQI Enabler use.

WMQI Enabler uses XML as the common semantic for integration, where adapters map application information data and commands to XML. WMQI Enabler also provides message routing, manipulation, application keys, cross-referencing, and logging services required to ensure delivery of the appropriate XML messages to the appropriate destination applications.

Chapter 2
Recommended operating environment

NOTE *System recommendations reflect the requirements needed to run the prerequisite MQSeries family of product and are based on WMQI Enabler development/testing.*

Recommended system requirements for the specified operating systems:

Windows NT® and Windows® 2000

1. Dual 500MHz Pentium III (or equivalent) processors.
2. 1 Gigabyte of RAM memory.
3. Windows NT 4.0 with ServicePak 6a.
4. 20 Gigabyte of free Hard drive space.
5. LAN capability with TCP/IP support using a fixed IP address.

AIX®

1. Dual 332MHz (or equivalent) processors.
2. 1 Gigabyte of RAM memory.
3. AIX® 4.3.3.
4. 20 Gigabyte of free Hard drive space.
5. LAN capability with TCP/IP support using a fixed IP address.

Solaris®

1. Dual 332MHz (or equivalent) processors.
2. 1 Gigabyte of RAM memory.
3. Solaris 2.8.
4. 20 Gigabyte of free Hard drive space.
5. LAN capability with TCP/IP support using a fixed IP address.

Prerequisite Windows® software

The WebSphere MQ Integrator Enabler (WMQI Enabler) software package was developed based on IBM's MQSeries family of software products. The following products must be installed prior to installing and using WMQI Enabler:

- **JDK 1.2.2** or higher
- **MQSeries 5.2**
  with the following prerequisite components:
  Active Directory Service Interface 2.0
  Microsoft Management Console 1.1
  Internet Explorer 5.5
  HTML help 1.2
- **WebSphere MQ Integrator 2.1 (WMQI)**
  with its prerequisite component:
  Microsoft Data Access Components v2.5
- **MQSeries WorkFlow 3.3.2**
- **IBM DB2 Universal Database 7.2 Enterprise Edition**

For a new installation, we recommend DB2 and the MQSeries family of products with their respective corrective services be installed before performing any configuration or customization on these products. Refer to each prerequisite product install manual for details on their installation.

Optional Windows® software

The following optional software can be used along with WMQI Enabler as desired:

- **MQSeries Support Pack MA0C** (Publish/Subscribe support)
  This Support pack can be used to help learn and understand more about the Publish/Subscribe concept, for distributing message data where it is wanted.
- **IBM Secure Way 3.2**
  IBM SecureWay product provides the interface to allow a user to create and use an LDAP (Lightweight Directory Access Protocol) table.

  NOTE *LDAP support capability is provided as is and limited to Windows NT systems only.*

- **MQSeries Integrator Support Pack IA08 LDAP 1.1 plug-in**
  This plug-in (for use with IBM Secure Way 3.2 and the HTTP Server) can be used in replacement of the WMQI Enabler SDR (Symbolic Destination Resolution) table. The plug-in is provided as is, without support or warranty from the IBM Corporation. An example usage of an LDAP table is provided.

- **IBM HTTP Server**
  The HTTP Server is required for use with IBM SecureWay product. It is also used by the product install process.

- **Adobe Acrobat Reader 4.05**
  The Reader tool can be used to view the WMQI Enabler publications.

NOTE *Refer to chapter 2 of the IBM Redbook,* ***Business Integration Solutions with MQSeries Integrator****, which provides a good "cookbook" approach on how to customize WMQI after installation.*

MQTester and configuration utility requirements

The sample, Windows based, MQTester and configuration utility applications provided with the WMQI Enabler Model Office requires JRE 1.2.2 or higher, and IBM's MQSeries Product Extension (Category 3) SupportPac MA88. MA88 provides MQSeries classes for Java and MQSeries classes for Java Message Service. It can be located on the web at:

http://www-4.ibm.com/software/ts/mqseries/txppacs/txpm2.html#cat3

Prerequisite AIX® / Solaris® software

The WebSphere MQ Integrator Enabler (WMQI Enabler) software package was developed based on IBM's MQSeries family of software products. The following products must be installed prior to installing and using WMQI Enabler:

- **MQSeries 5.2**

- **WebSphere MQ Integrator 2.1**
  with its prerequisite component:
  Java Runtime Environment v1.3

- **MQSeries WorkFlow 3.3.2**

- **IBM DB2 Universal Database 7.2 Enterprise Edition**

NOTE *For AIX® and Sun Solaris®, WebSphere MQ Integrator for Windows is also needed to support the Configuration Manager and the graphical Control Center from which to deploy message flows.*

To view, modify, or add to the sample MQSeries Workflow flows, the Workflow Buildtime component must also be installed on Windows.

For AIX® and Sun Solaris®, MQSeries for Windows is also needed to provide the interface between the Queue Managers, queues, and channels that are needed by WebSphere MQ Integrator and the supplied WMQI Enabler Test tool. Both WebSphere MQ Integrator and MQSeries Workflow rely on DB2 to provide their database support.

Chapter 3
Installing WMQI Enabler on Windows®

Prerequisite products explained

WMQI Enabler consists of customized WebSphere MQ Integrator (WMQI) message flows along with sample MQSeries Workflow flows that are used with the provided sample model Use Cases. For this reason, WMQI Enabler is very dependent upon IBM DB2 UDB to support data storage and to allow for the following functions:

MQSeries for message queuing.

WebSphere MQ Integrator for execution of the Message Flows.

MQSeries Workflow for execution of the sample Workflow flows.

For installation, you must manually install each of these products based on their respective installation manuals. Specific customization work needed for WMQI Enabler by each of these products is detailed in the *Installation procedure* section that actually describes the installation process for WMQI Enabler.

Prerequisite custom install notes

The following prerequisite product notes can be used to identify specific features used if they are not already installed within your existing systems environment.

- For IBM DB2 Universal Database 7.2 Enterprise Edition, install DB2 Enterprise Edition (optionally the DB2 Administration Client graphical tools), using the Typical installation as the minimum install options for WMQI Enabler.

- For MQSeries 5.2, WMQI Enabler needs the Server component as a minimum, including it's prerequisite components:
  Microsoft's ADSI
  MMC
  HTML help
  Acrobat Reader

- For WebSphere MQ Integrator 2.1, as a minimum, install the major components such as:
  Configuration Manager
  Broker
  User Name Server
  Control Center

- For MQSeries Workflow 3.3.2, under all components as a minimum, install the major components such as:
  Administration Utility
  Client
  Program Execution Agent
  Runtime Database Utilities
  Buildtime (component to view and/or alter the workflows)

  The standard default configuration for the Workflow Configuration Utility can be used by WMQI Enabler that defines its FMCQM queue manager and FMCDB database. For more information on the Workflow configuration, refer to ***MQSeries Workflow Installation Guide***, *Appendix E: Stand-alone setup on Windows NT/2000.*

  The sample WMQI Enabler TestSuites provided have their message profile records defined to use FMCQM as the queue manager and default local queues for queue message routing. If your environment does not already use the provided MQSeries Workflow default configuration, then the provided sample WMQI Enabler TestSuites that use Workflow will need to be altered, before executing them with the MQTester tool.

Installation

Before you begin, make certain you have:

- Windows Server 4.0 with all the necessary prerequisite software installed (refer to *Chapter 2, Specified Operating Environment*). If you are needing to upgrade software levels, we recommend that you reformat the hard drive and rebuild the system.

- A server with sufficient RAM (refer to *Chapter 2, Specified Operating Environment* to review the minimum hardware requirements).

NOTE *It is assumed that you are familiar with the MQSeries family of products and know how to install and configure each of them with the appropriate Queue Manager, Broker, Channels, and defined queues. The install process is not automated; files will need to be copied from WMQI Enabler and modified/executed as needed.*

Installation procedure

Once the prerequisite products are in place, to install WMQI Enabler on a stand-alone system, follow the procedure below using a user ID with Administrator rights. The same user ID can be used to execute all customization jobs.

Before you begin

Update the Userid, used by these installation steps, to authorize use for MQSeries and WebSphere MQ Integrator.

a. In Windows NT® under the **Start** button select **Programs**, **Administrative tools**, **User Manager**

b. Ensure the Userid used for customization or execution is a member of the following groups:

mqbrasgn

mqbrdevt

mqbrkrs

mqbrops

mqbrtpic

mqm

However, if the machine is a DOMAIN server, include:

DOMAIN mqbrasgn

DOMAIN mqbrdevt

DOMAIN mqbrkrs

DOMAIN mqbrops

DOMAIN mqbrtpic

DOMAIN mqm

Many provided command files are predefined with the user ID of **USERID** and password of **PASSWORD**. These may be modified to use some other valid user ID, if this ID isn't acceptable.

1. Create the WMQI Enabler DB2 database, catalog it, and register its Alias database names with ODBC .

FSE_CRF	(Alias for Cross Reference File table)
FSE_ERRL	(Alias for Error Log table)
FSE_MSGL	(Alias for Message Log table)

FSE_MSGP (Alias for Message Profile table)

FSE_SDR (Alias for Symbolic Destination Resolution table)

FSE_SESS (Alias for Session table)

FSE_STOF (Alias for Store Forward table)

FSE_SYSP (Alias for System Profile table)

FSE_TRAC (Alias for Trace table)

FSE_WFCO (Alias for Workflow Correlation table)

You can use the **Create_MQSFSE_Database** script file located in the **\NT** directory as input to the DB2 command processor to create this database and Alias names.

DB2 Command line example: **db2 < Create_MQSFSE_Database**

NOTE *This job also alters the Database configuration to allow for a larger number of application connections on the maxappls parameter.*

2. Create the required WMQI DB2 databases and catalog them to ODBC, if they don't already exist.

   Actual names for these databases is user defined with the default names shown below.

   MQSICMDB (ConFigMgr database)

   MQSIBKDB (Broker database)

   MQSIMRDB (Message Repository database)

   Refer to the file **MQSI_Databases** under directory **\NT** for a working example that creates these databases. Edit the files needed to resolve the directory location of the SQLLIB path.

   DB2 Command line example: **db2 < MQSI_Databases**

   NOTE *This job includes commands to alter the Database Manager for the number of database connections, to increase the App_Ctrl_Heap_sz and Applheapsz values, and to alter their Tablespace values*.

3. Create the WMQI Enabler database tables within the WMQI Enabler database and insert the required records.

   Using a DB2 Command line, process the script file **Create_MQSFSE_Tables** located under **\NT**.

   The inserted table records consist of the following:

MQSFSE Error codes.
Default Language record.
MQSFSE Message Profile records for the Hub commands.
Symbolic Destination Resolution record for MQSeries Workflow.
Authentication ID's for 3 specific ID's (USERID, userid, and SysAdmin).

DB2 Command line example: **db2 < Create_MQSFSE_Tables**

For more information about message profile, Symbolic Destination Resolution, WMQI Enabler commands, and so on, refer to the *Development Guide*, chapter 5.

4. Ensure a WMQI Queue Manager is defined and started.

   This can be accomplished within the MQSeries Explorer window by right clicking on the **Queue Managers** folder and selecting **New**, **Queue Manager**.

   The Queue Manager is required to have the following attributes:

   Defined as the (default, optional) Queue Manager; with any user specified name such as **MQSIQM**.

   Have a defined default Dead Letter queue as **SYSTEM.DEAD.LETTER.QUEUE**

   The **Log** file size set to **1024**.

   The number of **Log Primary** files set to at least **10**.

   The number of **Log Secondary** files set to at least **20**.

   Automatically start up after a reboot.

   Defined with **Server connection channel**.

   Defined with **Listener** configured for **TCP/IP**.

   Listener port set to **1414**
   (or some other port value if 1414 is unavailable).

   The Queue Manager will be used by the WebSphere MQ Integrator Broker.

   The Broker will be used to contain the deployed WMQI Enabler message flows (the Hub), identified in a later step. Sample testsuite test cases have the name MQSIQM defined as the Queue Manager and must be altered if this name is changed.

5. Ensure the WMQI **ConFigMgr**, **Broker**, and **UserNameServer** are defined and started.

   Refer to command files **CreateConfigMgr.cmd**, **CreateBroker.cmd** and **CreateUserNameServer.cmd**, located under **\NT** for working examples of how to define these services.

The Broker name (MQSIEBROKER) within the **CreateBroker.cmd** file can be modified to suit your naming conventions. This name is later used in the Topology section within the WMQI Control Center.

You may also want to set these services to start automatically after reboot by modifying the service definition for each, under the **Services** icon of the **Control Panel**.

NOTE *The **CreateConfigMgr** job issues a change request to increase the size of the Java Virtual machine to 1024 after it creates the ConfigMgr service. This was added to improve the success rate of the WMQI deploy process. Be sure to resolve the drive path and file location of the WMQI commands if they are used. Also, observe that the jobs use an assumed defined user ID of USERID to create these Services. This ID and password needs to be altered if another ID is needed to create these Services.*

6. Define the following WMQI Enabler local queues.

   HUB_IN

   HUB_IN_ERRORS

   HUB_IN_FAILURE

   HUB_ONLY_OFFLINE

   HUB_ONLY_OFFLINE_FAILURE

   HUB_ONLY_ONLINE

   HUB_ONLY_ONLINE_FAILURE

   HUB_R_IN

   HUB_R_IN_FAILURE

   HUB_RWF_IN

   HUB_RWF_IN_FAILURE

   MQWF_OUT

   MQWF_OUT_FAILURE

   MQWF_END

   MQWF_END_FAILURE

   MQWF_DEFAULT_ACTIVITY

   MQWF_DEFAULT_ACTIVITY_FAILURE

   ERROR_BUCKET

   You can use the **queues.def** script file, located under the **\NT** directory, as input to the MQSeries **runmqsc** program.

   Command line example: **runmqsc < queues.def**

These queues will be used under the WMQI Queue Manager, as referenced in step 4.

NOTE The **queues.def** file actually contains a larger set of local queue definitions than shown above. The additional queues are used by MQTester and sample MQAO adapters. The list shown above consist of the actual queue names used specifically by WMQI Enabler in the product Message Flows.

7. Ensure the MQSeries Workflow Configuration utility has been used to configure the components noted below. Typically, this was run during Workflow installation, using its Default Workflow configuration with its default names for the Queue Manager (FMCQM), DB2 instance, system, database names, etc.

   For an existing installation, the Workflow component names may be named differently. If this is the case, the sample TestSuites may need to be altered to reflect the component name changes. Regardless, the WMQI Enabler provided sample Workflow flows can use either the Default configuration or a user modified configuration.

   Server

   Runtime database utilities

   Buildtime

   Client

   8.i. Create 2 sets of channels under MQSeries Explorer that can be used between the WMQI Queue Manager and the MQSeries Workflow Queue Manager; if they don't already exist.

   Refer to files **mqsiqmcl.mqs** and **fmcqmcl.mqs**, under **\NT**, for working examples of these channel definitions along with their needed transmission queues.

   Make sure to alter the **CONNAME** field with the *local hostname* of the machine you are installing on before running these jobs.

   Assuming the Queue Managers are started (using the strmqm command or the graphical Explorer window), you can use the **runmqsc** program to process these two files, if needed.

   Command line example: **runmqsc MQSIQM < mqsiqmcl.mqs**

   Command line example: **runmqsc FMCQM < fmcqmcl.mqs**

   ii. Issue this command to start the following channel:

   Type: **runmqsc MQSIQM**

   Type: **START CHANNEL(MQSIQM.TO.FMCQM)**

   Type: **end**

iii. Issue this command to start the following channel:

Type: **runmqsc FMCQM**

Type: **START CHANNEL(FMCQM.TO.MQSIQM)**

Type: **end**

9. From the WMQI Control Center create the WMQI Enabler Broker (**MQSIEBROKER**), under the **Topology** tab.
   i. Right click on the **Topology** term in the Topology tab selecting **Check out** from the pop-up menu.
   ii. Right click on the **Topology** term again, selecting **Create**, **Create Broker** from the drop-down box.
   iii. In the Create Window specify the name of the Broker to **MQSIEBROKER** to match the Broker name defined in step 5.
   iv. In the same Create Window specify the Queue Manager name as **MQSIQM** to match the Queue Manager name defined in step 4.

   This Broker name can be altered, as needed, to conform to your system definition. If this is your first time loading the WMQI Control Center application, you will be prompted in the Configuration Manager Connection window to supply, your machine's Host Name, the port number used for its Queue Manager (1414), and the Queue Manager name (MQSIQM).
10. From the WMQI Control Center, import the WMQI Enabler message flows.

    Import the **MQSFSEMessageFlowExport.xml** file under the **\NT** directory.
    i. From the WMQI Control Center **File** menu, select **Import to Workspace**.
    ii. Check both **Topics** and **Message Flows** check boxes.
    iii. Click **Import** to import the XML file.
    iv. Click on the **File** menu and select **Check In**, **All (Save to Shared)** to save the import workspace.

    If you are an existing user of WMQI, the WMQI Enabler XML file will add additional Flows and Topics to your system.
11. From the WMQI Control Center, create the MQSIEBROKER configuration under the **Assignments** tab and perform the deploy process.
    i. Expand the **+** icon for Message Flows, in the 'Assignable Resources Center' window.
    ii. Locate the **HUB_IN_Fow** and drag it into the new broker's **default** Execution group.
    iii. Check in this configuration by, right clicking on the **default** Execution group and selecting **Check In** from the drop-down menu.
    iv. From the **File** menu select **Deploy**, **Complete Configuration, Normal**.

v. Click **OK** on the Confirmation screen.

   The deploy process can take several minutes to complete. It may be helpful to start the Windows Task Manager, selecting the **Performance** tab and observe the utilization rate of the processor(s) to detect when the deploy process has completed. The deploy process will only use one processor if more than one are installed.

vi. Click on the **Log** tab, check to see if the deploy was successful.

The success of a deploy process depends on several factors, many of which are hardware or configuration related.

Once the deploy of **HUB_IN_Flow** has successfully completed, the following main message flows must be deployed.

MQSFSE_HUB_ONLY_OFFLINE_Flow

MQSFSE_HUB_ONLY_ONLINE_Flow

MQSFSE_HUB_R_IN_Flow

MQSFSE_HUB_RWF_IN_Flow

MQSFSE_MQWF_END_Flow

MQSFSE_MQWF_DEFAULT_ACTIVITY_Flow

MQSFSE_MQWF_OUT_Flow

MQSFSE_StoreMessageTemplate_Flow

An illustration of the recommended configuration is shown in the following figure:

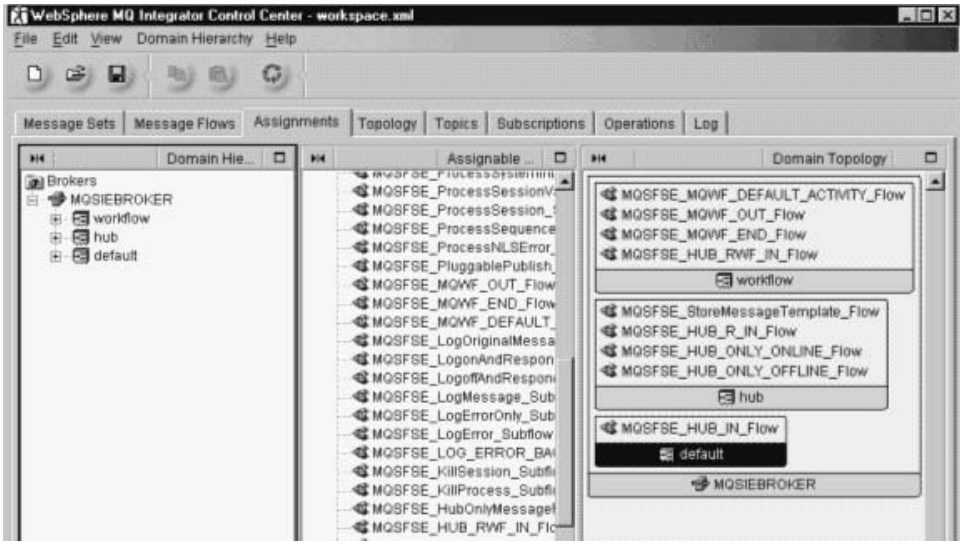

Figure 2: WMQI configuration example.

The following steps will instruct you on completing the deployment of all remaining flows. These instructions will be performed on windows in the Assignments tab. Refer to figure 2 for the recommended configuration

i. Check out the broker **MQSIEBROKER** by right-clicking on the broker **MQSIEBROKER** name and selecting **Check Out**.

ii. Create a new Execution group by right clicking on the broker **MQSIEBROKER** and selecting **Create Execution Group**. Refer to figure 2 for the recommended execution group configuration names.

iii. Drag one of the designated flows into the **Execution Group**

iv. Check in the **Execution Group** (and the broker **MQSIEBROKER** if necessary).

v. Right click on the **Execution Group** and select **Deploy**, **Delta Assignments Configuration**.

vi. Repeat step iii above until all flows for the execution group have been deployed. The execution group will need to be checked out in order to add a flow. Repeat step “i” above to build the remaining Execution group.

12. Optionally, create the MQSeries Workflow Runtime Client Lists.

The following are needed for debugging or for viewing any non-automated workflows.

Process Template List

Process Instance List

Work List

i. For a new user, load the MQSeries Workflow Client window and sign on with the predefined Workflow ID of **ADMIN** (with password = password).

ii. Fill in the userid and password fields and click on **OK**.

If the signon was successful, a tree view window should be displayed.

iii. Within this window, right click on **Process Template Lists** and fill in the name for the Process Template, such as, ProcessTemplateList, then click on **OK**.

iv. Perform this same task for **Process Instance Lists** and **Worklists**, providing just a name and clicking **OK**.

v. Once this is complete, simply close the Workflow Client window, answering **YES** when prompted to exit.

13. Under MQSeries Workflow Buildtime, import the sample Workflow definition file for use by the sample Use Cases.

i. Locate the **MQSFSEWorkFlowProcesses.fdl** file under **\NT**.

ii. From the **BUILDTIME** menu item, select **Import**.

iii. Enter the file name of **MQSFSEWorkflowProcesses.fdl**.

iv. Select the **Overwrite Import** flag only.

v. Do NOT check the **FDL from MQSeries Workflow Runtime** option under the Import flag.

vi. Click on **OK** to import these sample workflows.

14. Once imported to Buildtime, ensure the Userid to be used under Workflow Client is defined based on your system userid definitions.

This userid must be a valid Windows user for MQSeries Workflow to successfully process the workflows.

The imported Workflow processes from step 13 adds the ID **USERID** to the existing Workflow ID of **ADMIN**.

USERID parameters can be viewed to create a new user ID, if required, by clicking on the **STAFF** tab and expanding the Persons tree structure.

NOTE *the sample test suites provided with WMQI Enabler are set up to use the ID, USERID within Workflow and the test Use Case messages. The sample test suites would need to be altered if another user ID is used. Also, if another userid is used, this userid must be added to the WMQI Enabler database's Authentication Table, similar in definition to the USERID provided.*

15. From the **BUILDTIME** Menu item, select **Export**, keeping all selections, plus checking the export flag, **export deep** and export the complete Buildtime environment into a user defined file.

    When the configuration is exported, close the Buildtime program.

16. From a command line, execute the MQSeries WorkFlow program **fmcibie** to import the previous steps MQSeries WorkFlow Buildtime exported **.fdl** file into MQSeries Workflow Runtime.

    The syntax is:

    fmcibie -u ADMIN -p password -iC:\mqworkflowtemp\userBuild-time.fdl -o -t -f

    Where:

    ADMIN and **password** are workflow's default defined authorized system ID and password.

    C:\mqworkflowtemp is the drive and path of the exported file from step 15.

    UserBuild-time.fdl is the name of the file exported.

WMQI Enabler installation is now complete and ready for testing. Refer to *Chapter 6 Testing WMQI Enabler Installation* to verify the Installation.

Chapter 4
Installing WMQI Enabler on AIX®

Before you begin, make certain you have:

- AIX® V4.3.3 with all the necessary prerequisite software installed (refer to Chapter 2, *Specified Operating Environment*).
- All necessary hardware, including sufficient RAM (refer to Chapter 2, *Specified Operating Environment*).
- The WMQI Enabler.
- Read the **Readme.txt** file on WMQI Enabler for last minute fixes/updates.

NOTE *It is assumed that you are familiar with the DB2 and MQSeries products and how to customize each of them such that you have an active system with the appropriate Queue Managers, Broker, channels, and queue's defined or how to create them as needed.*

It is recommended that you install DB2, MQSeries, WebSphere MQ Integrator, MQSeries Workflow, plus all their respective Corrective Service Disks before performing any customization activity with any of these products. Follow the installation procedures described in the product specific Installation or Quick Beginning Manuals for step-by-step instructions.

WMQI Enabler consists of customized WebSphere MQ Integrator (WMQI) message flows along with sample MQSeries Workflow flows that are used with the provided sample model use cases. Because of this, WMQI Enabler is very dependent upon IBM DB2 UDB to support data storage, MQSeries for message queuing, WebSphere MQ Integrator for execution of the Message Flows, and MQSeries Workflow for execution of the sample Workflow flows.

You must manually install each of these products based on their respective installation manual. Specific customization work needed for WMQI Enabler by each of these products is detailed in the numbered steps shown below that describe the installation process for WMQI Enabler.

Many of the provided jobs use the user id of “userid” (all lowercase) within them. You must alter this id to match your system requirements.

Prerequisite install notes

The following prerequisite product notes can be used to identify specific features used if they are not already installed within your existing system environment.

- For IBM DB2 UDB 7.2 Enterprise Edition, run the **db2setup** file and install DB2 Enterprise Edition, server, client, administration, and communication support, as a minimum set of install options for WMQI Enabler. The default instance and administrator name are sufficient.

- For MQSeries 5.2, setup the recommended file systems and install the Server, client, java, and base components as a minimum. On Windows NT®, install the Server component.

- For WebSphere MQ Integrator 2.1, install the prerequisite JDK, create the recommended file systems, install the java.rte, base, and samples components.
  On Windows, install the Control Center and Configuration Manager support.
  On AIX®, be sure to run the **mqsi_setupdatabase** command to identify DB2 as the database support and to access the sample **profile.aix** file to use the ODBCINI environment variable,.

- For MQSeries Workflow 3.3.2, install the server, client, and fmc.default configuration as a minimum. The default Workflow configuration can be used by WMQI Enabler. The default configuration defines its **FMCQM** Queue Manager and **FMCDB** database which can be created during the MQSeries Workflow installation.
  On Windows, install the MQSeries Workflow product and run the default Configuration Utility to create the Buildtime environment with its database for later manipulation of the Workflow processses.
  On AIX, before the execution of the Workflow Configuration Utility, be sure to run the DB2 profile (**db2profile)** under userid **root** to allow the configuration to successfully resolve the DB2 instance name.

  The sample WMQI Enabler test suites provided have their message profile records defined to use FMCQM as the Queue manager and default local queues for queue message routing. Ensure the Workflow configuration is correctly defined by running the **fmczchk** utility on config id FMC (assuming you use the default values).

  For more information on this Workflow configuration, refer to ***MQSeries Workflow Installation Guide***.

  If your environment does not already use the provided MQSeries Workflow default configuration, then the provided sample WMQI Enabler TestSuites

that use Workflow will need alteration, if you want to execute them with the MQTester tool. Typical Workflow updates needed to user **fmc** profile file, include the following, before running the **fmczchk** utility:

```
DB2_RR_TO_RS=YES
LANG=en_US
Export DB2_RR_TO_RS
Export LANG
```

Many of the provided jobs use the userid of **userid** within them. You will need to alter this id to match your system needs.

Once the prerequisite software is installed, the WMQI Enabler product can be added to the AIX® System (along with Windows components), using the following procedure:

Installation procedure

1. Create the WMQI Enabler DB2 database , using the user **db2inst1**.

   Before this is accomplished, we need to ensure WMQI can access the WMQI Enabler DB2 database using TCP/IP due to the AIX® limit of accessing at most 10 shared memory segments.

   This can be done using the following procedures:

   i. Ensure the DB2COMM variable is set to use 'tcpip' by issuing the command **db2set -all DB2COMM**.
   This would already be set under a default installation.

   ii. Update the DB2 Database Manager configuration to use the services specified in **/etc/services** for the connection port to the DB2 instance.

   Issue the following command:
   db2 update dbm cfg using svcename db2cdb2inst1
   This would already be set under a default installation.

   iii. Using a DB2 authorized user, create the WMQI Enabler database, bind it, and catalog the database with a set of alias names for ODBC and catalog a local node to resolve the TCP/IP connection.

   You can use the **MQSFSEDB** file located under /unix directory in the MQSFSE.tar file as input to the DB2 command processor to create the database and alias names.

   Be sure to edit the localhost name to match the *localhost* name of the machine that this command is executed on.

   DB2 Command line example: **db2 < MQSFSEDB**

This job creates the database along with its alias names and catalogs them as shown below:

FSE_CRF	**(alias name for Cross Reference File table)**
FSE_ERRL	**(alias name for Error Log table)**
FSE_MSGL	**(alias name for Message Log table)**
FSE_MSGP	**(alias name for Message Profile table)**
FSE_SDR	**(alias name for Symbolic Destination Resolution table)**
FSE_SESS	**(alias name for Session table)**
FSE_STOF	**(alias name for Store Forward table)**
FSE_SYSP	**(alias name for System Profile table)**
FSE_TRAC	**(alias name for Trace table)**
FSE_WFCO	**(alias name for Workflow Correlation table)**

iv. Be sure to add the database alias names of the database to the **/var/mqsi/odbc/.odbc.ini** file for WMQI.

This will allow WMQI to resolve TCP/IP access to the databases which otherwise would be limited to 10 connections under normal DB2 processing for AIX®.

Since the WMQI Enabler database name must be unique from the alias names, you must ensure the actual database name is different from the alias names within the **.odbc.ini** file.

See the following as a partial **.odbc.ini** file for reference:

```
[ODBC Data Sources]
MQSIBKDB=IBM DB2 ODBC Driver
FSE_CRF=IBM DB2 ODBC Driver
FSE_ERRL=IBM DB2 ODBC Driver

.
.

[MQSIBKDB]
Driver=/u/db2inst1/sqllib/lib/db2.o
Description=MQSIBKDB DB2 ODBC Database
Database=MQSIBKDB

[FSE_CRF]
Driver=/u/db2inst1/sqllib/lib/db2.o
Description=FSE_CRF DB2 ODBC Database
Database=FSE_CRF

[FSE_ERRL]
Driver=/u/db2inst1/sqllib/lib/db2.o
Description=FSE_ERRL DB2 ODBC Database
Database=FSE_ERRL

.
.
```

The **.odbc.ini** database names are the alias names and the real database name is defined as a different name.

2. Create the required WMQI DB2 databases and register each Windows-based database with ODBC, if they don't already exist:

   MQSICMDB (ConFigMgr database on Windows)

   MQSIBKDB (Broker database AIX®)

   MQSIMRDB (Message Repository database on Windows)

   i. Locate file BrokerDB under the /unix directory in the MQSFSE.tar file for a working job of how to create the WMQI Broker database. Edit the file as needed to resolve the directory location of the SQLLIB path.

      DB2 Command line example: **db2 < BrokerDB**

   ii. Locate file MQSIDBNT under /unix directory in the MQSFSE.tar file for an example of how to create these databases for Windows.

      Copy this file to your Windows system and execute it using a valid, authorized DB2 userid. Edit the file to set the location of the SQLLIB directory for DB2.

      DB2 Command line example: **db2 < MQSIDBNT**

Keep in mind the two files are designed to create and catalog the two databases on Windows and the Broker database on AIX®. You may need to adjust these files to run on your system and to conform to your naming conventions.

NOTE *They alter the Database Manager Configuration and for the number of database connections and to increase the App_Ctrl_Heap_sz and Applheapsz values, and to alter their Tablespace values.*

3. Update the **User Id** that you are using to perform WebSphere MQ Integrator customization and DB2 calls.

   This user can be used to perform the remaining steps, unless otherwise noted.

   Under smitty, Security & Users, ensure the Userid is a member of the **db2iadm1**, **db2asgrp**, **mqbrkrs**, and **mqm** groups if you didn't do this after installing the MQSeries of products.

4. Using the updated user id from step 3, execute the script file **DBTables**, located under **/unix** directory in the MQSFSE.tar file, to create the WMQI Enabler database tables and to insert required records.

   The inserted table records consist of the following:

   MQSFSE Error codes.

   Default Language record.

   MQSFSE Message Profile records for the Hub commands.

   Symbolic Destination Resolution record for MQSeries Workflow.

   Authentication ID's for 3 specific ID's (USERID, userid, and SysAdmin).

   Command line example: **db2 < DBTables**

   For more information about message profile, Symbolic Destination Resolution, WMQI Enabler commands, etc., refer to the ***Development Guid***e, *Chapter* 5.

5. Ensure a MQSeries Queue Manager is defined and started on both AIX® (for WMQI Broker) and Windows (for the WMQI Configuration Manager).

   On AIX® and Windows Queue Managers are required to have the following attributes:

   Defined as the (default, optional) Queue Manager (with a user defined name such as **MQSIQM** for AIX® and **MQSIQMCM** for Windows). Other provided jobs and the sample test cases, reference these Queue Manager names and must be altered if other Queue Manager names are defined.

For AIX®, the file **crtq.cmd**, located in the /unix directory, provides a working example of how to create this Queue Manager and for Windows, the MQSeries Explorer graphical window can be used to create its Queue Manager.

Have a defined default Dead Letter queue as **SYSTEM.DEAD.LETTER.QUEUE**

The **Log** file size set to **1024**.

The number of **Log Primary** files be set to at least **10**.

The number of **Log Secondary** files be set to at least **20**.

Defined with **Listener** configured for **TCP/IP**.

Listener port set to **1414,** if available (for both Queue Managers).

On AIX, to start the Queue Manager and its Command Server, type the following at the command line: **strmqm MQSIQM** and **strmqcsv MQSIQM**

On Windows, use the MQSeries Explorer window to start the MQSIQMCM Queue Manager if it was not started when it was created.

6. Ensure the WMQI **ConFigMgr** (on Windows), **Broker**, and **UserNameServer** (AIX®) are defined and started.
   i. Refer to command file **CreateConfigMgr.cmd** located under **/unix** for a working example of how to define the Configuration manager on Windows.

      Note: If the Configuration Manager's Queue Manager name suggested in step 5 is not used (MQSIQMCM), be sure to alter the file to use your name instead. Be sure the Configuration Manager is started under Services within the Windows Control Panel.

   ii. Refer to files, **Broker** and **UserName**, located under **/unix** directory in the MQSFSE.tar file for working examples to create the **MQSIEBROKER** Broker and **UserNameServer** on AIX.

      This broker name, **MQSIEBROKER**, can be changed to suit your naming conventions. To start the Broker and UserNameServer on unix, issue **mqsistart MQSIEBROKER** and **mqsistart UserNameServer**

7. Define the WMQI Enabler local Queues:

   HUB_IN

   HUB_IN_ERRORS

   HUB_IN_FAILURE

   HUB_ONLY_OFFLINE

   HUB_ONLY_OFFLINE_FAILURE

   HUB_ONLY_ONLINE

   HUB_ONLY_ONLINE_FAILURE

HUB_R_IN

HUB_R_IN_FAILURE

HUB_RWF_IN

HUB_RWF_IN_FAILURE

MQWF_OUT

MQWF_OUT_FAILURE

MQWF_END

MQWF_END_FAILURE

MQWF_DEFAULT_ACTIVITY

MQWF_DEFAULT_ACTIVITY_FAILURE

ERROR_BUCKET

i. You can use the queuesunix.def script file located under the **/unix** directory in the MQSFSE.tar file, as input to the MQSeries **runmqsc** program.

Command line example: **runmqsc MQSIQM < queuesunix.def**

These queues will be used under the AIX® based WMQI Queue Manager, as referenced in step 5.

NOTE *The **queuesunix.def** file contains a larger set of local and remote queue definitions than shown above. The additional queues are used by MQTester and sample MQAO adapters. The list shown above consist of the actual queue names used specifically by WMQI Enabler in the product Message Flows.*

ii. To define the needed queues on Windows (used by MQTester) that correspond to the queues with AIX, use the queuesNT.def file located under the **/unix** directory in the MQSFSE.tar file, as input to the MQSeries **runmqsc** program on Windows.

Command line example: **runmqsc MQSIQMCM< queuesNT.def**

8. i. Creates 2 sets of channels under MQSeries that can be used between the WMQI Queue Manager and the MQSeries Workflow Queue Manager.

Refer to the files **mqsiqmcl.mqs** and **fmcqmcl.mqs**, under **/unix** directory in the MQSFSE.tar file, for working examples of these channel definitions along with their needed transmission queues.

This assumes the Workflow configuration utility (fmczutil) was used to create a workflow configuration to configure the server, runtime database, and client components. Typically this was run during MQSeries Workflow installation, using the default configuration names for its Queue Manager

FMC QM, database name, and system settings. If the default Queue Manager name was changed then these jobs need to reflect this name change as well.

Be sure to alter the **CONNAME** field with the *local hostname* of the machine before running these jobs.

Command line example: **runmqsc MQSIQM < mqsiqmcl.mqs**

Command line example: **runmqsc FMCQM < fmcqmcl.mqs**

ii. Issue this command on AIX® to start the following channel:

Type: **runmqsc MQSIQM**

Type: **START CHANNEL(MQSIQM.TO.FMCQM)**

Type: **end**

iii. Issue this command on AIX® to start the following channel:

Type: **runmqsc FMCQM**

Type: **START CHANNEL(FMCQM.TO.MQSIQM)**

Type: **end**

9. i. Create 2 sets of channels for the WMQI Queue Managers between AIX® and Windows and start them.

You can use files **mqsiunix.mqs** (run on AIX® system) and **mqsint.mqs**, (run on Windows system) located under directory **/unix** directory in the MQSFSE.tar file, as working examples for this set of channel definitions and their respective transmission queue.

These files assume certain Queue Manager names and must be modified if different names exist from those specified in step 5.

Be sure to alter the **CONNAME** field to match the correct *remote hostname* or *IP address* before running these jobs.

Command line (unix) example: **runmqsc MQSIQM < mqsiunix.mqs**
Command line (Windows) example: **runmqsc MQSIQMCM< mqsint.mqs**

ii. Issue this command on AIX® to start the following channel:
Type: **runmqsc MQSIQM**

Type: **START CHANNEL(MQSIQM.TO.MQSIQMCM)**

Type: **end**

iii. Issue this command on Windows to start the following channel:
Type: **runmqsc MQSIQMCM**

Type: **START CHANNEL(MQSIQMCM.TO.MQSIQM)**

Type: **end**

Optionally, it may be helpful to observe and directly manipulate the Queue Manager, queues, and channels from the Windows MQSeries Explorer window. To do this perform the following:

i. On AIX® issue the command: **strmqcsv MQSIQM**

ii. On AIX® issue the command: **runmqlsr -t tcp -p 1414 -m MQSIQM**

iii. On AIX® issue the command: **runmqsc MQSIQM**

Type: **DEFINE CHANNEL(SYSTEM.ADMIN.SVRCONN) CHLTYPE(SVRCONN)**

Type: **START CHANNEL(SYSTEM.ADMIN.SVRCONN)**

Type: **END**

iv. On Windows under the MQSeries Explorer window right click on the **Queue Managers** term selecting **Show Queue Manager** from the drop down box.

Select **Show a remote queue manager** radio button.

Enter remote Queue Manager name: **MQSIQM**.

Enter the connection name: **remotehostname(1414)**, where remotehostname is the IP address or hostname of the AIX system.

10. From the Windows WMQI Control Center create the WMQI Enabler Broker (**MQSIEBROKER**), under the **Topology** tab.

i. Right click on the **Topology** term in the Topology tab selecting **Check out** from the pop-up menu.

ii. Right click on the **Topology** term again, selecting **Create**, **Create Broker** from the drop-down box.

iii. In the Create Window specify the name of the Broker to **MQSIEBROKER** to match the Broker name defined in step 6.

iv. In the same Create Window specify the AIX Queue Manager name as **MQSIQM** to match the Queue Manager name defined in step 5.

This Broker name can be altered, as needed, to conform to your system definition. If this is your first time loading the WMQI Control Center application, you will be prompted in the Configuration Manager Connection window to supply your machine's host name, the port number used for its Queue Manager (1414), and the Queue Manager name (MQSIQMCM).

11. From the Windows WMQI Control Center, import the WMQI Enabler message flows. Import the **MQSFSEMessageFlowExport.xml** file under the **\NT** directory.

i. From the WMQI Control Center File menu, select **Import to Workspace**.

ii. Check both **Topics** and **Message Flows** check boxes.

iii. Click **Import** to import the XML file.

iv. Click on the **File** menu and select **Check In**, **All (Save to Shared)** to save the import workspace.

If you are an existing user of WMQI, the WMQI Enabler XML file will add additional Flows and Topics to your system.

12. From the Windows NT® WMQI Control Center, create the MQSIEBROKER configuration under the **Assignments** tab and perform the deploy process to send the configuration to the Broker on AIX®.

    i. Expand the **+** icon for Message Flows, in the Assignable Resources center window.

    ii. Locate the **HUB_IN_Flow** and drag it into the new broker's **default** execution group.

    iii. Check in this configuration by, right clicking on the **default** Execution group and selecting **Check In** from the drop-down menu

    iv. From the **File** menu select **Deploy**, **Complete Configuration, Normal**.

    v. Click **OK** on the Confirmation screen.

    The deploy process can take several minutes to complete. It may be helpful to start the Windows Task Manager, selecting the Performance tab and observe the utilization rate of the processor(s) to detect when the deploy process has completed. The deploy process will only use one processor if more than one are installed.

    vi. Click on the **Log** tab, check to see if the deploy was successful.

    The success of a deploy process depends on several factors, many of which are hardware or configuration related.

    Once the deploy of **HUB_IN_Flow** has successfully completed, the following main message flows must be deployed.

    MQSFSE_HUB_ONLY_OFFLINE_Flow

    MQSFSE_HUB_ONLY_ONLINE_Flow

    MQSFSE_HUB_R_IN_Flow

    MQSFSE_HUB_RWF_IN_Flow

    MQSFSE_MQWF_END_Flow

    MQSFSE_MQWF_DEFAULT_ACTIVITY_Flow

    MQSFSE_MQWF_OUT_Flow

    MQSFSE_StoreMessageTemplate_Flow

An illustration of the recommended configuration is shown in the figure below:

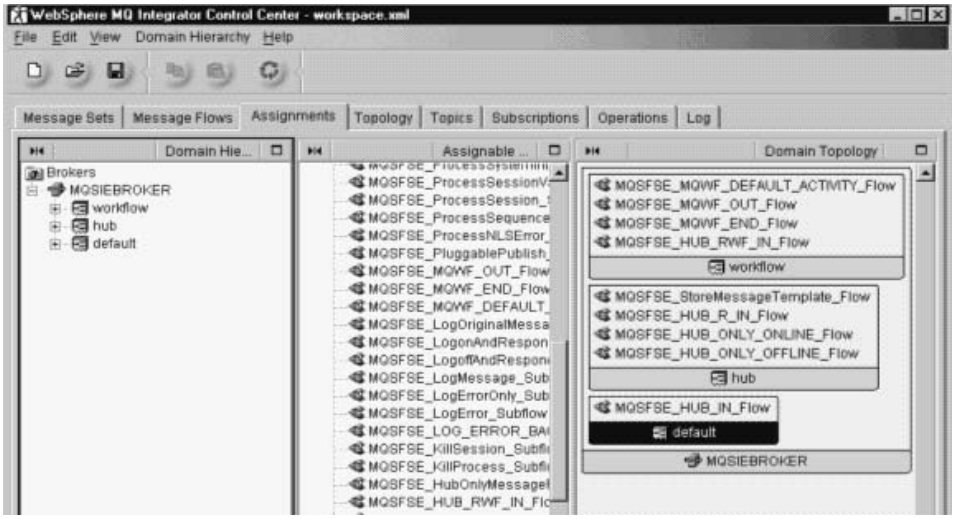

Figure 3: WMQI configuration example.

The following steps will instruct you on completing the deployment of all remaining flows. These instructions will be performed on windows in the Assignments tab. Refer to the figure above for the recommended configuration

i. Check out the broker **MQSIEBROKER** by right-clicking on the broker MQSIEBROKER name and selecting **Check Out**.

ii. Create a new Execution group by right clicking on the broker **MQSIEBROKER** and selecting **Create Execution Group**. Refer to the figure above for the recommended execution group configuration names.

iii. Drag one of the designated flows into the **Execution Group**

iv. Check in the **Execution Group** (and the broker **MQSIEBROKER** if necessary).

v. Right click on the **Execution Group** and select **Deploy**, **Delta Assignments Configuration**.

vi. Continue with step iii above until all flows for the execution group have been deployed. The execution group will need to be checked out in order to add a flow. Repeat the steps above to build the remaining Execution group.

13. Under MQSeries Workflow Buildtime (on Windows), import the sample Workflow definition file for use with the use cases.

i. Locate file **MQSFSEWorkFlowProcesses.fdl** in directory **\NT**.

ii. From the Builtime menu, select **Import** to import the workflow flows.

iii. Check the **Overwrite** option under Import Flags.

iv. Do NOT check the **FDL from MQSeries Workflow Runtime** option under Import Flags.

14. Once imported to Buildtime, ensure the Userid to be used under MQSeries Workflow Client on AIX® is defined. This userid must be a valid AIX® user for MQSeries Workflow to successfully process the flows.

    The imported Workflow processes from step 13 adds the ID **USERID** (with password of **PASSWORD**) to the existing Workflow ID of **ADMIN**. MQSeries Workflow displays the ID's in upper case only and a lower case name of userid will work but the password may need to be altered to match a new value such as **password**.

    USERID parameters can be viewed to create a new user ID, if required, by clicking on the **STAFF** tab and expanding the Persons tree structure.

    NOTE *the sample test suites provided with WMQI Enabler are set up to use the ID, USERID within Workflow and the test Use Case messages. The sample test suites would need to be altered if another user ID is used. Also, if another userid is used, this userid must be added to the WMQI Enabler database's Authentication Table, similar in definition to the USERID provided.*

15. On Windows, export the Buildtime environment as a user defined file and copy it to the AIX® system within the **/var/mqm** directory (if using the default file directories).

    i. From the **Buildtime** menu item, select **Export**, keeping all selections, plus checking the export flag, **export deep**.

    ii. When the configuration is exported, close the Buildtime program. Copy the userBuild-time.fdl file to AIX under /var/mqm.

16. On AIX, become user **fmc** by issuing the following command: **su - fmc**

17. On AIX®, using an authorized **fmcgrp** user, execute the MQSeries Workflow program **fmcibie** to import the previous steps MQSeries Workflow Buildtime exported **.fdl** file. The syntax is:

    fmcibie -u ADMIN -p password -i/var/mqm/userBuild-time.fdl -o -t -f

    Where:

    ADMIN and **password** are your previously defined authorized system ID and password.

    var/mqm is the directory path of the exported file.

    UserBuild-time.fdl is replaced with the name of the file exported.

WMQI Enabler customization is now complete and ready for testing. Be sure to Start the following if they weren't already started (This is especially important after a reboot):

Queue Managers

Sender Channels

Broker

UserNameServer

MQSeries Workflow

Refer to *Chapter 6 Testing WMQI Enabler Installation* to verify the Installation.

Chapter 5
Installing WMQI Enabler on Solaris®

Before you begin, make certain you have:

- Solaris® V2.8 with all the necessary prerequisite software installed (refer to Chapter 2, *Specified Operating Environment*).
- All necessary hardware, including sufficient RAM (refer to Chapter 2, *Specified Operating Environment*).
- WMQI Enabler.
- Read the **Readme.txt** file on WMQI Enabler for last minute fixes/updates.

NOTE *It is assumed that you are familiar with the DB2 and MQSeries products and how to customize each of them such that you have an active system with the appropriate Queue Managers, Broker, channels, and queue's defined or how to create them as needed.*

It is recommended that you install DB2, MQSeries, WebSphere MQ Integrator, MQSeries Workflow, plus all their respective Corrective Service Disks before performing any customization activity with any of these products. Follow the installation procedures described in the product specific Installation or Quick Beginning Manuals for step-by-step instructions.

WMQI Enabler consists of customized WebSphere MQ Integrator message flows along with sample MQSeries Workflow flows that are used with the provided sample model use cases. Because of this, WMQI Enabler is very dependent upon IBM DB2 UDB to support data storage, MQSeries for message queuing, WebSphere MQ Integrator for execution of the Message Flows, and MQSeries Workflow for execution of the sample Workflow flows.

You must manually install each of these products based on their respective installation manual. Specific customization work needed for WMQI Enabler by each of these products is detailed in the numbered steps shown below that describe the installation process for WMQI Enabler.

Many of the provided jobs use the user id of **userid** (all lowercase) within them. You will need to alter this id to match your system needs.

Prerequisite install notes

The following prerequisite product notes can be used to identify specific features used if they are not already installed within your existing system environment.

- For IBM DB2 UDB 7.2 Enterprise Edition, run the **db2setup** file and install DB2 Enterprise Edition, server, client, administration, and communication support, as a minimum set of install options for WMQI Enabler. The default instance and administrator name are sufficient.

- For MQSeries 5.2, setup the recommended file systems and install the Server, client, java, and base components as a minimum. On Windows, install the Server component.

  During installation, do not select **DCE Support** when prompted. Otherwise, segmentation faults could occur during creation of Queue Managers.

- For WebSphere MQ Integrator 2.1, install the prerequisite JDK, create the recommended file systems, install the java.rte, base, and samples components.
  On Windows, install the Control Center and Configuration Manager support.
  On Solaris®, be sure to run the **mqsi_setupdatabase** command to identify DB2 as the database support and to access the sample **profile.aix** file to use the ODBCINI environment variable,.

- For MQSeries Workflow 3.3.2, install the server, client, and fmc.default configuration as a minimum. The default Workflow configuration can be used by WMQI Enabler. The default configuration defines its **FMCQM** Queue Manager and **FMCDB** database.

  On Windows, install the MQSeries Workflow product and run the default Configuration Utility to create the Buildtime environment with its database for later manipulation of the Workflow processses.

  On Solaris®, just before the execution of the Workflow Configuration Utility, be sure to run the DB2 **profile** under userid **root** to allow the configuration to successfully resolve the DB2 instance name.

  The sample WMQI Enabler test suites provided have their message profile records defined to use FMCQM as the Queue manager and default local queues for queue message routing. Ensure the Workflow configuration is correctly defined by running the **fmczchk** utility on config id FMC (assuming you use the default values).

  For more information on this Workflow configuration, refer to ***MQSeries Workflow Installation Guide***.
  If your environment does not already use the provided MQSeries Workflow

default configuration, then the provided sample WMQI Enabler TestSuites that use Workflow will need alteration, if you want to execute them with the MQTester tool. Typical Workflow updates needed to user **fmc** profile file, include the following, before running the **fmczchk** utility:

```
DB2_RR_TO_RS=YES
LANG=en_US
Export DB2_RR_TO_RS
Export LANG
```

Many of the provided jobs use the userid of **userid** within them. You will need to alter this id to match your system needs.

Once the prerequisite software is installed, the WMQI Enabler product can be added to the Solaris® System (along with Windows components), using the following procedure:

Installation procedure

1. Create the WMQI Enabler DB2 database, using the user **db2inst1**.

   Before this is accomplished, we need to ensure WMQI can access the WMQI Enabler DB2 database using TCP/IP due to the limit of accessing at most 10 shared memory segments.

   This can be done using the following procedures:

   i. Ensure the DB2COMM variable is set to use 'tcpip' by issuing the command **db2set -all DB2COMM**.
   This would already be set under a default installation.

   ii. Update the DB2 Database Manager configuration to use the services specified in **/etc/services** for the connection port to the DB2 instance.

   Issue the following command:
   db2 update dbm cfg using svcename db2cdb2inst1
   This would already be set under a default installation.

   iii. Using a DB2 authorized user, create the WMQI Enabler database, bind it, and catalog the database with a set of alias names for ODBC and catalog a local node to resolve the TCP/IP connection.

   You can use the MQSFSEDB file located on the Product CD under /unix directory in theMQSFSE.tar file as input to the DB2 command processor to create the database and alias names.

   Be sure to edit the localhost name to match the localhost name of the machine that this command is executed on.

   DB2 Command line example: **db2 < MQSFSEDB**

It issues the following command:

db2 catalog db MQSFSE as <alias name> at node local

This job creates the database along with its alias names and catalogs them as shown below:

FSE_CRF	**(alias name for Cross Reference File table)**
FSE_ERRL	**(alias name for Error Log table)**
FSE_MSGL	**(alias name for Message Log table)**
FSE_MSGP	**(alias name for Message Profile table)**
FSE_SDR	**(alias name for Symbolic Destination Resolution table)**
FSE_SESS	**(alias name for Session table)**
FSE_STOF	**(alias name for Store Forward table)**
FSE_SYSP	**(alias name for System Profile table)**
FSE_TRAC	**(alias name for Trace table)**
FSE_WFCO	**(alias name for Workflow Correlation table)**

iv. Be sure to add the database alias names of the database to the **/var/mqsi/odbc/.odbc.ini** file for WebSphere MQ Integrator.

This will allow WebSphere MQ Integrator to resolve TCP/IP access to the databases which otherwise would be limited to 10 connections under normal DB2 processing for Solaris®.

Since the WMQI Enabler database name must be unique from the alias names, you must ensure that the actual database name is different from the alias names within the **.odbc.ini** file.

See the following as a partial **.odbc.ini** file for reference:

```
[ODBC Data Sources]
MQSIBKDB=IBM DB2 ODBC Driver
FSE_CRF=IBM DB2 ODBC Driver
FSE_ERRL=IBM DB2 ODBC Driver

.
.

[MQSIBKDB]
Driver=/export/home/db2inst1/sqllib/lib/libdb2.so
Description=MQSIBKDB DB2 ODBC Database
Database=MQSIBKDB

[FSE_CRF]
Driver=/export/home/db2inst1/sqllib/lib/libdb2.so
Description=FSE_CRF DB2 ODBC Database
Database=FSE_CRF

[FSE_ERRL]
Driver=/export/home/db2inst1/sqllib/lib/libdb2.so
Description=FSE_ERRL DB2 ODBC Database
Database=FSE_ERRL

.
.
```

The **.odbc.ini** database names are the alias names and the real database name is defined as a different name.

2. Create the required WMQI DB2 databases and register each Windows based database with ODBC, if they don't already exist:

   MQSICMDB (ConFigMgr database on Windows)

   MQSIBKDB (Broker database on Solaris®)

   MQSIMRDB (Message Repository database on Windows)

   i. Locate file BrokerDB under the /unix directory in the MQSFSE.tar file for a working job of how to create the WMQI Broker database. Edit the file as necessary to resolve the directory location of the SQLLIB path.

   DB2 Command line example: **db2 < BrokerDB**

   ii. Locate file MQSIDBNT under /unix directory in the MQSFSE.tar file for an example of how to create these databases for Windows.

   Copy this file to your Windows system and execute it using a valid, authorized DB2 userid. Edit the file to set the location of the SQLLIB directory for DB2.

   DB2 Command line example: **db2 < MQSIDBNT**

Keep in mind the two files are designed to create and catalog the two databases on Windows and the Broker database on Solaris®. You may need to adjust these files to run on your system and to conform to your naming conventions.

NOTE *They alter the Database Manager Configuration and for the number of database connections and to increase the App_Ctrl_Heap_sz and Applheapsz values, and to alter their Tablespace values.*

3. Update the **UserId** that you are using to perform WebSphere MQ Integrator customization by launching the Solaris Admintool.

i. Right click on the Solaris desktop.

ii. Select Tools, Admintool from the popup menu. This brings up the admin tool, which was used to create/modify userids/groups in Solaris.

iii. Ensure the Userid is a member of the **db2iadm1**, **mqbrkrs** and **mqm** groups.

This user can be used to perform the remaining steps, unless otherwise noted.

4. Using the updated userid from step 3, execute the script file **DBTables**, located under **/unix** directory in the MQSFSE.tar file, to create the WMQI Enabler database tables and to insert required records.

The inserted table records consist of the following:

MQSFSE Error codes.

Default Language record.

MQSFSE Message Profile records for the Hub commands.

Symbolic Destination Resolution record for MQSeries Workflow.

Authentication ID's for 3 specific ID's (USERID, userid, and SysAdmin).

Command line example: **db2 < DBTables**

For more information about message profile, Symbolic Destination Resolution, WMQI Enabler commands, etc., refer to the ***Development Guid***e, *Chapter* 5.

5. Ensure an MQSeries Queue Manager is defined and started on both Solaris® (for WMQI Broker) and Windows (for the WMQI Configuration Manager).

On Solaris® and Windows Queue Managers are required to have the following attributes:

Defined as the (default, optional) Queue Manager (with a user defined name such as **MQSIQM** for Solaris® and **MQSIQMCM** for Windows). Other provided jobs and the sample test cases, reference these Queue Manager names and must be altered if other Queue Manager names are defined.

For Solaris®, the file **crtq.cmd**, located in the /unix directory, provides a working example of how to create this Queue Manager and for Windows, the MQSeries Explorer graphical window can be used to create its Queue Manager.

Have a defined default Dead Letter queue as **SYSTEM.DEAD.LETTER.QUEUE**

The **Log** file size set to **1024**.

The number of **Log Primary** files be set to at least **10**.

The number of **Log Secondary** files be set to at least **12**.

Defined with **Listener** configured for **TCP/IP**.

Listener port set to **1414**, if available (for both Queue Managers).

On Solaris, to start the Queue Manager (MQSIQM) and its Command Server, type the following at a command line: **strmqm MQSIQM** and **strmqcsv MQSIQM**

On Windows, use the MQSeries Explorer window to start the MQSIQMCM Queue Manager if it was not started when it was created.

6. Ensure the WMQI **ConFigMgr** (on Windows), **Broker**, and **UserNameServer** (Solaris®) are defined and started.
   i. Refer to command file **CreateConfigMgr.cmd** located under **/unix** for a working example of how to define the Configuration manager on Windows.

      Note: If the Configuration Manager's Queue Manager name suggested in step 5 is not used (MQSIQMCM), be sure to alter the file to use your name instead. Be sure the Configuration Manager is started under Services within the Windows Control Panel.

   ii. Refer to files, **Broker** and **UserName**, located under **/unix** directory in the MQSFSE.tar file for working examples to create the **MQSIEBROKER** Broker and **UserNameServer** on Solaris.

      This broker name, **MQSIEBROKER**, can be changed to suit your naming conventions. To start the Broker and UserNameServer on unix, issue **mqsistart MQSIEBROKER** and **mqsistart UserNameServer**.

7. Define the WMQI Enabler local Queues:

   HUB_IN

   HUB_IN_ERRORS

   HUB_IN_FAILURE

   HUB_ONLY_OFFLINE

   HUB_ONLY_OFFLINE_FAILURE

   HUB_ONLY_ONLINE

HUB_ONLY_ONLINE_FAILURE

HUB_R_IN

HUB_R_IN_FAILURE

HUB_RWF_IN

HUB_RWF_IN_FAILURE

MQWF_OUT

MQWF_OUT_FAILURE

MQWF_END

MQWF_END_FAILURE

MQWF_DEFAULT_ACTIVITY

MQWF_DEFAULT_ACTIVITY_FAILURE

ERROR_BUCKET

i. You can use the queuesunix.def script file, located under the **/unix** directory in the MQSFSE.tar file, as input to the MQSeries **runmqsc** program.

Command line example: **runmqsc MQSIQM < queuesunix.def**

These queues will be used under the Solaris® based WMQI Queue Manager, as referenced in step 5.

NOTE *The **queuesunix.def** file contains a larger set of local and remote queue definitions than shown above. The additional queues are used by MQTester and sample MQAO adapters. The list shown above consist of the actual queue names used specifically by WMQI Enabler in the product Message Flows.*

ii. To define the needed queues on Windows (used by MQTester) that correspond to the queues with Solaris, use the queuesNT.def file located, under the **/unix** directory in the MQSFSE.tar file, as input to the MQSeries **runmqsc** program on Windows.

Command line example: **runmqsc MQSIQMCM < queuesNT.def**

8. Ensure the MQSeries Workflow configuration utility programs:

fmczutil for Solaris

fmczacfg for Windows

Are used to configure the following components:

Server

Runtime and its database

Buildtime (as needed on the Windows system)

Client

Before starting the Workflow configuration utility, be sure to have the **root** user execute the db2profile and have the DB2 variable DB2_RR_TO_RS set to ensure a clean configuration build. For example:

```
. /export/home/db2inst1/sqllib/db2profile
export DB2_RR_TO_RS=YES
```

Further details for the Solaris configuration utility are listed below:

i. As root, run **fmczutil**
ii. Click **c** to create a new runtime environment
iii. Leave the configuration id **FMC**
Press **Enter** to continue.
iv. Enter the userid to administer the environment
The default is **fmc**
v. Select the category
Type **s** to select server components
Press **Enter** to continue
Next type **x** to start customization
vi. Select an existing or new runtime database
Type **n** to define a new runtime database
Press **Enter** to continue
vii. Select local or remote database
Type **l** to use a local database instance
Press **Enter** to continue
viii. Select the database instance to use
The default is set to the default database instance created
Change this if your instance is other than **db2inst1**
ix. Select the name of the database
FMCDB is the default
Press **Enter** to accept the default name
x. Input the database admin id
Default is **fmc**
xi. Press **Enter** to accept the next 4 questions about directory locations
xii. Press **Enter** to have database space managed by system
xiii. Press **Enter** to accept **fmc** as the user id to use to access this database
xiv. Press **Enter** to accept **FMCGRP** as the system group name
xv. Press **Enter** to accept **FMCSYS** as the system name
xvi. Press **Enter** to accept **FMCQM** as the queue manager name

xvii. Press **Enter** to accept **FMC** as the queue prefix

xviii. Press **Enter** to accept circular logging

xix. Press **Enter** to accept default log file location

xx. Press **Enter** to accept Channel def table

xxi. Specify DNS name of host
Press **Enter** to continue

xxii. Specify port address for the runtime queue manager
Press **Enter** to accept the default value **5010**

xxiii. Input the principal name **fmc**

xxiv. Input a new cluster name or accept the default **FMCGRP**

xxv. Select if this is the first queue manager in the cluster **f**

xxvi. Enter the id of the transaction coordinator
The default is **fmc**

xxvii. Enter if **fmc** is the user id that will normally start the queue manager

xxviii. Press **c** to create a configuration file

The configuration is now finished. Next you will run the configuration to create the runtime components that are needed.

i. Type **y** to create the database

ii. Enter the password requested **fmc** (default) and its confirmation value

iii. Type **y** to create the queue manager

iv. Type **x** to exit utility

The default Workflow configurations for Windows can be used to specify Workflow's Queue Manager, database names, system names, etc. as is without any user modified configuration.

9. i. Create 2 sets of channels under MQSeries that can be used between the WMQI Queue Manager and the MQSeries Workflow Queue Manager.

Refer to the files **mqsiqmcl.mqs** and **fmcqmcl.mqs**, under **/unix**, for working examples of these channel definitions along with their needed transmission queues.

This assumes the Workflow configuration utility (fmczutil) was used to create a workflow configuration. If the default Queue Manager name was changed then these jobs need to reflect this name change as well.

Be sure to alter the **CONNAME** field with the **local hostname** of the machine.

Command line example: **runmqsc MQSIQM < mqsiqmcl.mqs**

Command line example: **runmqsc FMCQM < fmcqmcl.mqs**

ii. Issue this command on Solaris® to start the following channel:

Type: **runmqsc MQSIQM**

Type: **START CHANNEL(MQSIQM.TO.FMCQM)**

Type: **end**

iii. Issue this command on Solaris® to start the following channel:

Type: **runmqsc FMCQM**

Type: **START CHANNEL(FMCQM.TO.MQSIQM)**

Type: **end**

10. i. Create 2 sets of channels for the WMQI Queue Managers between Solaris® and Windows and start them.

You can use files **mqsiunix.mqs** (run on Solaris® system) and **mqsint.mqs**, (run on Windows system) located on the Product CD under directory **/unix** directory in the MQSFSE.tar file, as working examples for this set of channel definitions and their respective transmission queue.

These files assume certain Queue Manager names and must be modified if different names exist from those specified in step 5.

Be sure to altered the **CONNAME** field to match the correct *remote hostname* (or IP address) before running these jobs.

Command line (unix) example: **runmqsc MQSIQM < mqsiunix.mqs**
Command line (Windows) example: **runmqsc MQSIQMCM < mqsint.mqs**

ii. Issue this command on Solaris® to start the following channel:
Type: **runmqsc MQSIQM**

Type: **START CHANNEL(MQSIQM.TO.MQSIQMCM)**

Type: **end**

iii. Issue this command on Windows to start the following channel:
Type: **runmqsc MQSIQMCM**

Type: **START CHANNEL(MQSIQMCM.TO.MQSIQM)**

Type: **end**

Optionally, it may be helpful to observe and directly manipulate the Queue Manager, queues, and channels from the Windows MQSeries Explorer window. To do this perform the following:

i. On Solaris® issue the command: **strmqcsv MQSIQM**

ii. On Solaris® issue the command: **runmqlsr -t tcp -p 1414 -m MQSIQM**

iii. On Solaris® issue the command: **runmqsc MQSIQM**

Type: **DEFINE CHANNEL(SYSTEM.ADMIN.SVRCONN) CHLTYPE(SVRCONN)**

Type: **START CHANNEL(SYSTEM.ADMIN.SVRCONN)**

Type: **END**

iv. On Windows under the MQSeries Explorer window right-click on the **Queue Managers** term, selecting **Show Queue Manager** from the drop-down box.

Select **Show a remote queue manager** radio button.

Enter remote Queue Manager name: **MQSIQM**.

Enter the connection name: **remotehostname(1414)**, where remotehostname is the IP address or hostname of the Solaris system.

11. From the Windows WMQI Control Center create the WMQI Enabler Broker (**MQSIEBROKER**), under the **Topology** tab.

    i. Right click on the **Topology** term in the Topology tab selecting **Check out** from the pop-up menu.

    ii. Right click on the **Topology** term again, selecting **Create**, **Create Broker** from the drop-down box.

    iii. In the Create Window specify the name of the Broker to **MQSIEBROKER** to match the Broker name defined in step 5.

    iv. In the same Create Window specify the Solaris Queue Manager name as **MQSIQM** to match the Queue Manager name defined in step 4.

    This Broker name can be altered, as needed, to conform to your system definition. If this is your first time loading the WMQI Control Center application, you will be prompted in the Configuration Manager Connection window to supply your machine's host name, the port number used for its Queue Manager (1414), and the Queue Manager name (MQSIQMCM).

12. From the Windows WMQI Control Center, import the WMQI Enabler message flows. Import the **MQSFSEMessageFlowExport.xml** file under the **\NT** directory.

    i. From the WMQI Control Center File menu, select **Import to Workspace**.

    ii. Check both **Topics** and **Message Flows** check boxes.

    iii. Click **Import** to import the XML file.

    iv. Click on the **File** menu and select **Check In**, **All (Save to Shared)** to save the import workspace.

    If you are an existing user of WMQI, the WMQI Enabler XML file will add additional Flows and Topics to your system.

13. From the Windows WMQI Control Center, create the MQSIEBROKER configuration under the **Assignments** tab and perform the deploy process to send the configuration to the Broker on Solaris®.
    i. Expand the **+** icon for Message Flows, in the Assignable Resources center window.
    ii. Locate the **HUB_IN_Flow** and drag it into the new broker's **default** execution group.
    iii. Check in this configuration by, right clicking on the **default** Execution group and selecting **Check In** from the drop-down menu
    iv. From the **File** menu select **Deploy**, **Complete Configuration, Normal**.
    v. Click **OK** on the Confirmation screen.

       The deploy process can take several minutes to complete. It may be helpful to start the Windows Task Manager, selecting the Performance tab and observe the utilization rate of the processor(s) to detect when the deploy process has completed. The deploy process will only use one processor if more than one are installed.
    vi. Click on the **Log** tab, check to see if the deploy was successful.

    The success of a deploy process depends on several factors, many of which are hardware or configuration related.

    Once the deployment of **HUB_IN_Flow** has successfully completed, the following main message flows must be deployed.

    MQSFSE_HUB_ONLY_OFFLINE_Flow

    MQSFSE_HUB_ONLY_ONLINE_Flow

    MQSFSE_HUB_R_IN_Flow

    MQSFSE_HUB_RWF_IN_Flow

    MQSFSE_MQWF_END_Flow

    MQSFSE_MQWF_DEFAULT_ACTIVITY_Flow

    MQSFSE_MQWF_OUT_Flow

    MQSFSE_StoreMessageTemplate_Flow

An illustration of the recommended configuration is shown in the following figure:

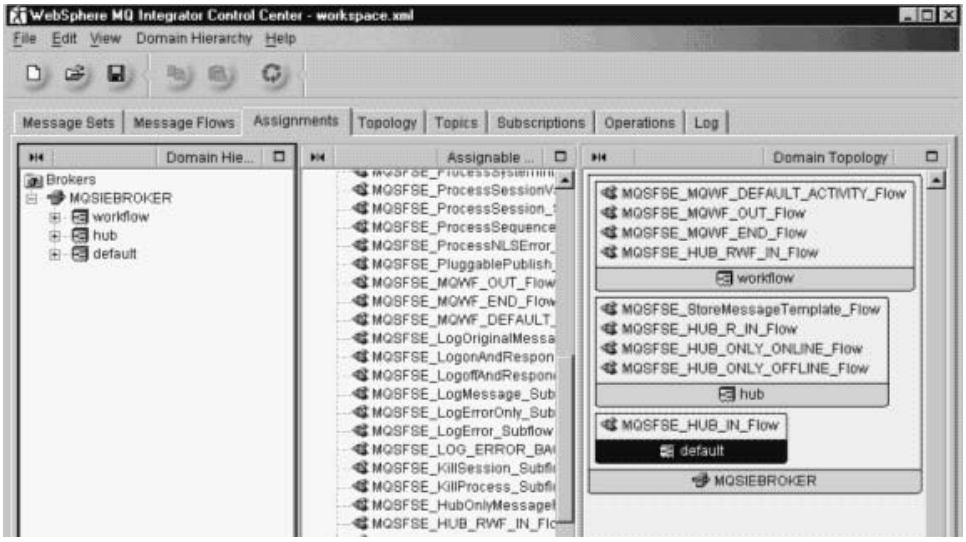

Figure 4: WMQI configuration example.

The following steps will instruct you on completing the deployment of all remaining flows. These instructions will be performed on windows in the Assignments tab. Refer to the figure above for the recommended configuration

i. Check out the broker **MQSIEBROKER** by right-clicking on the broker MQSIEBROKER name and selecting **Check Out**.

ii. Create a new Execution group by right clicking on the broker **MQSIEBROKER** and selecting **Create Execution Group**. Refer the figure above for the recommended execution group configuration names.

iii. Drag one of the designated flows into the **Execution Group**

iv. Check in the **Execution Group** (and the broker **MQSIEBROKER** if necessary).

v. Right click on the **Execution Group** and select **Deploy**, **Delta Assignments Configuration**.

vi. Continue with step iii above until all flows for the execution group have been deployed. The execution group will need to be checked out in order to add a flow. Repeat the steps above to build the remaining Execution group.

14. Under MQSeries Workflow Buildtime (on Windows), import the sample Workflow definition file for use with the use cases.

i. Locate the **MQSFSEWorkFlowProcesses.fdl** file in directory **\NT**

ii. From the Buildtime menu, select **Import** to import the workflow flows.

iii. Check the **Overwrite** option under Import Flags.

iv. Do NOT check the **FDL from MQSeries Workflow Runtime** option under Import Flags.

15. Once imported to Buildtime, ensure the Userid to be used under MQSeries Workflow Client on Solaris® is defined. This userid must be a valid Solaris® user for MQSeries Workflow to successfully process the flows.

    The imported Workflow processes from step 14 adds the ID **USERID** (with password of **PASSWORD**) to the existing Workflow ID of **ADMIN**. MQSeries Workflow displays the ID's in upper case only and a lower case name of userid will work but the password may need to be altered to match a new value such as **password**.

    USERID parameters can be viewed to create a new user ID, if required, by clicking on the **STAFF** tab and expanding the Persons tree structure.

    NOTE *the sample test suites provided with WMQI Enabler are set up to use the ID, USERID within Workflow and the test Use Case messages. The sample test suites would need to be altered if another user ID is used. Also, if another userid is used, this userid must be added to the WMQI Enabler database's Authentication Table, similar in definition to the USERID provided.*

16. On Windows, export the Buildtime environment as a user defined file and copy it to the Solaris® system within the **/var/mqm** directory (if using the default file directories).

    i. From the **Buildtime** menu item, select **Export**, keeping all selections, plus checking the export flag, **export deep**.

    ii. When the configuration is exported, close the Buildtime program. Copy the userBuild-time.fdl file to Solaris under /var/mqm.

17. On Solaris, become user **fmc** by issuing the following command: **su - fmc**

18. On Solaris, using an authorized **fmc fmcgrp** user, execute the MQSeries Workflow program **fmcibie** to import the previous steps MQSeries Workflow Buildtime exported **.fdl** file. The syntax is:

    fmcibie -u ADMIN -p password -i/var/mqm/userBuild-time.fdl -o -t -f

    Where:

    ADMIN and **password** are your previously defined authorized system ID and password.

    var/mqm is the directory path of the exported file.

UserBuild-time.fdl is replaced with the name of the file exported.

WMQI Enabler customization is now complete and ready for testing. Be sure to Start the following if they weren't already started (This is especially important after a reboot):

Queue Managers

Sender Channels

Broker

UserNameServer

MQSeries Workflow

Refer to *Chapter 6 Testing WMQI Enabler Installation* to verify the Installation.

Chapter 6
Testing WMQI Enabler installation

Overview

To verify the WMQI Enabler product installation, the MQTester application is provided with WMQI Enabler and executes on a Windows platform. The tool provides a graphical user interface (GUI) to run a Use Case test such as Show Party. It uses a Sender application to put an XML message request on a queue that is routed through WMQI Enabler to the Receiver application. The Receiver application in turn picks up the request from its input queue and processes the request and sends a response back through WMQI Enabler to the Sender application. To accomplish this, both Sender and Receiver applications use user defined XML messages that consist of the request and the response to the request. Keep in mind that these applications take the place of regular front-end and back-end applications and their adapters to handle any necessary data conversion before an XML message is normally processed using WMQI Enabler.

NOTE *The MQTester tool requires JRE v1.2.2 or higher and support pack MA88 to function.*

Setup

A few steps need to be followed before using the MQTester tool:

1. i. If the MQTester tool is running on the same machine as the deployed WMQI Enabler message flows. The following Sender/Receiver local Queues listed below are needed:

   FEIN (receives responses from WMQI Enabler back to the Sender application)

   BEIN (receives requests from WMQI Enabler to the Receiver application)

   BEIN2 (receives requests from WMQI Enabler as a 2nd Receiver application)

   These queue names are used by the sample TestSuite files and were included in the **queues.def** or **queuesNT.def** file.

   ii. If the MQTester tool is running on a different machine relative to the deployed WMQI Enabler message flows, the queues must be defined as follows:

   a. Make sure **FEIN**, **BEIN**, **BEIN2** are defined as local queues and **HUB_IN** is defined as remote queue on the machine running the MQTester tool.

b. Make sure **HUB_IN** is defined as local queue and **FEIN**, **BEIN**, **BEIN2** are defined as remote queues on the machine running the deployed WMQI Enabler message flows.

2. Create a user-defined directory to hold the MQTester tool and its TestSuites.

   For example: **\tester**.

3. From WMQI Enabler, under **\tester** issue an **xcopy** command to copy all the files and the sub-directory structures for the MQTester tool to the user defined directory in step 2.

   All TestSuite XML files have **C:** and **tester** as the drive letter and directory hard coded in them as the execution location and must be modified if executing on a different drive or directory.

   You may also copy the zipped files **MQTester.zip** and Testsuites to other user directories for other test needs. The unzipped TestSuites will create their set of subdirectories and files.

Execution

Once all the databases, Queue Managers, queues, channels, MQSeries Services are defined and/or activated, you are ready to test WMQI Enabler with the MQTester tool.

To do this, from the user defined test tool application directory, run the **MQTester.bat** file.

The test tool will present a graphical window from which to start.

1. If you are performing testing on a stand alone system,
   From the **File** menu select **Open** Testsuite to open the **HUB_TestSuite.xml** file; Otherwise select the **HUB_TestSuiteunix.xml** file if testing a mix of Windows to a unix system or another Windows based system.

   This should display a list of 3 UseCaseGroups within the window below the menu bar.

2. Click on the first **UseCaseGroup** which in turn will enable the go button (green running figure).

3. Click on this **go** button which will begin the process of sending pre-defined sample message requests and responses.

   As the UseCaseGroup processes, information on the right side of the window is displayed showing the status.

Once the MQTester tool completes execution for the first UseCaseGroup, simply click on the second UseCaseGroup to enable the go button and click on the **go** button to begin its execution.

Finally, when the second UseCaseGroup completes the third UseCaseGroup can be selected and run.

NOTE *For detailed setup instructions regarding MQTester see the* ***Model Office Reference Manual****.*

Chapter 7
Using MQTester

This tool is intended to provide a mechanism to verify the constructed message flows in the WMQI Enabler hub early on during the implementation process, even prior to the final construction of adapter programs. The MQTester tool simulates an WMQI Enabler environment by creating process threads of two types:

MQSender	Simulates a front-end application, submits a request message, and waits for a response.
MQReceiver	Simulates a back-end application, receives a request, and sends the appropriate response.

Different configurations with single or multiple back-end or front-end applications can be simulated and used in conjunction with WMQI Enabler. No actual or real processing is done, but rather the message request is tied to the message response with data mapping where appropriate.

This testing tool allows users to specify XML formatted message files that should be sent to and received from the WMQI Enabler product from the perspectives of a user-defined front-end and back-end MQSeries applications. In addition, users can build compare files for comparing selected output values and also have the ability to specify text or attribute values that must be transferred from the request to the response.

NOTE *For additional information on MQTester see the* ***Model Office Reference Manual****.*

Using MQTester

1. Start MQTester using the **MQTester.bat** file.

   An icon file, **MQTester.ico**, is provided if you want to make a shortcut to MQTester.bat and assign an icon to the shortcut.

   The application starts and appears as below:

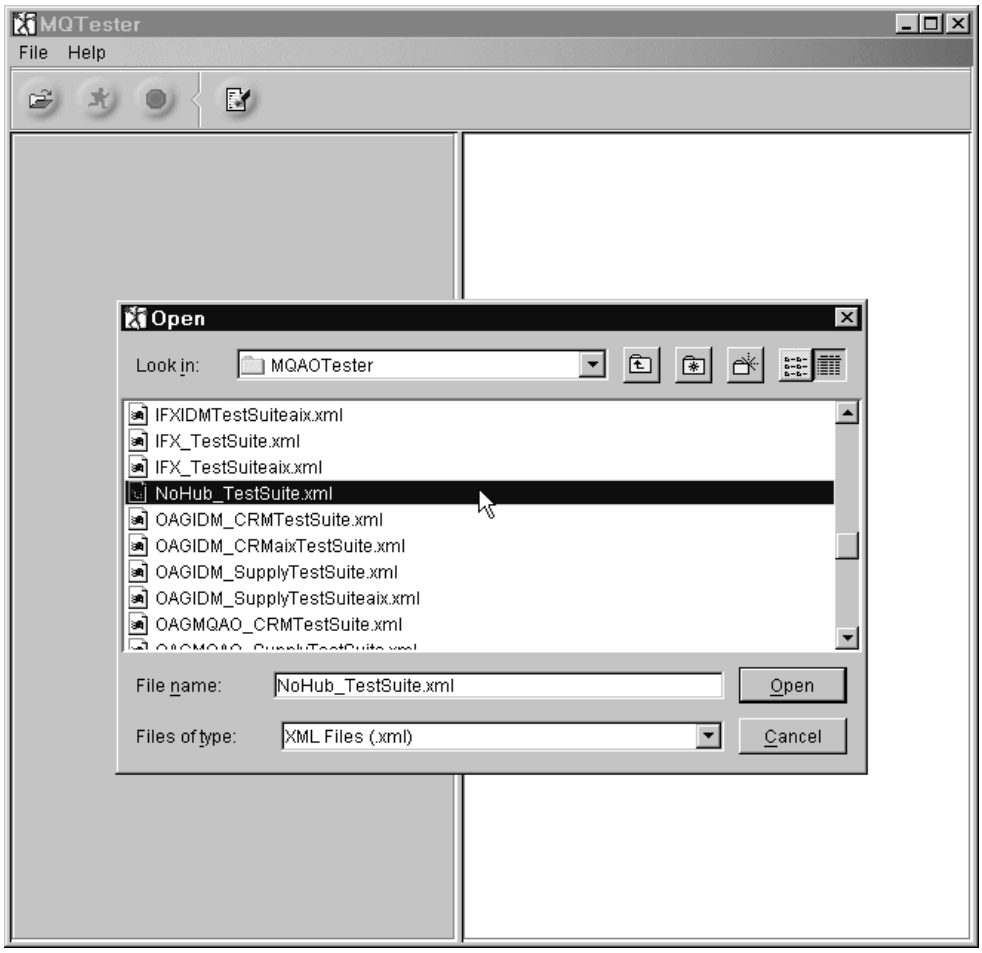

Figure 5: Starting MQTester.

MQTester is shipped with seven predefined TestSuites:

a. Hub: a simple example of using the WMQI Enabler product with an AddParty request and the required system messages for initialization.

b. NoHub: a simple example that connects a front-end directly to a back-end and sends an AddParty request.

c. OAG_CRM: CRM OAG messages and their associated system messages.

d. OAG_SupplySide: Supply Management OAG messages and their associated system messages.

e. OAGIDM-CRM: CRM OAG messages built using IBM's OAG data model.

f. OAGIDM-SupplySide: Supply Management OAG messages built using IBM's OAG data model.

g. Two Backend: is an example of output to two back-end systems.

h. IAACBP: Complex Business Processes example where the MQSeries Workflow process Supervisor is started to initiate a request/response scenario.

2. Select the TestSuite XML file, **No_HubTestSuite.xml** and click **Open**.

   MQTester displays the TestSuite as a tree collapsed to the UseCaseGroup level on the left as shown below:

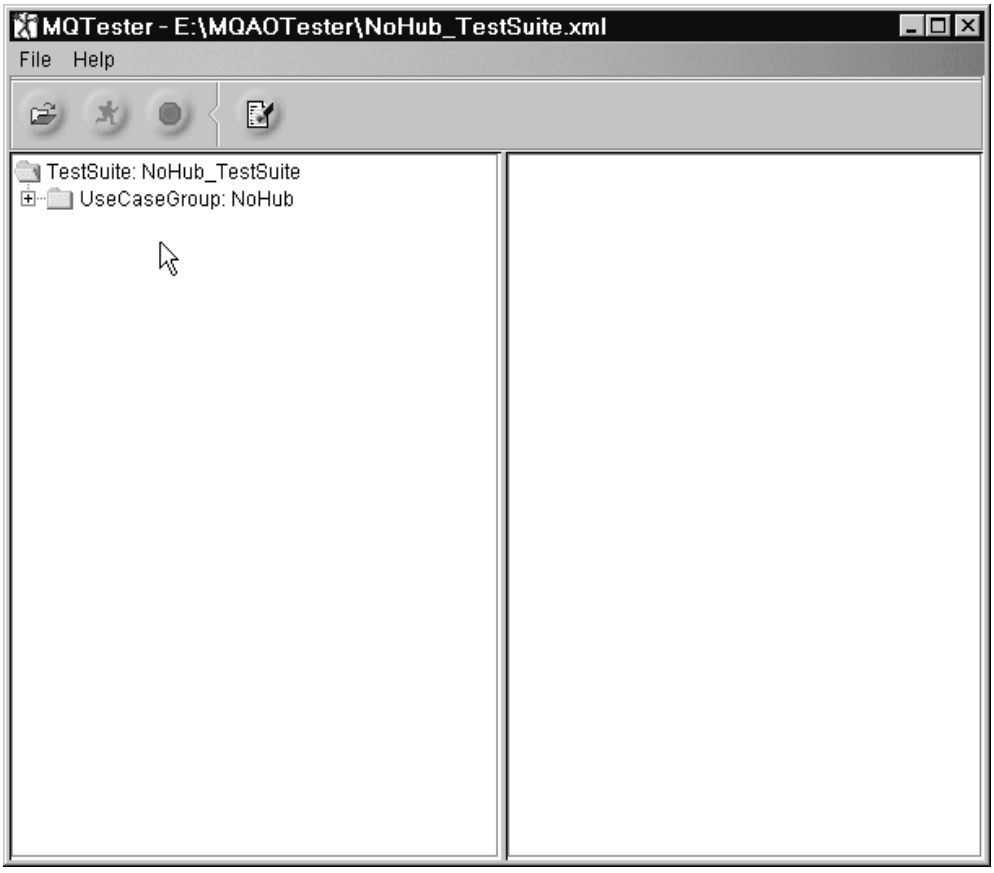

Figure 6: Loading a TestSuite in MQTester.

Expand, collapse, and select any items in the tree, however, the only selections that count are UseCaseGroup selections. Expanding the tree will show you the configurations of the UseCaseGroups.

An example of an expanded tree is shown below:

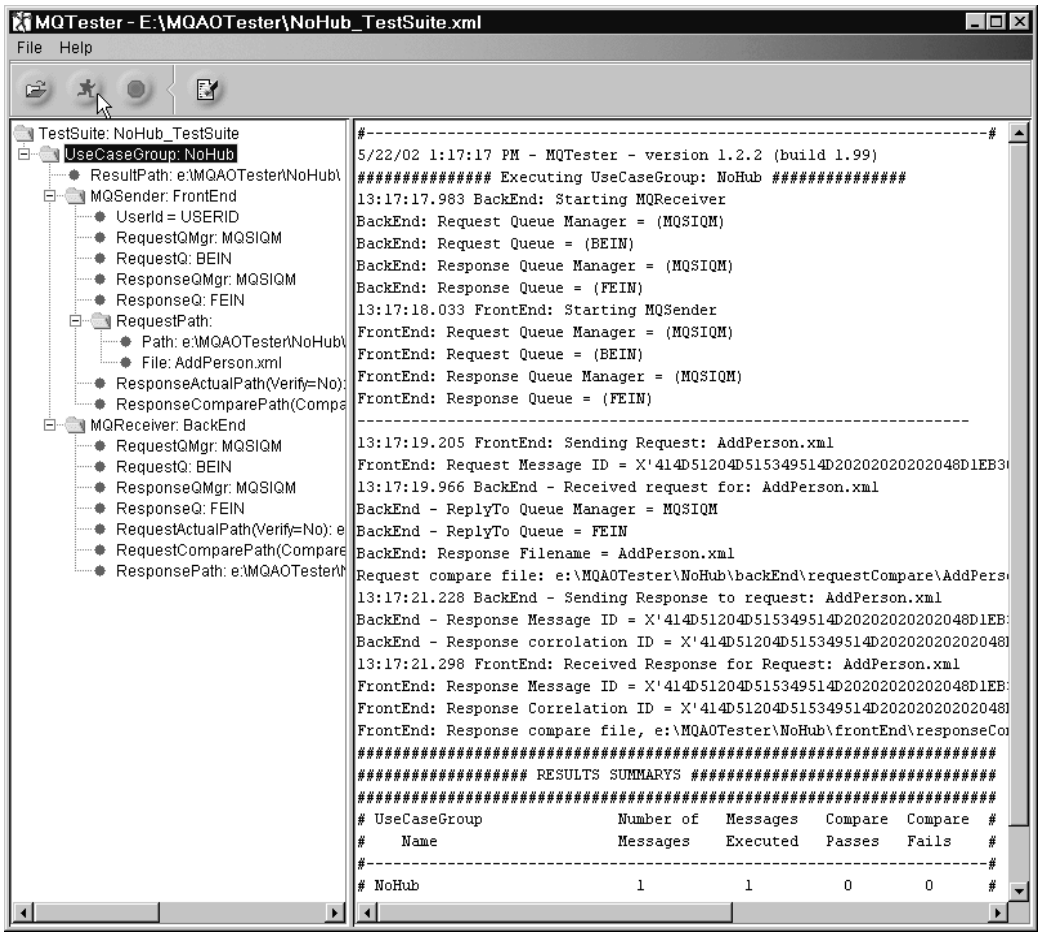

Figure 7: Configuring UseCaseGroups in MQTester.

3. Select the desired use case groups (in this case there is only one).
4. Then execute them using the **go** button (green man running).

   Test results are displayed in the text area on the right and written to the **ResultsPath** directory.

   The results text area can be cleared by pressing the **delete** button (eraser head on paper).

To load and run additional MQTester TestSuite files, select the **open** icon from the tool bar or the File menu.

A Result file is written to the **\No_Hub** directory named **No_Hub.log**, which has the same messages as displayed to the screen.

Configuration Possibilities

The MQTester TestSuite XML file can be configured in several ways with the following restrictions:

- Multiple MQSenders/MQReceivers: The senders and receivers must not share target path definitions or input queue definitions.
- MQSender Only: The back-end that is used must put the MQMD header messageId field in the request into the MQMD header correlationId of the response.
- MQReceiver Only: For an MQReceiver to correlate the incoming requests with the desired response, the request XML must contain a comment to communicate the file name that should be used to compare and respond to the request. The comment must be in the exact form as the example below for file **AddPerson.xml**:

```
<!--filename=AddPerson.xml-->
```

NOTE *comments may not come before the XML declaration, which must be the first item in the document.*

- Multiple MQTester Processes: Multiple MQTester processes can be used if not relying on the same target paths or input queues. Also, results will contain only results for MQSenders and MQReceivers in that process.
- Multiple Machine MQTester Processes: Same as above.

NOTE *For additional configuration options regarding MQTester see the **Model Office Reference Manual**.*

Chapter 8
Installing WMQI Enabler - Configurator

The WMQI Enabler - Configurator is installed by performing the following steps:

1. Create a directory called **D:\wmqiecfg** for the configuration tool.
   Where **D:** is any available drive partition.
2. Unzip the **wmqiecfg.zip** file into the **\wmqiecfg** directory.

Presently, only a Windows version of the WMQI Enabler - Configurator tool exists. This application also requires Java 1.2 runtime or higher and MQSeries Support Pack MA88.

NOTE *Refer to chapter 5 of the **Model Office Reference Manual**, which provides additional configuration options regarding the WMQI Enabler - Configurator.*

Appendix A
Error handling

Error handling in WMQI Enabler is designed to:

1. Catch the error.
2. Record where the error occurred.
3. Report back to the originating system what the error was.

The major flows within WMQI Enabler are constructed to provide this processing. Following the input of a message, the flows begin immediately to check for errors and enters the reporting process. The following comments discuss this architected approach in WMQI terms to demonstrate the emphasis that was given to error handling.

The major message flows have an AdvancedInput_Subflow at the beginning which includes a Try/Catch node positioned just after the Input node to handle any errors that are thrown within these flows. If an error is caught, the message goes to a sub-flow called LogOriginalMessage_Subflow. This sub-flow puts a copy of the message as it was when it initially entered the HUB into the Original_Message_Table of the FSE_ERRL database. This database insert is immediately committed before the database rollback is caused.

Throughout all of the WMQI Enabler WMQI message flows, there are ErrorHandler_Subflows named differently by context and positioned to throw errors back to the LogError_Subflow in each of the major flows. Each ErrorHandler_Subflow has been named to point the user to the place in the flow where the error occurred. For instance, the PrepHubOnlyFlag node in the Hub_In_Flow has its Failure Terminal connected to an instance of the ErrorHandler_Subflow called Hub_In_ErrorHandler_Subflow. If an error were to occur in the PrepHubOnlyFlag node, the Hub_In_ErrorHandler_Subflow would send the message through process NLS error handling. Messages returned to the sending system due to an error will have an ErrorInfo section containing information about the error which occurred in the HUB. Next the message would be set up to route to the LogError_Subflow.

The LogError_Subflow takes care of logging all start and end tracing to the to the Trace_Table of the FSE_ERRL database. Error information is also added to the ErrorTable of the same database. Next the message is checked to make sure that it is not looping through the LogError_Subflow with the same error. Next a response message is built to go back to the sending system. WorkFlow messages are handled a little differently so that MQWorkFlow will get the appropriate return codes and container structure back. Finally the message is sent through the SDR_Subflow to be routed back to the sender. Once the message has been

successfully put to a queue, the process of causing a real failure occurs in order to cause a database rollback. In this process, the message eventually gets all the way back to the AdvancedInput_Subflow at the beginning of the main flow it was in. In this subflow, the message is caught and processed as mentioned in the first paragraph of this section (above).

This table below contains the user specific error codes and descriptions. If you need to report an error, look here for an existing code that suits your purpose. If none is found, create a new one using the "range" information included below.

Reserved	0000 - 0999 <== reserved for development use.
General	1000 - 1499
CRF	1500 - 1999
Reserved	9000 - 9999 <== reserved for customer use.

Error code	Description
1001	Invalid Body Category: specifies the body category which is invalid. Verify that the body category specified in either the bodyCategory of the CrfActionGroup or the cmdType in the COMMAND section of the hub only message has a message profile which exists in the Message_Profile_Table.
1002	Message Type Disabled: specifies the message type that is disabled. This message indicates that the body category specified in either the bodyCategory of the CrfActionGroup or the cmdType in the COMMAND section of the hub only message has a message profile within the Message_Profile_Table that has the MQSI_MSG_ENABLED_FLG set to False.
1003	Invalid Session ID: specifies the session id state which is either Invalid or TimedOut. The sessionId attribute in the message header is Invalid if it contains a value that is not in the Session_Table. SessionId's are issued through the Logon command. The sessionId is TimedOut if it exists on the table, but has been inactive for the time out interval specified in the Session_Table. The interval is set at 60 minutes.
1004	Invalid Sequence: contains the message type dependency for the message being processed. This indicates that the current message requires it's dependency to have been the last processed message type for the same session id.

Error code	Description (Continued)
1005	Invalid Command Type: The hub only command being processed does not have a valid cmdType in the COMMAND section of the message.
1006	Null Message Id: The message sent into the hub for processing does not have a Message Id in it's MQMD header. The attribute name in the header is MsgId.
1007	Error creating/retrieving process id: If the message sent into the hub has a CorrelId in the MQMD header, a corresponding message with a message id that is the same as the CorrelId must exist in the Message_Log_Table. These messages would be part of the same process and would use the same ProcessId. If that corresponding message does not exist, this error is thrown.
1008	Add Work Area Failure: specifies the invalid message type. If the message sent into the hub does not have a high level tag of Message or WfMessage, this error is thrown.
1009	System Interaction Problems: specifies down system. If system interaction checks are performed and find that a required system is not active, this error is generated and specifies which required systems are down.
1010	Queue not found in SDR table: specifies symbolic which failed processing. All system symbolic used as destinations and sources must have a valid entry in the SDR_Table. If the symbolic does not exist in the table, or it's entry does not specify a queue, this error is thrown.
1011	Queue Manager not found in SDR table: specifies symbolic which failed processing. All system symbolic used as destinations and sources must have a valid entry in the SDR_Table. If the symbolic's entry does not specify a queue manager, this error is thrown.
1012	The following required information is null: specifies which required information is NULL. When a message is sent from Workflow into MQWF_OUT, a check is made on information that is required from Workflow for proper WMQI processing. If one of the items that is required does not exist, this error is thrown.
1013	Invalid Hardware Platform entry.

Error code	Description (Continued)
1014	Invalid Product Version entry.
1015	Invalid Default Language entry.
1499	System Generated Exception: This exception is generated when the WMQI product triggers an error, i.e. database exceptions, parsing errors etc. Error 1499 will also be used if a user generated exception is received in the NLS Process Error subflow and the NLS number is not found in the NLS_Error_Message_Table.
1500	CRF Error at...
1502	CRF Error. Translation was requested, but a destinationLogicalId attribute was NOT found on neither the Message nor CrfActionGroup tags. Error at...
1503	CRF Error. Invalid state. Error at...
1504	CRF Error. NULL state attribute on AlternateId. Error at...
1505	CRF Error. NULL keyGroupType attribute on KeyGroup tag. Error at...
1506	CRF Error. Invalid UUID within KeyGroup. Error at...
1507	CRF Error. NULL value attribute on AlternateId. Error at...
1508	CRF Error. NULL sourceLogicalId on AlternateId. Error at...
1509	CRF Error. An AlternateId with a state of 'exists' must be the first and only AlternateId in a KeyGroup. Error at...
1510	CRF Error. AlternateId does not exist in database. Error at...
1512	CRF Error. The AlternateId to be deleted will be deleted by a previous AlternateId that specifies a delete. Error at...
1513	CRF Error. The AlternateId to be deleted was not found in the database. Error at...
1514	CRF Error. The database record for this AlternateId shows it connected to a UUID that does not match the UUID on the KeyGroup. Error at...

Error code	Description (Continued)
1515	CRF Error. The AlternateId(modify) would be deleted by a previous AlternateId(delete). Error at...
1516	CRF Error. The AlternateId(modify) would be modified by a previous AlternateId(modify). Error at...
1517	CRF Error. The AlternateId(modify) was not found in the database. Error at...
1518	CRF Error. The database record for this AlternateId(modify) shows it connected to a UUID that does not match the UUID on the KeyGroup. Error at...
1519	CRF Error. The AlternateId(modify) has sourceLogicalId, newValue, and keyGroupType attributes that already exists in the database. Error at...
1520	CRF Error. The AlternateId(modify) has the same sourceLogicalId, newValue, and keyGroupType attributes as a previous AlternateId(modify). Error at...
1521	CRF Error. An AlternateId with the given sourceLogicalId, keyGroupType, and value already exists in the database. Error at...
1522	CRF Error. Exceeded the per message maximum 999999 AlternateId(s) with a state of "Add". Error at...
1523	CRF Error. No AlternateIds with given keyGroupType and destinationLogicalId were found in database connected to the given UUID. Error at...
1524	CRF Error. NULL value attribute returned from database on lookup. Error at...
1525	CRF Error. The AlternateId to be added is being added by a previous AlternateId(add). Error at...
1526	CRF Error. The AlternateId to be added duplicates a previous AlternateId(modify). Error at...
1527	CRF Error. Invalid System Symbolic. Error at...

Table 1: Error code descriptions.

Appendix B
Database designs

WebSphere MQ Integrator Enabler (WMQI Enabler) uses its database to accomplish several different tasks.

These include the following:

- Support to track the state of XML messages through WMQI Enabler
- Logging of the actual xml messages.
- Recording message profiles of XML requests.
- Maintaining a store & forward capability if systems are inactive.
- Maintaining profiles about each system.
- Providing error log and trace capability.
- Storing MQSeries Workflow interaction parameters.
- Maintaining a Cross Reference File for systems.
- Maintaining a Symbolic Destination Resolution table for systems that WMQI Enabler interacts with.
- Support for NLS error message handling.

The WMQI Enabler database and its tables can be recreated by issuing the following two commands in a DB2 command line:

Create_MQSFSE_Database	(for Windows)
MQSFSEDB	(for Unix)
Create_MQSFSE_Tables	(for Windows)
DBTables	(for Unix)

See the installation instructions in *Chapters 3, 4, and 5* for more details depending on your operating system.

The tables are also populated with records required to configure WMQI Enabler for general setup, i.e., user Ids, Hub Message profiles, NLS errors codes, Installation information, and etc.

Database tables key:

Column Name:	States the name of the column entry in a given database table.
XML Tag Name:	States the XML tag associated with the column names from the database tables as used within the WMQI Enabler Hub commands. If the column name is not used within a Hub command then the Not Applicable ('N/A') symbol is shown instead.
Data Type:	States the data type of the data element stored in the databases.
Nullable:	States if the data element is nullable or not.
Example Value:	Shows an example value for a particular data element.
Description:	Provides a brief description for the data element.

The WMQI Enabler database and its respective tables are listed in the following pages.

WMQI Enabler Database

Generic Message Table Format This is the format of the table that is used when logging messages.

Column Name	XML Tag Name	DataType	Nullable	Example Value	Description
VERSION	N/A	Integer	YES		Structure version number.
REPORT	N/A	Integer	YES		Options for report message.
MSGTYPE	N/A	Integer	YES		Message Type.
EXPIRY	N/A	Integer	YES		Message lifetime.
FEEDBACK	N/A	Integer	YES		Feedback or reason code.
ENCODING	N/A	Integer	YES		Data Encoding.
CODEDCHARSETID	N/A	Integer	YES		Coded character set identifier.
FORMAT	N/A	Char [8]	YES		Format name.
PRIORITY	N/A	Integer	YES		Message priority.
PERSISTENCE	N/A	Integer	YES		Message persistance.
MSGID	N/A	Char [24] FOR BIT DATA	YES		Message identifier.
CORRELID	N/A	Char [24] FOR BIT DATA	YES		Correlation identifier.
BACKOUTCOUNT	N/A	Integer	YES		Backout counter.
REPLYTOQ	N/A	Char [48]	YES		Name of reply queue.
REPLYTOQMGR	N/A	Char [48]	YES		Name of reply queue manager.
USERIDENTIFIER	N/A	Char [12]	YES		User identifier.
ACCOUNTINGTOKEN	N/A	Char [32] FOR BIT DATA	YES		Accounting token.
APPLIDENTITYDATA	N/A	Char [32]	YES		Application data relating to identity.
PUTAPPLTYPE	N/A	Integer	YES		Type of application that put the message.
PUTAPPLNAME	N/A	Char [28]	YES		Name of application that put the message.
PUTDATE	N/A	Date [8]	YES		Date when message was put.
PUTTIME	N/A	Time [8]	YES		Time when message was put.
APPLORIGINDATA	N/A	Char [4]	YES		Application data relating to origin.
GROUPID	N/A	Char [24] FOR BIT DATA	YES		Group identifier.
MSGSEQNUMBER	N/A	Integer	YES		Sequence number of logical message within group.
OFFSET	N/A	Integer	YES		Offset of data physical message from start of logical message.
MSGFLAGS	N/A	Integer	YES		Message flags.
ORIGINALLENGTH	N/A	Integer	YES		Length of original message.
STORED_TIMESTAMP	N/A	TIMESTAMP	NO		Timestamp.
SESN_ID	<sessionId>	Char (26)	YES		Used for searching.
PROCESS_ID		Char (67)	YES		Used for searching.
XML _VERSION	<version>	Char (10)	YES		Version of the XML message.
XML _STANDALONE		Char (10)	YES		Identifies whether or not the DTD is included.
XML_ENCODING	<encoding>	Char (10)	YES		Type of the XML encoding.
XML_DOCTYPE	<Doctype>	VarChar (254)	YES	"MQSFSE_2001.dtd"	Stores the doctype as it is specified in the message.
USER_ID	N/A	Char (26)	YES	"USERID"	UserId of user who wrote, modified, or invalidated this record.
DATE_TIME_ON	N/A	Timestamp	NO	"2001-04-28 09:20:04.139999"	Timestamp signifying date this record became valid.
DATE_TIME_OFF	N/A	Timestamp	YES	"2001-04-28 09:20:04.139999"	Timestamp signifying date this record is no longer valid.
RECORD_STATUS	N/A	Char (50)	YES	"Modified"	User column for possible auditing purposes.
MSG_NAME	N/A	VarChar (254)	YES		By default will be set to ORIGINAL for the original message and must be set by Workflow for any consecutive messages in the same process.
LOG_POINT_NAME	N/A	VarChar (254)	YES		This is set by the LogMessageSubflow. Must be a unique across all records with the same process id.
MSG	N/A	BLOB (2147483647)	YES		Entire set of information related to the message. Not logged compact.

Interaction_Dependency_Table

Primary Key for this table: (MSGID, SYS_SYMBOLIC, DATE_TIME_ON)

Column Name	XML Tag Name	DataType	Nullable	Example Value	Description
MSGID	N/A	Char [24] FOR BIT DATA	NO		Id of the message stored in the table above.
USER_ID	N/A	Char (26)	YES	"USERID"	UserId of user who wrote, modified, or invalidated this record.
DATE_TIME_ON	N/A	Timestamp	NO	"2001-04-28 09:20:04.139999"	Timestamp signifying date this record became valid.
DATE_TIME_OFF	N/A	Timestamp	YES	"2001-04-28 09:20:04.139999"	Timestamp signifying date this record is no longer valid.
RECORD_STATUS	N/A	Char (50)	YES	"Modified"	User column for possible auditing purposes.
SYS_SYMBOLIC	<SystemSymbolic>	VarChar (254)	NO		Symbolic of the system it needs to use.

Message Profile Database (FSE_MSGP) Alias

Message_Profile_Table Used to store all message profile records.

Primary Key(s) for this table: (MSG_TYPE_NAME, DATE_TIME_ON)

Column Name	XML Tag Name	DataType	Nullable	Example Value	Description
USER_ID	N/A	Char (26)	YES	"USERID"	UserId of user who wrote, modified, or invalidated this record.
DATE_TIME_ON	N/A	Timestamp	NO	"2001-04-28 09:20:04.139999"	Timestamp signifying date this record became valid.
DATE_TIME_OFF	N/A	Timestamp	YES	"2001-04-28 09:20:04.139999"	Timestamp signifying date this record is no longer valid.
RECORD_STATUS	N/A	Char (50)	YES	"Modified"	User column for possible auditing purposes.
MQSI_MSG_ENABLED_FLG	<MQSIMessageEnabledFlag>	VarChar (5)	YES	"True"/"False"	Determines whether this message type is allowed for hub processing or not.
MQSI_SESN_VALID_FLG	<MQSISessionValidationFlag>	VarChar (5)	NO	"True"/"False"	Determines whether WMQI is going to do a session validation or not.
MQSI_MSG_SEQ_VALID_FLG	<MQSIMessageSequenceValidationFlag>	VarChar(5)	NO	"True"/"False"	Determines whether WMQI is going to do sequence validation or not.
MQSI_SYS_INTERACTION_CHK_FLG	<MQSISystemInteractionCheckFlag>	VarChar(5)	NO	"True"/"False"	Determines whether WMQI is going to do a system interaction check or not.
TRACE_FLG	<TraceFlag>	VarChar(5)	NO	"True"/"False"	Determines whether or not tacing will be done for this message type.
PUBLISH_FLG	<PublishFlag>	VarChar(5)	YES	"True"/"False"	Determines whether the message should be forwarded to subscribed systems in a non-error situation or not.
OVERRIDE_FLG	<OverrideFlag>	VarChar(5)	YES	"True"/"False"	Determine whether the publish attribute in the message header can be used to specify publishing or not.
PUBLISH_ERRORS_FLG	<PublishErrorsFlag>	VarChar(5)	YES	"True"/"False"	Determine whether the message should be forwarded to subscribed systems or not when an error occurs in this message type.
WRKFLW_MGMNT_FLG	<WorkFlowManagementFlag>	VarChar(5)	NO	"True"/"False"	Determines whether WMQI is going to communicate with Workflow or not.
WRKFLW_DATA_STRUCTURE_NAME		VarChar(254)	YES		The name of the workflowImplInvokeResponse container
WRKFLW_PROCESS_NAME	<WorkFlowProcessName>	VarChar(254)	YES	"SetDestinationIDM"	Allows a message profile to specify a workflow process used to satify that messsage type
WRKFLW_Q_MGR	<WorkFlowQueueManager>	VarChar (254)	YES	"FMCQM"	Specifies the queue manger used to communicate with Workflow.
WRKFLW_Q	<WorkFlowQueue>	VarChar (254)	YES	"FMC.FMCGRP.EXE.XML	Specifies the queue used to communicate with Workflow.

WRKFLW_SYMBOLIC	<WorkFlowSymbolic>	VarChar (254)	YES	"WorkFlow1"	This provides the ability to use a symbolic to be resolved to a queue and queue manager by the SDR.
WRKFLW_REPLY_TO_Q_MGR	<WorkFlowReplyToQueueManager>	VarChar (254)	YES	"MQSIQM"	The queue manager to which WorkFlow sends its reply.
WRKFLW_REPLY_TO_Q	<WorkFlowReplyToQueue>	VarChar (254)	YES	"MQWFEnd"	The queue to which WorkFlow sends its reply.
DEFAULT_DEST_SYMBOLIC	<DefaultDestinationSymbolic>	VarChar (254)	YES	"Party"	The destination symbolic to use when the message goes
HUB_QUEUE_MANAGER	N/A	VarChar (254)	NO	"MQSIQM"	The default MQ Queue Manager Name
PUBLISH_TOPIC	<PublishTopic>	VarChar(254)	YES		Indicates what topic the message will be considered when published. The default is the MSG_TYPE_NAME.
MSG_TYPE_NAME	<MessageTypeName>	VarChar (254)	NO	"AddParty"	This is the unique table key. MessageType is going to correlate to bodyCategory. If bodyCategory equals
MSG_TYPE_DEPENDENCY	<MessageTypeDependency>	VarChar (254)	YES	"ShowParty"	Used to make sure the message sequnce is correct. Contains the message type of the message which must be successfully completed before the current message can run.
USE_HUB_QMGR_AS_REPLY_FLG	<UseHubQMgrAsReplyFlag>	VarChar (5)	YES	"True", “False”	Indicates whether or not response messages to the requesting system should use the HUB_QUEUE_MANAGER specified in the profile as the destination queue manager. If it is “False”, the queue manager in the reply to information of the original request is used.

NOTE: If the WorkFlowQueue column is null then the WorkFlowSymbolic is used. If both are null then the default symbolic of "WorkFlow" is used. The message type list in the MessageTypeDependency column must be the last messagetype in the current session to have completed successfully for the current message to be processed in a valid sequence.

Message_Template_Table

See Generic Message Table Format on page 1 for structure. This table holds a message template. Generally it is used by a process initiated with Workflow communications. A Workflow process would specify a template in this table that would then be retrieved, populated by changes specified by Workflow, and then sent into HUB_IN for processing.

Primary Key(s) for this table: (MSGID, LOG_POINT_NAME, DATE_TIME_ON)

WorkFlow_Parameters_Table

This table holds the parameters to pass in the container section of the initial Workflow meesage.

Primary Key(s) for this table: (MSGID, WF_CORREL_ID, PARAMETER_NAME)

Column Name	XML Tag Name	DataType	Nullable	Example Value	Description
MSGID	N/A	Char [24] FOR BIT DATA	NO		Id of the message stored in the table above.
WF_CORREL_ID	N/A	Char(80)	NO	RUESIDJAIDOSIOSIDFJA	The WorkFlow CorrelationID given to this message by WorkFlow.
USER_ID	N/A	Char (26)	YES	"USERID"	UserId of user who wrote, modified, or invalidated this record.
DATE_TIME_ON	N/A	Timestamp	NO	"2001-04-28 09:20:04.139999"	Timestamp signifying date this record became valid.
DATE_TIME_OFF	N/A	Timestamp	YES	"2001-04-28 09:20:04.139999"	Timestamp signifying date this record is no longer valid.
RECORD_STATUS	N/A	Char (50)	YES	"Modified"	User column for possible auditing purposes.
REQUIRED_PARAMETER_FLG	<RequiredParameterFlag>	VarChar(5)	NO	"True"/"False"	Determines whether or not the above fields are required.
MSG_TYPE_NAME	<MessageTypeName>	VarChar (254)	YES	"AddParty"	This is not a database key, but it is the column that you will be using as the key.
DEFAULT_VALUE	<DefaultValue>	VarChar (254)	YES	"ok"	The value of the field specified by the ParameterName column if the path specified by the ParameterPath column is null.

PARAMETER_NAME	<ParameterName>	VarChar (254)	NO	"WFTransition"	This is the name of the field to place in the WorkFlow message container section.
PARAMETER_PATH	<ParameterPath>	VarChar (1000)	NO	"Message.ErrorInfo.errorS	ESQL code that defines the location within the message to retrieve the value to place in the field specified by the ParameterName column.

NOTE: If the RequiredParameter column is False and the DefaultValue column is null then the parameter is not passed to workflow. Also if the RequiredParameter column is True, the DefaultValue column is null, and the path given by the ParameterPath column does not lead to a value in the message, then an error will occur and the workflow program will not receive the message. This error will be sent back to the sending system. The order o all records with the same MessageTypeName value determines the order in which the Parameters will be sent to the workflow program. This table will be used when using ProcessWorkFlowStart.

Example:

RequiredParameterFlag	DefaultValue	Parameter Path Results	Processing Results	Description
True	NOT NULL	NOT NULL	Message value sent	
True	NOT NULL	NULL	Default value sent	
True	NULL	NOT NULL	Message value sent	
True	NULL	NULL	Error sent to sender	
False	NOT NULL	NOT NULL	Message value sent	
False	NOT NULL	NULL	Default value sent	
False	NULL	NOT NULL	Message value sent	
False	NULL	NULL	Parameter not sent	

Build_WorkFlow_Parameters_Table This table holds the parameters to pass in the container

Primary Key(s) for this table: (MSG_TYPE_NAME, PARAMETER_NAME, DATE_TIME_ON)

Column Name	XML Tag Name	DataType	Nullable	Example Value	Description
USER_ID	N/A	Char (26)	YES	"USERID"	UserId of user who wrote, modified, or invalidated this record.
DATE_TIME_ON	N/A	Timestamp	NO	"2001-04-28 09:20:04.139999"	Timestamp signifying date this record became valid.
DATE_TIME_OFF	N/A	Timestamp	YES	"2001-04-28 09:20:04.139999"	Timestamp signifying date this record is no longer valid.
RECORD_STATUS	N/A	Char (50)	YES	"Modified"	User column for possible auditing purposes.
REQUIRED_PARAMETER_FLG	<RequiredParameterFlag>	VarChar(5)	NO	"True"/"False"	Determines whether or not the above fields are required.
MSG_TYPE_NAME	<MessageTypeName>	VarChar (254)	NO	"AddParty"	This is not a database key, but it is the column that you will be using as the key.
DEFAULT_VALUE	<DefaultValue>	VarChar (254)	YES	"ok"	The value of the field specified by the ParameterName column if the path specified by the ParameterPath column is null.
PARAMETER_NAME	<ParameterName>	VarChar (254)	NO	"WFTransition"	This is the name of the field to place in the WorkFlow message container section.
PARAMETER_PATH	<ParameterPath>	VarChar (1000)	NO	"Message.ErrorInfo.errorS	ESQL code that defines the location within the message to retrieve the value to place in the field specified by the ParameterName column.

NOTE: If the RequiredParameter column is False and the DefaultValue column is null then the parameter is not passed to workflow. Also if the RequiredParameter column is True, the DefaultValue column is null, and the path given by the ParameterPath column does not lead to a value in the message, then an error will occur and the workflow program will not receive the message. This error will be sent back to the sending system. The order o all records with the same MessageTypeName value determines the order in which the Parameters will be sent to the workflow program. This table will be used when using ProcessWorkFlowStart.

System_Interaction_Table This table contains a list of systems that a message type can interact with.

Primary Key(s) for this table: (SYS_SYMBOLIC, MSG_TYPE_NAME, DATE_TIME_ON)

Column Name	XML Tag Name	DataType	Nullable	Example Value	Description
USER_ID	N/A	Char (26)	YES	"USERID"	UserId of user who wrote, modified, or invalidated this record.
DATE_TIME_ON	N/A	Timestamp	NO	"2001-04-28 09:20:04.139999"	Timestamp signifying date this record became valid.

DATE_TIME_OFF	N/A	Timestamp	YES	"2001-04-28 09:20:04.139999"	Timestamp signifying date this record is no longer valid.
RECORD_STATUS	N/A	Char (50)	YES	"Modified"	User column for possible auditing purposes.
SYSTEM_ORDER_NUMBER	N/A	Integer	YES		Indicates the order in which the systems appear in the system interaction list. Ensures when the systems are retrieved from the database, they are sorted in the same order as when they are put to the database.
REQUIRED_INTERACTION_FLG	N/A	VarChar(5)	NO	"True"/"False"	This states do I have to talk to the system to be able to process this message.
SYS_SYMBOLIC	N/A	VarChar (254)	NO	"CIIS"	This is the same as the system symbolic used to do SDR resolution.
SYS_BACKUP	N/A	VarChar (254)	YES	"CISSBackup1"	This can be a different system that happens to be, in this case, doing the same processing, but it may not be the exact same system.
MSG_TYPE_NAME	N/A	VarChar (254)	NO	"AddParty"	This is not a database key, but it is the column that you will be using as the key.

System Profile Database (FSE_SYSP) Alias

System_Status_Table States whether or not systems are up or down and if they want to go down.

Primary Key(s) for this table: (SYS_SYMBOLIC, DATE_TIME_ON)

Column Name	XML Tag Name	DataType	Nullable	Example Value	Description
USER_ID	N/A	Char (26)	YES	"USERID"	UserId of user who wrote, modified, or invalidated this record.
SYS_REQ_SHUTDOWN_TIMESTAMP	N/A	TIMESTAMP	YES		The time at which the hub started to process the shutdown request message.
EST_TIMEUP	N/A	TIMESTAMP	YES	...	Don't know if we are going to be using.
DATE_TIME_ON	N/A	Timestamp	NO	"2001-04-28 09:20:04.139999"	Timestamp signifying date this record became valid.
DATE_TIME_OFF	N/A	Timestamp	YES	"2001-04-28 09:20:04.139999"	Timestamp signifying date this record is no longer valid.
RECORD_STATUS	N/A	Char (50)	YES	"Modified"	User column for possible auditing purposes.
ERROR_MSG_DEST	N/A	VarChar(7)	YES	"SOURCE", "BOTH", "DEFAULT", "NONE"	Indicates the desired destination of error messages in relation to the source system and the default error queue.
RETURN_SUCCESS_CODE	N/A	VarChar(5)	YES	"65338"	Indicates the successful code required by this system when sending a response.
RETURN_FAILURE_CODE	N/A	VarChar(5)	YES	"0"	Indicates the failure code required by this system when sending a response.
SYS_ACTIVE_FLG	N/A	VarChar(5)	NO	"True"/"False"	States whether the system is up or not. The default is False.
SYS_REQ_SHUTDOWN_FLG	N/A	VarChar(5)	NO	"True"/"False"	States whether the system requested to be down or not. The default is False.
BLOCKED_BY_SYS_INTERACTION_FLG	N/A	VarChar(5)	NO	"True"/"False"	Determines if the system was blocked from shutting down due to other processes being in their interaction check phase when the shutdown request was received. The defult is False.
LANGUAGE	N/A	Char(2)	YES	"10"	Indicates the NLS default language
SYS_SYMBOLIC	N/A	VarChar (254)	NO	"CIIS"	Unique table key.

System_Backup_Table List of secondary systems that can process messages when primary systems are unvailable.

Primary Key(s) for this table: (SYS_SYMBOLIC, MSG_TYPE_NAME, DATE_TIME_ON)

Column Name	XML Tag Name	DataType	Nullable	Example Value	Description
USER_ID	N/A	Char (26)	YES	"USERID"	UserId of user who wrote, modified, or invalidated this record.
DATE_TIME_ON	N/A	Timestamp	NO	"2001-04-28 09:20:04.139999"	Timestamp signifying date this record became valid.
DATE_TIME_OFF	N/A	Timestamp	YES	"2001-04-28 09:20:04.139999"	Timestamp signifying date this record is no longer valid.
RECORD_STATUS	N/A	Char (50)	YES	"Modified"	User column for possible auditing purposes.

SYS_SYMBOLIC	<SystemSymbolic>	VarChar (254)	NO	"CIISMirror1"	This is the same as the system symbolic used to do SDR resolution.
MSG_TYPE_NAME	<MessageTypeName>	VarChar (254)	NO	"AddParty"	This is not a database key, but it is the column that you will be using as the key.
NEXT_BACKUP	<NextBackup>	VarChar (254)	YES	"CIISMirror2"	This is the link (pointer) to the next backup in the chain. If it's null then the chain ends.

System_Store_Flag_Table Whether or not we are going to be storing the message if we can't get to the system.

Primary Key(s) for this table: (SYS_SYMBOLIC, MSG_TYPE_NAME, DATE_TIME_ON)

Column Name	XML Tag Name	DataType	Nullable	Example Value	Description
USER_ID	N/A	Char (26)	YES	"USERID"	UserId of user who wrote, modified, or invalidated this record.
DATE_TIME_ON	N/A	Timestamp	NO	"2001-04-28 09:20:04.139999"	Timestamp signifying date this record became valid.
DATE_TIME_OFF	N/A	Timestamp	YES	"2001-04-28 09:20:04.139999"	Timestamp signifying date this record is no longer valid.
RECORD_STATUS	N/A	Char (50)	YES	"Modified"	User column for possible auditing purposes.
STORE_FLG	<StoreFlag>	VarChar(5)	NO	"True"/"False"	Do I want to store the message or kill it if a system interaction problem takes place.
SYS_SYMBOLIC	<SystemSymbolic>	VarChar (254)	NO	"Web1"	For this particular system a unique key.
MSG_TYPE_NAME	<MessageTypeName>	VarChar (254)	NO	"AddParty"	For this particular message.

Session Database (FSE_SESS) Alias

Session_Table This table keeps up with whether there is a valid session or not, and whether it is timed out or not.

Primary Key for this table: (SESN_ID)

Column Name	XML Tag Name	DataType	Nullable	Example Value	Description
SESN_ID	N/A	Char (26)	NO		This will be unique to this table.
SESN_INACTIVE_TMOUT_START	N/A	TIMESTAMP	YES		The last time a process completed and it was the last process running. This gets set.
SESN_INACTIVE_TMOUT_INTRVL	N/A	Integer	NO		This is set and not changed once this session record is build.
USER_ID	N/A	Char (26)	YES	"USERID"	UserId of user who wrote, modified, or invalidated this record.
DATE_TIME_ON	N/A	Timestamp	NO	"2001-04-28 09:20:04.139999"	Timestamp signifying date this record became valid.
DATE_TIME_OFF	N/A	Timestamp	YES	"2001-04-28 09:20:04.139999"	Timestamp signifying date this record is no longer valid.
RECORD_STATUS	N/A	Char (50)	YES	"Modified"	User column for possible auditing purposes.
ACTIVE_PROCESSES_FLG	N/A	VarChar(5)	NO	"True"/"False"	States whether or not there are currently processes running in this session.

Session_Processes_and_System_Usage_Table Keeps up with what processes are being performed and which systems each are using. There are multiple records for each process. One for each system the process is using.

Column Name	XML Tag Name	DataType	Nullable	Example Value	Description
PROCESS_ID	N/A	Char (67)	NO		The Id of the running process.
SESN_ID	N/A	Char (26)	YES		Not every process belongs to a session, so it is nullable.
USER_ID	N/A	Char (26)	YES	"USERID"	UserId of user who wrote, modified, or invalidated this record.
DATE_TIME_ON	N/A	Timestamp	NO	"2001-04-28 09:20:04.139999"	Timestamp signifying date this record became valid.
DATE_TIME_OFF	N/A	Timestamp	YES	"2001-04-28 09:20:04.139999"	Timestamp signifying date this record is no longer valid.
RECORD_STATUS	N/A	Char (50)	YES	"Modified"	User column for possible auditing purposes.
SYS_REQUESTED_SHUTDDOWN_FLG	N/A	VarChar(5)	NO	"True"/"False"	whether or not the system requested to shutdown while it was being used by this process.
SYS_SYMBOLIC	N/A	VarChar (254)	YES	"Party"	Symbolic of a system that the process is using.

Process_State_Table This will keep up with the state of processes.

Primary Key for this table: (PROCESS_ID)

Column Name	XML Tag Name	DataType	Nullable	Example Value	Description
SESN_ID	N/A	Char (26)	YES		Has to have one.
PROCESS_ID	N/A	Char (67)	NO		Has to have one.
STATE_CHANGE_TIMESTAMP	N/A	TIMESTAMP	YES		Time the state changed.
PROCESS_INITIATOR_REPLYTOQ	N/A	Char (48)	YES		What queue to respond to.
PROCESS_INITIATOR_REPLYTOQ_MGR	N/A	Char (48)	YES		QMGR Name of queue to repond to.
PROCESS_COMPLETION_TIMESTAMP	N/A	Char (28)	NO		The time the process completed.
USER_ID	N/A	Char (26)	YES	"USERID"	UserId of user who wrote, modified, or invalidated this record.
DATE_TIME_ON	N/A	Timestamp	NO	"2001-04-28 09:20:04.139999"	Timestamp signifying date this record became valid.
DATE_TIME_OFF	N/A	Timestamp	YES	"2001-04-28 09:20:04.139999"	Timestamp signifying date this record is no longer valid.
RECORD_STATUS	N/A	Char (50)	YES	"Modified"	User column for possible auditing purposes.
PROCESS_STATE	N/A	Char (50)	YES		This identifies the current state of the process.
PROCESS_INITIATOR_SYMBOLIC	N/A	VarChar (254)	YES		System that issued the request.
MSG_TYPE_NAME	N/A	VarChar (254)	NO		States the type of the message so it can be correlated with MessageTypeDependency.

NOTE: WMQI assigns a ProcessId to all Use cases i.e. Work Flow process.

Session_Authentication_Table Table must be updated by the user for all userid's if using session validation

Primary Key for this table: (AUTHENTICATION_ID)

Column Name	XML Tag Name	DataType	Nullable	Example Value	Description
AUTHENTICATION_ID	N/A	Char (50)	NO	"dbashant"	When logging on to a session, if one of these values matches the AuthenticationId in the message, a session is awarded.
USER_ID	N/A	Char (26)	YES	"USERID"	UserId of user who wrote, modified, or invalidated this record.
DATE_TIME_ON	N/A	Timestamp	NO	"2001-04-28 09:20:04.139999"	Timestamp signifying date this record became valid.
DATE_TIME_OFF	N/A	Timestamp	YES	"2001-04-28 09:20:04.139999"	Timestamp signifying date this record is no longer valid.
RECORD_STATUS	N/A	Char (50)	YES	"Modified"	User column for possible auditing purposes.
ATTRIBUTE_STRING	N/A	VarChar (254)	YES	"Siebel"	Extra information to keep about an AuthenticationId.

Error Log Database (FSE_ERRL) Alias

Error_Table WMQI Enabler processing errors.

Column Name	XML Tag Name	DataType	Nullable	Example Value	Description
MSGID	N/A	Char [24] FOR BIT DATA	YES		Used for searching.
PROCESS_ID	N/A	Char (67)	YES		Used for searching.
SESN_ID	N/A	Char (26)	YES		Used for searching.
RECORD_TIMESTAMP	N/A	TIMESTAMP	YES		When this record was created.
USER_ID	N/A	Char (26)	YES	"USERID"	UserId of user who wrote, modified, or invalidated this record.
DATE_TIME_ON	N/A	Timestamp	YES	"2001-04-28 09:20:04.139999"	Timestamp signifying date this record became valid.
DATE_TIME_OFF	N/A	Timestamp	YES	"2001-04-28 09:20:04.139999"	Timestamp signifying date this record is no longer valid.
RECORD_ STATUS	N/A	Char (50)	YES	"Modified"	User column for possible auditing purposes.
ERROR	N/A	VarChar (1000)	YES	"SDR_Failure"	Type of error.

Exception_Table One record for each exception associated with the message.

Column Name	XML Tag Name	DataType	Nullable	Example Value	Description

MSGID	N/A	Char [24] FOR BIT DATA	YES		Used for searching.
PROCESS_ID	N/A	Char (67)	YES		Used for searching.
SESN_ID	N/A	Char (26)	YES		Used for searching.
USER_ID	N/A	Char (26)	YES	"USERID"	UserId of user who wrote, modified, or invalidated this record.
SEVERITY	N/A	Integer	YES		
LINE	N/A	Integer	YES		
DATE_TIME_ON	N/A	Timestamp	YES	"2001-04-28 09:20:04.139999"	Timestamp signifying date this record became valid.
DATE_TIME_OFF	N/A	Timestamp	YES	"2001-04-28 09:20:04.139999"	Timestamp signifying date this record is no longer valid.
RECORD_STATUS	N/A	Char (50)	YES	"Modified"	User column for possible auditing purposes.
FILE	N/A	VarChar (254)	YES		File name and path.
FUNCTION	N/A	VarChar (254)	YES		
TYPE	N/A	VarChar (254)	YES		
NAME	N/A	VarChar (254)	YES		
LABEL	N/A	VarChar (254)	YES		
TEXT	N/A	VarChar (254)	YES		Node throwing exception.
CATALOG	N/A	VarChar (254)	YES		Version of WMQI.

Message_Table This is how the message looked when the error happened.
See Generic Message Table Format on page 1.

Stored_Message_Table See Generic Message Table Format on page 1.

Trace_Table Records trace entries when WMQI errors are thrown.

Column Name	XML Tag Name	DataType	Nullable	Example Value	Description
MSGID	N/A	Char [24] FOR BIT DATA	YES		Used for searching.
PROCESS_ID	N/A	Char (67)	YES		Used for searching.
SESN_ID	N/A	Char (26)	YES		Used for searching.
TRACE_NAME	N/A	Char (10)	YES		Name of the trace.
TRACE_STATUS	N/A	Char (50)	YES		Is it still recording or is the trace complete.
TRACE_RESULT	N/A	Char (50)	YES		Outcome of the trace.
TRACE_TYPE	N/A	Char(5)	YES		Used to specify a Start or End trace.
USER_ID	N/A	Char (26)	YES	"USERID"	UserId of user who wrote, modified, or invalidated this record.
DATE_TIME_ON	N/A	Timestamp	YES	"2001-04-28 09:20:04.139999"	Timestamp signifying date this record became valid.
DATE_TIME_OFF	N/A	Timestamp	YES	"2001-04-28 09:20:04.139999"	Timestamp signifying date this record is no longer valid.
RECORD_STATUS	N/A	VarChar (50)	YES	"Modified"	User column for possible auditing purposes.

Original_Message_Table It stores the original state of the message.
See Generic Message Table Format on page 1.

Message Log Database (FSE_MSGL) Alias

Message_Log_Table See Generic Message Table Format on page 1.
Primary Key(s) for this table: (MSGID, LOG_POINT_NAME, DATE_TIME_ON)

WorkFlow Correlation Database (FSE_WFCO) Alias

WorkFlow_Parameters_Table
Primary Key(s) for this table: (MSGID, WF_CORREL_ID, PARAMETER_NAME)

Column Name	XML Tag Name	DataType	Nullable	Example Value	Description
MSGID	N/A	Char [24] FOR BIT DATA	NO		Id of the message stored in the table above.

WF_CORREL_ID	N/A	Char(80)	NO	RUESIDJAIDOSIOSIDFJA	The WorkFlow CorrelationID given to this message by WorkFlow.
USER_ID	N/A	Char (26)	YES	"USERID"	UserId of user who wrote, modified, or invalidated this
DATE_TIME_ON	N/A	Timestamp	NO	"2001-04-28	Timestamp signifying date this record became valid.
DATE_TIME_OFF	N/A	Timestamp	YES	"2001-04-28	Timestamp signifying date this record is no longer valid.
RECORD_STATUS	N/A	Char (50)	YES	"Modified"	User column for possible auditing purposes.
REQUIRED_PARAMETER_FLG	N/A	VarChar(5)	NO	"True"/"False"	Determines whether or not the above fields are required.
DEFAULT_VALUE	N/A	VarChar (254)	YES	"ok"	The value of the field specified by the ParameterName
MSG_TYPE_NAME	N/A	VarChar (254)	YES	"AddParty"	This is not a database key, but it is the colunm that you will
PARAMETER_NAME	N/A	VarChar (254)	NO	"WFTransition"	This is the name of the field to place in the WorkFlow message container section.
PARAMETER_PATH	N/A	VarChar (1000)	NO	"Message.ErrorInfo.errorS	ESQL code that defines the location within the message to retrieve the value to place in the field specified by the

NOTE: If the RequiredParameter column is False and the DefaultValue column is null then the parameter is not passed to workflow. Also if the RequiredParameter column is True, the DefaultValue column is null, and the path given by the ParameterPath column does not lead to a value in the message, then an error will occure and the workflow program will not recieve the message. This error will be sent back to the sending system. The order of all records with the same MessageTypeName value determines the order in which the Parameters will be sent to the workflow program.

Example:

RequiredParameterFlag	DefaultValue	Parameter Path Results	Processing Results	Description
True	NOT NULL	NOT NULL	Message value sent	
True	NOT NULL	NULL	Default value sent	
True	NULL	NOT NULL	Message value sent	
True	NULL	NULL	Error sent to sender	
False	NOT NULL	NOT NULL	Message value sent	
False	NOT NULL	NULL	Default value sent	
False	NULL	NOT NULL	Message value sent	
False	NULL	NULL	Parameter not sent	

WorkFlow_Correl_Table

Primary Key(s) for this table: (MSGID, WF_CORREL_ID)

Column Name	XML Tag Name	DataType	Nullable	Example Value	Description
WF_CORREL_ID	N/A	Char (80)	NO	"RUESIDJAIDOSIOSIDFJ	The WorkFlow CorrelationID given to this message by WorkFlow.
MSGID	N/A	Char [24] FOR BIT DATA	NO		The MsgID in the MQMD of this message.
USER_ID	N/A	Char (26)	YES	"USERID"	UserId of user who wrote, modified, or invalidated this record.
DATE_TIME_ON	N/A	Timestamp	NO	"2001-04-28 09:20:04.139999"	Timestamp signifying date this record became valid.
DATE_TIME_OFF	N/A	Timestamp	YES	"2001-04-28	Timestamp signifying date this record is no longer valid.
RECORD_STATUS	N/A	Char (50)	YES	"Modified"	User column for possible auditing purposes.
INCOMING_MSG_NAME	N/A	VarChar (254)	YES	"Message2"	Incoming message should be given this name.
WRKFLOW_DATA_STRUCTURE_NAME	N/A	VarChar (254)	NO		The name of the response data strcuture to workflow.
ORIGINAL_MSGID	N/A	Char [24] FOR BIT DATA	NO	"9999999999999"	The MQMD message ID of the original message that started a workflow process.

Cross Reference File (CRF) Database (FSE_CRF) Alias

CRF_Table

Primary Key(s) for this table: (SYS_SYMBOLIC, SYS_KEY, KEY_TYPE, DATE_TIME_ON)

Column Name	XML Tag Name	DataType	Nullable	Example Value	Description
UUID	N/A	Char (27) FOR BIT DATA	NO	"20010417002302222111	The UUID of this entity as it is known to the HUB.

USER_ID	N/A	Char (26)	YES	"USERID"	UserId of user who wrote, modified, or invalidated this record.
DATE_TIME_ON	N/A	Timestamp	NO	"2001-04-28 09:20:04.139999"	Timestamp signifying date this record became valid.
DATE_TIME_OFF	N/A	Timestamp	YES	"2001-04-28 09:20:04.139999"	Timestamp signifying date this record is no longer valid.
RECORD_STATUS	N/A	Char (50)	YES	"Modified"	User column for possible auditing purposes.
SYS_SYMBOLIC	N/A	VarChar (254)	NO	"Siebel"	The common name for the System using this record.
SYS_KEY	N/A	VarChar (254)	NO	"Siebel0000112233"	Key of this entity as it is referred to by the owning System.
KEY_TYPE	N/A	VarChar (254)	NO	"Person"	Type of key
ATTRIBUTE_STRING	N/A	VarChar (254)	YES	"Tall, Dark and Scary"	Extra info a system may want to save about an entity.

Symbolic Destination Resolution (SDR) Database (FSE_SDR) Alias

SDR_Table

Primary Key(s) for this table: (SYS_SYMBOLIC, DATE_TIME_ON)

Column Name	XML Tag Name	DataType	Nullable	Example Value	Description
USER_ID	N/A	Char (26)	YES	"USERID"	UserId of user who wrote, modified, or invalidated this record.
DATE_TIME_ON	N/A	Timestamp	NO	"2001-04-28 09:20:04.139999"	Timestamp signifying date this record became valid.
DATE_TIME_OFF	N/A	Timestamp	YES	"2001-04-28 09:20:04.139999"	Timestamp signifying date this record is no longer valid.
RECORD_STATUS	N/A	Char (50)	YES	"Modified"	User column for possible auditing purposes.
SYS_SYMBOLIC	<SystemSymbolic>	VarChar (254)	NO	"Siebel"	The common name for the System using this record.
Q_NAME	<Queue>	VarChar (254)	NO	"FEIN"	Queue on which this System looks for HUB output.
Q_MGR_NAME	<QueueManager>	VarChar (254)	YES	"SBLQM1"	Queue Manager on which this System looks for HUB output.

Install_Data_Table

Column Name	XML Tag Name	DataType	Nullable	Example Value	Description
HARDWARE_PLATFORM	<Hardware_Platform>	Char (1)	NO	"NT"	The type of hardware on which the HUB is installed.
PRODUCT_VERSION	<Product_Version>	Char (2)	NO	"01"	Indicates the code release number
DEFAULT_LANGUAGE	<Default_Language>	Char(2)	NO	"10"	Indicates the NLS default language
RECORD_STATUS	N/A	Char (50)	YES	"Modified"	User column for possible auditing purposes.
DATE_TIME_ON	N/A	Timestamp	YES	"2001-04-28 09:20:04.139999"	Timestamp signifying date this record became valid.
DATE_TIME_OFF	N/A	Timestamp	YES	"2001-04-28 09:20:04.139999"	Timestamp signifying date this record is no longer valid.

NLS_Error_Message_Table

Column Name	XML Tag Name	DataType	Nullable	Example Value	Description
MESSAGE_NUMBER	<Message_Number>	Char (9)	NO		Four digit long value used to identify where and why an erro occurred.
SECTION_NUMBER	<Section_Number>	Char (5)	NO		Used to maintain the proper sequence of text and values within a message.
MESSAGE_TEXT	<Message_Text>	Char(250)	NO		Used to describe the error that has occurred or to give supporting information to a value that is placed in the message.
SPACE_BEFORE	<Space_Before>	Integer	NO		The number of spaces to be placed in the message string before the textual portion of the message is added.
SPACE_AFTER	<Space_After>	Integer	NO		The number of spaces to be placed in the message string after the textual or value portion of the message is added.
TEXT_ONLY	<Text_Only>	Char(5)	NO		No numeric value will be assigned to this part of the message.

	RECORD_STATUS	N/A	Char (50)	NO	"Modified"	User column for possible auditing purposes.
	DATE_TIME_ON	N/A	Timestamp	YES	"2001-04-28 09:20:04.139999"	Timestamp signifying date this record became valid.
	DATE_TIME_OFF	N/A	Timestamp	YES	"2001-04-28 09:20:04.139999"	Timestamp signifying date this record is no longer valid.

Appendix C
Notices

This information was developed for products and services offered in the U.S.A. and Europe. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
500 Columbus Avenue
Thornwood, NY 10594
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 publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you. Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM United Kingdom Laboratories
Hursley Park
WINCHESTER, Hampshire
SO21 2JN
United Kingdom

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same

on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

The following terms are trademarks or services of IBM Corporation in the United States or other countries or both:

- IBM®
- MQSeries®
- DB2®
- AIX®
- Solaris®
- System 390®
- Netfinity®
- IAA®
- Insurance Application Architecture®

The following terms are trademarks or services of Microsoft Corporation in the United States or other countries or both:

- Windows®
- Windows NT®
- Internet Explorer®
- Microsoft Management Console®
- Active Directory Service Interfaces®

Acrobat Reader® is trademark of Adobe Systems in the United States, other countries, or both.

OAG is a trademark of the Open Architecture Group in the United States or other countries or both.

Other company, product, and service names may be trademarks or service marks of others.

Permission Statement

Glossary

This glossary defines 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.

A

ACORD
A US standards body. ACORD have produced the ACORD Insurance Service Business Message Specification for Property and Casualty.

Adapters
(1) A part that electrically or physically connects a device to a computer or to another device.

(2) A circuit board that adds function to a computer.

(3) Event Adapter: In a Tivoli environment, software that converts events into a format that the Tivoli Enterprise Console can use and forwards the events to the event server. Using the Tivoli Event Integration Facility, an organization can develop its own event adapters, tailored to its network environment and specific needs.

AIX: Advanced Interactive Executive
IBM's implementation of the UNIX operating system. The RS/6000 system, among others, runs the AIX operating system.

API: Application Programming Interface
(1) A software interface that enables applications to communicate with each other. An API is the set of programming language constructs or statements that can be coded in an application program to obtain the specific functions and services provided by an underlying operating system or service program.

(2) In VTAM, the language structure used in control blocks so that application programs can reference them and be identified to VTAM.

Attribute
(1) A characteristic that identifies and describes a managed object. The characteristic can be determined, and possibly changed, through operations on the managed object.
(2) Information within a managed object that is visible at the object boundary. An attribute has a type, which indicates the range of information given by the attribute, and a value, which is within that range.

C

CIIS: Client Information Integration Solution
An implementation of a Party Management System based on the IAA model.

Class
A UML class. A description of an object.

CRF: Cross Reference Function
This refers specifically to the storage system WMQI Enabler uses in order to keep track of creations of and attachments to UUID's.

D

DB2
An IBM relational database management system that is available as a licensed program on several operating systems. Programmers and users of DB2 can create, access, modify, and delete data in relational tables using a variety of interfaces.

DTD: 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 may 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.

E

EJB: Enterprise Java Bean

I

IAA: Insurance Application Architecture
IBM's business model for the insurance and financial services industry.

IAA-XML: Insurance Application Architecture-eXtensible Markup Language.
The Common Language used across integrated applications in MQSeries.

IFX: Interactive Financial eXchange
A cooperative industry effort among major financial institutions produced by the IFX Business Message Specification for the financial services industry.

L

LDAP: Lightweight Directory Access Protocol
An open protocol that (a) uses TCP/IP to provide access to directories that support an X.500 model and (b) does not incur the resource requirements of the more complex X.500 Directory Access Protocol (DAP). Applications that use LDAP (known as directory-enabled applications) can use the directory as a common data store and for retrieving information about

people or services, such as e-mail addresses, public keys, or service-specific configuration parameters. LDAP was originally specified in RFC 1777. LDAP version 3 is specified in RFC 2251, and the IETF continues work on additional standard functions. Some of the IETF-defined standard schemes for LDAP are found in RFC 2256.

M

MQMD: MQSeries Message Descriptor
The WebSphere MQ Integrator (WMQI) header that contains basic control information that must travel with the message.

O

OAG: Open Applications Group
A non-profit industry consortium comprised of many of the most prominent stakeholders in the business software component interoperability arena in the world. It was formed in February, 1995 in response to the rapidly expanding problem of tying disparate software applications together.

OASIS: Organization for the Advancement of Structured Information Standards
A non-profit international consortium founded in 1993 to advance the open interchange of documents and structured information objects. Originally focused on SGML, OASIS has evolved to more actively support XML.

Object
Instance of a class.

Operation
A function defined on a class and executable on the object.

P

Party
Any person or organization that the 'company' has, or had, or may have a business interest in.

Property
A data value of a type.

Property tag
An XML tag representing a property of an IAA type (represented as an UML attribute).

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

Root type
A type which is (one of) the main type(s) of an IAA component. The type to which types outside of the component are related to, and on which most of the types of the component depend on.

S

SQL: Structured Query Language
A programming language that is used to define and manipulate data in a relational database. It is often embedded in general purpose programming languages.

T

Tag
An XML construct <Tag....>.

Type tag
An XML tag representing an IAA type.

U

UML: Unified Modeling Language

W

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.

WMQI Enabler: WebSphere MQ Integrator Enabler
A complete scalable messaging and information integration add-on to the MQSeries family of products. Especially designed for the needs of the financial services industry, WebSphere MQ Integrator Enabler can integrate front-end systems with back-end systems using a hub/spoke architecture using XML as the common vocabulary across systems.

X

XML
eXtensible Markup Language. XML is a markup language for message definition, and is an open and public domain standard. XML is a subset of SGML designed for easy implementation in commercial and web environments.

XML attribute (or just attribute)
Appears in an opening tag, used to specify values in the tag. <Tag attribute='val'...>

Index

X