MQSeries Integrator for AS/400 and DB2/400, Version 1.1.1: Installation and configuration information

This document provides information about the software prerequisites, installation process, and configuration actions needed for the MQSeries Integrator for AS/400 and DB2/400 Version 1.1.1 product. It also provides information that will help you migrate from MQSeries Integrator for AS/400 and DB2/400 Version 1.1 (if you have that product installed). Refer to the chapter on the Installation Verification Procedure in the *MQSeries Integrator Installation and Configuration Guide* that is supplied with this product for further information about:

- Editing Rules daemon configuration files
- Run putdata
- Run getdata

Before you start

Before you start the installation process, make sure you have the following software installed on your AS/400 system:

- OS/400 V4R3 or later.
- MQSeries for AS/400 V4R2.1 (program number 5769MQ2), which is supplied with the MQSeries Integrator product. See the *MQSeries for AS/400 V4R2.1 Administration Guide* for information on how to install this product.
- The ILE C++ compiler (program number 5799GDW), so that you can create programs that use the MQSeries Integrator user exits.

You must install Client Access/400 on the client Windows NT workstation before you install the MQSeries Integrator client. Make sure you select the AS/400 Operations Navigator component (and its Database subcomponent).

In addition, we recommend that you set the AS/400 system security level to 40 or 50. To work with this setting, run the following command:

```
WRKSYSVAL SYSVAL(QSECURITY)
```

The documentation for the MQSeries Integrator and MQSeries for AS/400 products is provided as Portable Document Format (PDF) files on the MQSeries Integrator Administration GUI CD-ROM. Use the Adobe Acrobat Reader to view or print these files.

Migration issues

These migration issues only apply to AS/400 systems that already have the MQSeries Integrator software installed on them. You do not need to follow these steps if this is a new installation.

For MQSeries Integrator for AS/400 and DB2/400 Version 1.1.1, the metadata for MQSeries Integrator on AS/400 has been changed to match the implementation on OS/390. This change will impact some of the Formatter definitions contained in the MQSeries database. Those impacted definitions will need to be

updated to reflect the new metadata implementation. The following steps will update the MQSeries database with the new metadata:

1. Make a backup of the library that the MQSeries Integrator database resides in. It is extremely important that this step is done. The following commands should be used to backup the default collection that was shipped with the original code (the default collection resides in library QUSRMQSI):

   ```
   CRTSAVF FILE(MYLIB/BACKUPMQSI)
   SAVLIB LIB(QUSRMQSI) DEV(*SAVF) SAVF(MYLIB/BACKUPMQSI)
   ```

   Repeat this for each additional MQSeries Integrator database collection.

2. Following the instructions described in the installation process section of this document, install the software contained on the MQSeries Integrator for AS/400 and DB2/400 Version 1.1.1 CD-ROM. A new migration tool (DBUPDATE CL command) is shipped with this code. You will need this command to update the MQSeries Integrator MetaData definitions.

3. Issue the DBUPDATE command against the collection that contains the formats that use the old metadata. The DBUPDATE command takes an AS/400 collection (SQL library) and converts any datatypes stored in that collection to reflect the new metadata implementation.

   ```
   ADDLIBLE LIB(QMQSI)
   DBUPDATE COLLNAME(QUSRMQSI)
   ```

4. At this point, the collection identified as input to the DBUPDATE command is ready to be used with the MQSeries Integrator for AS/400 and DB2/400 Version 1.1.1 product code. By running the DBUPDATE command, you do not have to change any of your formats or your data.

Metadata changes

The metadata for MQSeries Integrator on AS/400 has been changed to match the implementation on OS/390. The String datatype represents an EBCDIC string. The Numeric datatype contains EBCDIC characters 0-9. The ASCII_Data datatype contains information encoded in the ASCII character set.

These metadata changes for the OS/400 platform might not be reflected on UNIX or Windows NT versions of MQSeries Integrator. This can affect your ability to import data from UNIX or Windows NT platforms to OS/400. These metadata changes are described in the following table:

Datatype as defined by Formatter	Description
String	String data is always in the native character set of the machine on which the Rules engine is running: ASCII on UNIX and Windows NT, and EBCDIC on OS/400.
Numeric	Numeric really means graphic characters 0-9. Numeric characters are represented in the native character set on which the Rules engine is running: ASCII on UNIX and Windows NT, and EBCDIC on OS/400.
ASCII Data	The ASCII_Data datatype contains information encoded in the ASCII character set.

If you export formats from an MQSeries Integrator for AS/400 and DB2/400 Version 1.1 database and later import them to a Version 1.1.1 database, these metadata changes could cause unexpected results.

You should use the Formatter GUI to review the imported data carefully with regard to the detail in the above table and in your MQSeries Integrator application.

See “Appendix A. Data Types” in the *MQSeries Integrator Version 1.1.1 System Management Guide* for further details. (The PDF file for this book is shipped on the MQSeries Integrator Administration GUI CD-ROM).

Installation process

In order to use the MQSeries Integrator product, you will need to have the MQSeries for AS/400 V4R2.1 product installed. At this time there are no checks in the MQSeries Integrator installation procedure to check if MQSeries for AS/400 is installed.

To install MQSeries Integrator, insert the MQSeries Integrator CD-ROM and run the following command:

```
RSTLICPGM LICPGM(5697F49) DEV(OPT01)
```

changing *OPT01* to the actual name of your CD-ROM drive.

Verifying the installation

To verify that the product was installed correctly:

1. Issue the Display Software Resources (DSPSFWRSC) command to display a list of the software products on your system.
2. Page down until you see 5697F49. There should be two entries for 5697F49 (one for the Base and one for the 2924 English language feature). You will see ERROR if the product is not installed correctly.
3. If there is an error, check the job log (using the DSPJOBLOG command) for errors that occurred during installation and take the necessary actions before repeating the installation procedure. The Install History Log, viewed by selecting option 50 from the Licensed Program Menu (GO LICPGM), may also provide useful information.

Where objects reside

Programs, service programs, and other OS/400 objects reside in the QMQSI system library, the size of which is 80 megabytes. The default SQL database that is supplied with the product resides in a library called QUSRMQSI (13 megabytes). MQSeries Integrator for AS/400 also has an IFS directory structure, as follows:

```
/QIBM/ProdData/Mqsi            Product Directory
/QIBM/ProdData/Mqsi/bin        Configuration/Connection Files
/QIBM/ProdData/Mqsi/examples   Sample formats, Configuration and
                                Connection Files, etc
/QIBM/ProdData/Mqsi/include    C++ files
```

Formatter and Rules client setup

The following steps are required for completing the installation of an MQSeries Integrator client in a Windows NT to AS/400 client-server configuration:

- Run the MQSeries Integrator client install program
- After the MQSeries Integrator software is installed on the client, verify that a Client Access connection is configured to the selected AS/400 server that the database collection will reside on. If one does not exist, configure a connection for a collection. For assistance see your Client Access documentation or online help.
- Configure an ODBC connection to the database collection. The following steps explain how to do this:
  1. Go to control panel, open up the ODBC Data Source Administrator.
  2. Under the User DSN tab, select Add.
  3. Select Client Access ODBC Driver (32-bit).
  4. Enter a Data source name on the General tab. This name can be any name you choose (try to make it descriptive).
  5. Select the System and a valid user ID for that system.
  6. Select the Server tab.
  7. For the default library, enter QUSRMQSI if you are going to use the default supplied SQL database collection, or a database collection library of your choice. Leave Commit mode to default.
  8. Select the Other tab.
  9. Under the Scrollable cursor, select Always Scrollable.

  The ODBC connection is completed.

Make sure that the DB2/400 SQL section of the PowerBuilder PBODB60.INI file contains the following lines:

```
PBSupportBindSelect='NO'
PBSupportBindUpdate='NO'
PBSupportDBBind='NO'
```

If it does not, use any text editor (such as WordPad or Notepad) to add the lines.

If you selected the default installation path when installing the client, the PBODB60.INI file is located in the mqi\gui directory.

Installing Visual Tester

When Visual Tester is installed, the installation program registers the OCXs for the selected database.

Visual Tester ships with OCXs compiled for different databases. To connect to a database other than the one selected at installation, you must register the proper OCXs (NNOBJS.OCX and NNMGRS.OCX) for the desired database type.

You can only configure Visual Tester to run on one database type at a time. For example, to switch from Oracle to Sybase, you must register the Sybase OCXs. After this is done, Visual Tester can only connect to Sybase until a new set of OCXs is registered.

This procedure is also in the Visual Tester Online Help.

To register OCXs:

1. In Windows Explorer, open the directory in which NNVT is installed, for example, C:\MQI\gui. This directory contains the following subdirectories:

MQSeries_Client	Remote MQSeries queuing
MQSeries_Server	Local MQSeries queuing
NEONetMQ	NEONet Messaging and Queuing
NoQueuing	No queuing type selected

2. Open the subdirectory for the type of queuing that you use. This subdirectory contains a directory for each database type, plus the user exit directory. The directories are:
   - db2
   - oracle
   - mssql
   - sybase
   - nnfexit
3. Open the directory for your database type. The database directory contains the following files:
   - NNFExit.dll
   - NNMgrLib.dll
   - NNMgrs.ocx
   - NNObjLib.dll
   - NNObjs.ocx
4. Double-click the NNMgrs.ocx file.
5. If the Open With dialog box appears:
   - If regsvr32.exe is in the list, select it, check the Always use this program to open this file checkbox, and click OK.
   - If regsvr32.exe is not in the list, click Other. A File Find dialog box appears. Open \winnt\system32 and select regsvr32.exe. Click Open. The Open With dialog box reappears with regsvr32.exe selected. Check the "Always use this program" box to open this file checkbox and click OK.
   - The OCX file is registered. A message box appears stating the registration succeeded. Click OK.
   - Double-click NNObjs.ocx. The OCX file is registered. A message box appears stating the registration succeeded. Click OK.

Deleting MQSeries Integrator

Before issuing the Delete License Program command, check to see if the group profile MQSIADMIN has group members. To do this, issue the command:

```
DSPUSRPRF MQSIADMIN TYPE(*GRPMBR)
```

The command returns a list of user profiles that contain this group profile (MQSIADMIN). Remove MQSIADMIN from each of these user profiles before continuing to delete MQSeries Integrator.

If the MQSeries Integrator directories are in use, issue the command:

```
RLSIFSLCK OBJ('/QIBM/ProdData/Mqsi/bin')
```

to free the directory.

Also, check to see if any users are locking the QMQSI library (because it might be in their library list), by issuing the command:

```
WRKOBJLCK OBJ(QMQSI) OBJTYPE(*LIB)
```

To delete MQSeries Integrator, run the Delete License Program command:

```
DLTLICPGM LICPGM(5697F49)
```

Deleting MQSeries Integrator removes the following from your AS/400 system:

- The QMQSI library
- The /QIBM/Proddata/Mqsi directory
- The MQSIADMIN and MQSIUSER user profiles
- The software resource entry

The default database collection library QUSRMQSI is **not** deleted (because it might contain user data).

Note: You can reinstall the product after you have deleted it. The installation process recognizes that the QUSRMQSI library exists and does not try to overwrite it.

MQSIMAIN menu

The AS/400 menu MQSIMAIN is created to make MQSeries Integrator usage easier. Run the following commands to access this menu:

```
ADDLIBLE LIB(QMQSI)
GO MENU(MQSIMAIN)
```

You will see the following screen:

```
MQSIMAIN           MQSeries Integrator Main Menu
                                                    System:  MYSYSTEM
Select one of the following:

     1.  Start the MQSeries Integrator Daemon               MQSIRULENG
     2.  Put a message to a queue                           MQSIPUTDTA
     3.  Get a message from a queue                         MQSIGETDTA
     4.  Encrypt the password in daemon file                MQSIENCRYP

    11.  Test an input control                              CALL APITEST
    12.  Test an input control/output control combination   CALL MSGTEST
    13.  Encrypt the password in a sqlsvses.cfg file        CALL NNCRYPTCFG
    14.  Import/Export formats                              CALL NNFIE
    15.  Import/Export rules                                CALL NNRIE
    16.  Test a message against a specific rule             CALL NNRTRACE
    17.  Test which rules are hit by a message              CALL RULETEST

    21.  Install a new database instance                    DBINSTALL


Selection
===> _____________________________________________________

F3=Exit  F4=Prompt  F9=Retrieve  F12=Cancel
```

Reading stream files

Before using MQSeries Integrator applications, it is important to understand how stream files can be opened and read. An AS/400 stream file has a code page associated with the data in the file. Type the following to see the code page:

```
DSPLNK OBJ('<file_name>')
```

Then choose Option 8 - Display Attributes.

Some applications read stream files in binary mode. This means that the code page of the file is ignored when reading in the file. The bytes contained in the file are read in exactly as they exist in the file. Some applications read stream files in text mode. This means that the operating system performs translation when the data is read in. The data is translated from the code page of the stream file to the code page of the job (from the CCSID of the job). With MQSeries Integrator, some applications read stream files in text mode, while others read stream files in binary mode. See “Using commands and programs” on page 10 for further details.

Database connection configuration files

Some MQSeries Integrator applications use the sqlsvses.cfg file for retrieving information for database connectivity. Others use a file with a .mpf extension. These files must all reside in IFS. There are example template versions of these files in the /QIBM/ProdData/MQSI/bin directory. These files are read in text mode.

The sqlsvses.cfg file

The sqlsvses.cfg file controls access to the SQL database and is used by several applications, including APITEST, MSGTEST, NNRIE, NNFIE, and RULETEST. The file must reside in the current directory of the AS/400 job that is running the application (for example, your AS/400 display session).

Although DB2/400 is classified as a DB2 database, MQSeries Integrator for AS/400 references it in a similar manner to Microsoft SQL/Sybase.

A typical entry in this file could look like this:

```
New_format_demo:<Server>:<Userid>:<Password>:<Instance>
```

where:

<Server>
Specifies the SQL database. You can get this value by using the WRKRDBDIRE command and finding the entry with the Remote Location Name of *LOCAL. Typically, this is the same name as the AS/400 system. If there is no *LOCAL entry, you must add one.

<Userid>
Specifies the user profile to connect with.

<Password>
Specifies the password associated with the given user profile.

<Instance>
Specifies the name of the database collection. This is the name of the library that contains the SQL database. The collection that is created on installation is called QUSRMQSI.

MQSIruleng.mpf

The MQSIruleng.mpf file is used for the MQSIRULENG program (the main MQSeries Integrator daemon). This file is specified as a parameter when starting MQSIRULENG.

Note these entries in MQSIruleng.mpf:

QueueManagerName
The queue manager on the AS/400 that you will be using.

InputQueueName
The MQSeries queue from which MQSeries Integrator for AS/400 will read messages.

NoHitQueueName
The MQSeries queue on which MQSeries Integrator for AS/400 will place messages when they do not match any rules.

FailureQueueName
The MQSeries queue on which MQSeries Integrator for AS/400 will place messages when rule processing failures occur.

DefaultAppGroup
The Default Application Group containing your rules (named when you define your rules).

DefaultMsgType
The Default Message Type containing your formats (named when you define your formats).

LogLevel
Ideally, set to 0 (all logging) when testing your rules. You can then change it to 2 or 3 (to save disk space) when your system goes live.

ServerName = <data_source>
Specifies the data source. You can get this value by using the WRKRDBDIRE command and finding the entry with the Remote Location of *LOCAL. Typically, this will be the same name as the AS/400 system. If there is no *LOCAL entry, you must add one.

UserId = <user>
Specifies the user ID to connect with.

Password = <password>
Specifies the password to connect with.

DatabaseInstance = <database collection>
Specifies the name of the database collection. The collection that is created on the installation process is QUSRMQSI.

DatabaseType = 5
Specifies a DB2 database.

Setting environment variables

There are two ways to set an environment variable on the AS/400.

1. Using the 'export' command from within the QSHELL environment.
2. Using the WRKENVVAR, ADDENVVAR, CHGENVVAR and RMVENVVAR CL commands. Restriction: You must have *JOBCTL special authority to use these commands. This is the recommended method.

In OS/400 V4R3M0, any environment variables apply to the job that set them and will be lost when that job ends (for example, signing off from your workstation). If you want to 'keep' environment variables, you should set them in your initial program for interactive sessions, or ensure that they are set by any submitted jobs. In OS/400 V4R4M0 you have the option of specifying LEVEL(*SYS) on the ADDENVVAR command that sets the environment variable permanently and makes it available to all jobs.

Alert messages

CL command:

```
ADDENVVAR ENVVAR(NN_ALERT_FILE_NAME) VALUE=(NNStatus.Log)
```

QSHELL command:

```
export NN_ALERT_FILE_NAME=NNStatus.Log
```

Sets the name of the file that holds the Alert messages. The default is NEONetStatusLog (which will be in the existing current directory).

Alert error reporting

CL command:

```
ADDENVVAR ENVVAR(NN_ALERT) VALUE='[ALL|CRITICAL|OFF]'
```

QSHELL command:

```
export NN_ALERT=[ALL|CRITICAL|OFF]
```

Sets or disables the Alert error reporting mechanism. If you specify CRITICAL, only critical errors are reported. No errors are reported if you specify OFF. The default is ALL.

Time display

CL command:

```
ADDENVVAR ENVVAR (NN_LOCAL_TIME) VALUE='[GMT|TRUE]'
```

QSHELL command: export NN_LOCAL_TIME=[GMT|TRUE]

Sets the time display to local time or GMT for message reporting mechanisms. The default is GMT.

Line Feed or Carriage Return-Line Feed

CL command:

```
ADDENVVAR ENVVAR (MQSI_STRIP_CRLF) VALUE('N')
```

QSHELL command: export MQSI_STRIP_CRLF=NO

The programs APITEST, MSGTEST, NNRTRACE, RULETEST, and MQSIPUTDATA take a file that contains a message as input. By default on the AS/400 system, these programs strip off any Line Feed (LF) or Carriage Return-Line Feed (CRLF) combinations from the end of input messages. If you do not want to have the LF or CRLF stripped off the end of messages, set the MQSI_STRIP_CRLF environment variable to a value of 'N'.

Using commands and programs

The following describes the usage of MQSeries Integrator programs on AS/400. The intent is not to describe each program in full detail, but to provide AS/400-specific information for these programs.

MQSIRULENG - Start the MQSI daemon

MQSIRULENG is a command. It is the equivalent of the MQSIruleng program on other platforms.

The command has a PARMFILE parameter in which an MQSIruleng.mpf file should be specified. The

MQSIruleng.mpf file is read in text mode. See “Database connection configuration files” on page 7 for more information on this file.

Command syntax

```
MQSIRULENG PARMFILE('parm_file_path_name')
```

where PARMFILE is the name of the parameter file to be used. See the section called "Edit the Rules Daemon Configuration Files" in the Installation Verification Procedure chapter in the *MQSeries Integrator V1.1 Installation and Configuration Guide* for details of the contents of this file.

Example invocation

```
MQSIRULENG PARMFILE('/QIBM/ProdData/MQSI/bin/MQSIruleng.mpf')
```

Shut down the Rules daemon

To shut down the Rules daemon:

- Add the following to the [Put Options] section of the MQSIputdata.mpf file:

  ```
  OPT_SHUTDOWN = SHUTDOWN
  ```

  and use an empty (null) message.

  See the section called *Shutdown Messages* in the *MQSeries Integrator V1.1 System Management Guide* for further details of this option.

- Alternatively, you can issue the following command:

  ```
  ENDSBS SBS(QMQSI1) OPTION(*IMMED)
  ```

- Or, you can specify an input queue with get inhibited, as follows:

  ```
  CHGMQMQ QNAME(MQSI_INPUT_QUEUE) GETENBL(*NO)
  ```

  Change GETENBL to (*YES) if you later want to get enable the input queue.

MQSIPUTDTA - Put a message to a queue

MQSIPUTDTA is a command. It is the equivalent of the MQSIputdata program on other platforms.

The command has a PARMFILE parameter in which an MQSIputdata.mpf file should be specified. The MQSIputdata.mpf file is read in text mode.

Most of the parameters in the MQSIputdata.mpf file are self-explanatory but you consider the following:

OPT_APP_GRP
: This is the Application Group that contains your rules. It is named when you define your rules.

OPT_MSG_TYPE
: This is the Message Type that contains your formats. It is named when you define your formats.

The MQSIputdata.mpf file should contain a line specifying a value for inputFileName. This inputFileName should be a stream file in IFS. This file is read in binary mode.

Command syntax

```
MQSIPUTDTA PARMFILE('parm_file_path_name')
```

where PARMFILE is the name of the parameter file to be used. See the section called "Run putdata" in the Installation Verification Procedure chapter in the *MQSeries Integrator V1.1 Installation and Configuration Guide* for details of the contents of this file.

Example invocation

```
MQSIPUTDTA PARMFILE('/QIBM/ProdData/MQSI/bin/MQSIputdata.mpf')
```

MQSIGETDTA - Get a message from a queue

MQSIGETDTA is a command. It is the equivalent of the MQSIgetdata program on other platforms.

The command has a PARMFILE parameter in which an MQSIgetdata.mpf file should be specified. The MQSIgetdata.mpf file is read in text mode.

The MQSIgetdata.mpf file should contain a line specifying a value for outputFileName. This outputFileName should be a stream file in IFS. This file is written in binary mode.

Command syntax

```
MQSIGETDTA PARMFILE('parm_file_path_name')
```

where PARMFILE is the name of the parameter file to be used. See the section called "Run getdata" in the Installation Verification Procedure chapter in the *MQSeries Integrator V1.1 Installation and Configuration Guide* for details of the contents of this file.

Example invocation

```
MQSIGETDTA PARMFILE('/QIBM/ProdData/MQSI/bin/MQSIgetdata.mpf')
```

MQSIENCRYP - Encrypt a configuration file

MQSIENCRYP is a command. It is the equivalent of the MQSIencrypt program on other platforms.

The command encrypts the Password field of an MQSIruleng.mpf stream file.

The command has an INFILE parameter and an OUTFILE parameter. INFILE is the input MQSIruleng.mpf file. OUTFILE is the new, generated MQSIruleng.mpf file that contains the encrypted password.

The input file is read in text mode. The output file is written in text mode.

Command syntax

```
MQSIENCRYP INFILE('input_file_path_name') OUTFILE('output_file_path_name')
```

where INFILE is the name of the input MQSIruleng.mpf file.

Example invocation

```
MQSIENCRYP INFILE('/QIBM/ProdData/MQSI/bin/MQSIruleng.mpf')
 OUTFILE('/QIBM/ProdData/MQSI/bin/MQSIruleng.mpf.encrypted')
```

APITEST

APITEST is a program. It is the equivalent of the apitest program on other platforms.

The sqlsvses.cfg file must reside in the current directory.

APITEST prompts the user for the name of an input file (containing a message) and an input format. This file is read in binary mode.

Example invocations

```
CALL PGM(APITEST)
CALL PGM(APITEST) PARM('-h')   (Lists possible parameters for this program)
```

MSGTEST

MSGTEST is a program. It is the equivalent of the msgtest program on other platforms.

The sqlsvses.cfg file must reside in the current directory.

MSGTEST prompts the user for the name of an input file (containing a message). This file is read in binary mode.

MSGTEST prompts the user for the name of an output file (where a new message will be written). This file is written in binary mode.

Example invocations

```
CALL PGM(MSGTEST)
CALL PGM(MSGTEST) PARM('-h')   (Lists possible parameters for this program)
```

NNCRYPTCFG - Encrypt the SQL Services configuration file

NNCRYPTCFG is a program. It is the equivalent of the NNcryptCfg program on other platforms.

The command encrypts the Password fields of an sqlsvses.cfg stream file.

The command has an '-inputFile' parameter and a '-outputFile' parameter. '-inputFile' is the input sqlsvses.cfg file. '-outputFile' is the new, generated sqlsvses.crypt that contains the encrypted passwords.

The input file is read in text mode. The output file is written in text mode.

MQSeries Integrator programs that refer to the sqlsvses.cfg file check for the existence of the sqlsvses.crypt file and, if it exists in the current directory, will use this instead of the sqlsvses.cfg file.

Refer to the information on encrypting the sqlsvses.cfg file in the *MQSeries Integrator V1.1 Installation and Configuration Guide* for further details.

An example invocation

Note: The parameter values are case sensitive so you must enter them as shown or the command will not work.

```
CALL PGM(NNCRYPTCFG) PARM('-inputFile' '/QIBM/ProdData/MQSI/bin/sqlsvses.cfg'
 '-outputFile' '/QIBM/ProdData/MQSI/bin/sqlsvses.crypt')
```

NNFIE - Formats import/export utility

NNFIE is a program. It is the equivalent of the NNFie program on other platforms.

The sqlsvses.cfg file must reside in the current directory.

On an import, NNFIE takes an input file name as a parameter (containing formats to be imported). This file should be a stream file in the IFS. This file is read in text mode.

On an export, NNFIE takes an output file name as a parameter (to place exported formats). This file should be a stream file in the IFS. This file is written in text mode.

When moving formats from System A to System B, the process is as follows:

1. On System A, run NNFIE with the -e option to export the formats to a stream file.
2. Using FTP, move the stream file (in text or ASCII mode) from System A to System B.
3. On System B, run NNFIE with the -i option to import the formats into a database collection.

There are times when it would be useful to import the literals from another system that uses the ASCII character set and have those literals converted to the native character set for the AS/400, which is EBCDIC. There is an option on the AS/400 that is only valid when importing formats. By specifying this option, `'-cvtAscLitToEbcdic'` (convert ASCII Literals to EBCDIC), all literal values will be converted from the ASCII character set to the EBCDIC character set. NNFIE assumes that all literals are in ASCII when this option is specified. If you have literals of both character sets and want your ASCII literals to be translated, you must move them to a separate file which contains only ASCII literals and then specify the `'-cvtAscLitToEbcdic'` option when importing that file. Do not specify this option if you have literals that are in the EBCDIC character set.

Example invocations

Note: The parameter values are case sensitive so you must enter them as shown or the command will not work.

```
List possible parameters
  CALL PGM(NNFIE) PARM('-h')

Import your formats
  CALL PGM(NNFIE) PARM('-i' 'myFormats' '-s' 'SessionName')

Export your formats
  CALL PGM(NNFIE) PARM('-e' 'myFormats' '-s' 'SessionName')
```

where SessionName is the “tag” associated with the entry in the sqlsvses.cfg file that you want to use. For example, where the SessionName is 'Import':

```
  Import:MYAS400:MYUSERID:MYPASSWORD:QUSRMQSI
```

NNRIE - Rules import/export utility

NNRIE is a program. It is the equivalent of the NNRie program on other platforms.

The sqlsvses.cfg file must reside in the current directory.

On an import, NNRIE takes an input file name as a parameter (containing rules to be imported). This file should be a stream file in the IFS. This file is read in text mode.

On an export, NNRIE takes an output file name as a parameter (to place exported rules). This file should be a stream file in the IFS. This file is written in text mode.

When moving rules from System A to System B, the process is as follows:

1. On System A, run NNRIE with the -e option to export the rules to a stream file.
2. Using FTP, move the stream file (in text or ASCII mode) from System A to System B.
3. On System B, run NNRIE with the -i option to import the rules into a database collection.

Example invocations

Note: The parameter values are case sensitive so you must enter them as shown or the command will not work.

```
List possible parameters
  CALL PGM(NNRIE)

Import your rules
  CALL PGM(NNRIE) PARM('-i' 'myRules' '-s' 'SessionName')

Export your rules
  CALL PGM(NNFIE) PARM('-e' 'myRules' '-s' 'SessionName')
```

where SessionName is the “tag” associated with the entry in the sqlsvses.cfg file that you want to use. For example, where the SessionName is 'Import':

```
Import:MYAS400:MYUSERID:MYPASSWORD:QUSRMQSI
```

NNRTRACE - Rules trace utility

NNRTRACE is a program. It is the equivalent of the NNRTrace program on other platforms.

The program has an 'Input-File-Name' parameter. This file should be a stream file residing in IFS. It is read in binary mode.

Example invocations

Note: The parameter values are case sensitive so you must enter them as shown or the command will not work.

```
CALL PGM(NNRTRACE) PARM('-i' 'myInFile' '-a' 'myAppGroup' '-m' 'myMsgType' '-r' 'myRuleName')
CALL PGM(NNRTRACE) PARM('-h')  (Lists possible parameters for this program)
```

RULETEST

RULETEST is a program. It is the equivalent of the ruletest program on other platforms.

The sqlsvses.cfg file must reside in the current directory.

The program has an 'Input-File-Name' parameter. This file should be a stream file residing in IFS. It is read in binary mode.

Example invocations

Note: The parameter values are case sensitive so you must enter them as shown or the command will not work.

```
CALL PGM(RULETEST) PARM('-i' 'myInFile' '-a' 'myAppGroup' '-m' 'myMsgType')
CALL PGM(RULETEST) PARM('-h')
            (Lists possible parameters for this program)
```

DBINSTALL - Create an MQSI database collection

DBINSTALL is a command. It creates an AS/400 collection SQL Library and populates it with default rules and formats.

Notes:

1. When installing MQSeries Integrator on an AS/400 system, a collection called QUSRMQSI is automatically created. You need use the DBINSTALL command only if you want additional collections.
2. If you have run other MQSeries Integrator programs or commands, we suggest that you sign off and then sign on again after using the DBINSTALL command, to reset the SQL environment.

Command syntax

```
DBINSTALL COLLNAME(collection_name) REPLACE(*NO|*YES)
```

where:

COLLNAME Is the name of the collection to be created.

REPLACE Valid values for this parameter are *NO (the default) and *YES. Specifying *YES deletes a library with name of COLLNAME if it exists, before creating the collection.

Note: *YES deletes **any** library, not just a collection, with this name.

Example invocation

```
DBINSTALL COLLNAME(MYDATA) REPLACE(*YES)
```

FORMATCC - Format consistency checker

FORMATCC is a command. It runs a report against the formats in the database and looks for situations that could be a potential problem.

Command syntax

```
FORMATCC COLLECTION(collection_name) DATASOURCE(datasource_name)
 FILE(*DFT|file_name) LOGFILE(*DISPLAY|file_name)
```

where:

COLLECTION Specifies the name of the database collection to access.

DATASOURCE Specifies the data source to connect to. To determine the data source, use the WRKRDBDIRE command and find the Relational Database at location *LOCAL. This parameter is case sensitive.

FILE Specifies the path of the SQL file (a file in IFS) to use for consistency checking. Specifying *DFT causes the default file /QIBM/ProdData/Mqsi/bin/formatcc.sql to be used.

LOGFILE Specifies the name of an IFS file that will contain the results of the consistency checker. If you specify *DISPLAY, the results are sent to standard output.

Example invocation

```
FORMATTCC COLLECTION(QUSRMQSI) DATASOURCE(MYDATASOURCE) FILE(*DFT) LOGFILE(*DISPLAY)
```

Note: If you have run other MQSeries Integrator programs or commands, we suggest that you sign off and then sign on again after using the FORMATCC command, to reset the SQL environment.

RULECC - Rules consistency checker

RULECC is a command. It runs a report against the rules in the database and looks for situations that could be a potential problem.

Command syntax

```
RULECC COLLECTION(collection_name) DATASOURCE(datasource_name)
 FILE(*DFT|file_name) LOGFILE(*DISPLAY|file_name)
```

where:

COLLECTION Specifies the name of the database collection to access.

DATASOURCE Specifies the data source to connect to. To determine the data source, use the WRKRDBDIRE command and find the Relational Database at location *LOCAL. This parameter is case sensitive.

FILE Specifies the path of the SQL file (a file in IFS) to use for consistency checking. Specifying *DFT causes the default file /QIBM/ProdData/Mqsi/bin/rulecc.sql to be used.

LOGFILE Specifies the name of an IFS file that will contain the results of the consistency checker. If you specify *DISPLAY, the results are sent to standard output.

Example invocation

```
RULECC COLLECTION(QUSRMQSI) DATASOURCE(MYDATASOURCE) FILE(*DFT) LOGFILE(*DISPLAY)
```

Note: If you have run other MQSeries Integrator programs or commands, we suggest that you sign off and then sign on again after using the RULECC command, to reset the SQL environment.

Work management objects

A subsystem description is created by the install process. The subsystem is used to run the MQSeries Integrator product.

The following AS/400 work management objects are created in the installation process.

- A subsystem description named QMQSI1.
- A class object called QMQSI1. This has a run priority of 20.
- A job queue entry that will connects the job queue to the subsystem description. This job queue entry has 2 for the maximum number of jobs.
- A routing entry for the subsystem description using our QMQSI1 class.
- A job description called QMQSI1.

The work management objects reside in library QMQSI. The objects can be changed by the MQSeries Integrator administrator user using the relevant OS/400 commands. You are strongly advised not to change these objects unless you are fully aware of OS/400 work management principles.

User exits

The NEON Formatter provides a wide range of functionality, but occasionally a user has a need to add functionality to customize the Formatter. It is for this purpose that the Formatter supports user exits.

The Formatter is shipped with a service program (*SRVPGM) object named USEREXIT. This is a stub service program in the sense that it contains symbols that resolve the user exit lookup function NNGetUserExitFuncPtrs(), although this function does nothing as shipped. If the user wishes to define an exit function, the service program must be replaced by one that the user has made. The new service program must contain a lookup function by the same name, defined so that when the name of the user exit function is passed in, the lookup function returns a pointer to the user-defined function.

Creating and using a new user exit library

In order to implement your own user exit, you must do the following:

1. Create a module (or modules) that will contain the customized functions. An example module is in /QIBM/ ProdData/Mqsi/examples/userexit.cpp.
2. Create a new USEREXIT service program (*SRVPGM). It is recommended that you create the user exit service program in a library other than QMQSI.
3. Before performing the next step, make sure that the MQSeries Integrator Rules Daemon is shut down. There are several ways of doing this, such as issuing the end subsystem (ENDSBS) command against the QMQSI1 subsystem.
4. Update the MQSI Rules/Formatter service program (RULESFMT) in the QMQSI library to point to the new user exit service program instead of the default in the QMQSI library.

To illustrate how these steps might be accomplished, here is an example of how to use the sample user exit. This example assumes that the user exit service program will be created in a library named UELIB.

1. Create the module:

```
CRTCPPMOD MODULE(UELIB/USEREXIT) SRCSTMF('/QIBM/ProdData/Mqsi/examples/userexit.cpp')
 DBGVIEW(*ALL) DEFINE(*NONE) INCDIR('/QIBM/ProdData/Mqsi/include')
```

2. Create the service program:

```
CRTSRVPGM SRVPGM(UELIB/USEREXIT) MODULE(UELIB/USEREXIT) EXPORT(*ALL) BNDSRVPGM(QMQSI/RULESFMT)
 OPTION(*UNRSLVREF) ALWLIBUPD(*YES)
```

3. Shut down the Rules engine:

```
ENDSBS SBS(QMQSI1) OPTION(*IMMED)
```

4. Update the Rules/Formatter service program:

```
UPDSRVPGM SRVPGM(QMQSI/RULESFMT) MODULE(*NONE) BNDSRVPGM(UELIB/USEREXIT) SRVPGMLIB(UELIB)
 OPTION(*UNRSLVREF)
```

When issuing these commands you may use your own library or object names, but make sure that you specify all the options provided in this example. Failure to do so may result in improper linkage to the new user exit service program.

For more information about user exits refer to the *MQSeries Integrator Programming Reference for NEONFormatter*.

```
Example user exit module:
//
// This example demonstrates the use of User Exits.
//
// The intent of this example is to perform a simple conversion of length
// measurements from English to Metric units. Input messages are in the form:
//
//  <value> <space> <unit> <newline>
//
// where <value> is a numeric decimal number, <space> is a single blank
// space (hex 20) and <unit> can be one of the literals "inch", "foot",
// "yard", or "mile". The requirement is to convert measurements from inches,
// feet, yards or miles to centimeters, decimeters, meters or kilometers
// respectively. Thus, given an input message like the following:
//
//  37.35 inch
//
// the reformatted message should be:
//
//  94.869000 centimeter
//
// The input and output formats to accomplish this are trivial but
// illustrate the point. The output controls for the value
// and unit fields will be implemented as User Exits which will invoke
// the functions UE_ConvertValue and UE_ConvertUnit, defined below.
//
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <iostream.h>

#include "nnexit.h"
#include "neodefs.h"
#include "dbtypes.h"
#include "interface.h"
#include "sqlapi.h"
#include "formatter.h"


NNExitRet
UE_ConvertUnit(const DbmsSession &rSession, const NNParsedFields &rFields)
{
      NNExitRet oER;
      const char *f=rFields.GetFieldAscii("unit", 0);
      char* unit = "unknown";

      if ( strcmp(f,"inch") == 0 )
        unit = "centimeter";
      else if ( strcmp(f,"foot") == 0 )
        unit = "decimeter";
      else if ( strcmp(f,"yard") == 0 )
        unit = "meter";
```

```
|       else if ( strcmp(f,"mile") == 0 )
|         unit = "kilometer";

|   oER.SetByteArrayValue(unit, strlen(unit));

|   return oER;

|  }


|  NNExitRet
|  UE_ConvertValue(const DbmsSession &rSession, const NNParsedFields &rFields)
|  {
|       NNExitRet oER;
|       const char *f=rFields.GetFieldAscii("unit", 0);
|       const char *v=rFields.GetFieldAscii("value", 0);
|       double factor = 1.0;

|       if ( strcmp(f,"inch") == 0 )
|              factor = 2.54;
|       else if ( strcmp(f,"foot") == 0 )
|              factor = 3.048;
|       else if ( strcmp(f,"yard") == 0 )
|              factor = 0.9144;
|       else if ( strcmp(f,"mile") == 0 )
|              factor = 1.609;

|   oER = atof(v) * factor;

|   return oER;
|  }

|  extern "C" void
|  NNGetUserExitFuncPtrs(
|      char*       acFuncName,
|      NN_EXIT_FUNC_t            &rUEptr,
|      NN_EXIT_CLEANUP_FUNC_t    &rUEClUpPtr) {

|  if(strcmp(acFuncName, "UE_ConvertUnit") == 0) {
|         rUEptr = UE_ConvertUnit;
|         rUEClUpPtr = NULL;
|  }
|  else if(strcmp(acFuncName, "UE_ConvertValue") == 0) {
|         rUEptr = UE_ConvertValue;
|         rUEClUpPtr = NULL;
|  }
|  else {
|         rUEptr = NULL;
|         rUEClUpPtr = NULL;
|  }
|  }
```

Data descriptions should be the same as on UNIX or Windows NT platforms.

Tracing

Here are some steps you should take to trace the SQL statements that are executed in any MQSeries Integrator program.

- You will need to have the AS/400 system in a dedicated state. All SQL statements executed on the system are traced.
- Run the command:

  ```
  STRDBMON OUTFILE(QGPL/OUTPUT) JOB(*ALL) TYPE(*DETAIL)
  ```

- Run the program that you want to trace.
- Run the command:

  ```
  ENDDBMON JOB(*ALL)
  ```

- To inspect the output, look at the QGPL/OUTPUT file with the command:

  ```
  DSPPFM FILE(QGPL/OUTPUT)
  ```

  Scroll several pages to the right (using PF20) to see the SQL statements that were executed.

Note: Running STRDBMON for only the job that is executing the MQSeries Integrator program does not trace the required SQL statements. Database access is traced in a different job; you therefore need to monitor all jobs by using the STRDBMON command with the JOB(*ALL) parameter.

System authorities on MQSeries Integrator objects

Two user profiles are created automatically by the installation process if they do not already exist.

MQSIADMIN This profile is created with user class SECOFR. It is not assigned a password. This profile is given *ALL authority to all objects in the object library and all collections created with DBINSTALL.

MQSIUSER This profile is created with user class USER. It is not assigned a password. This profile is given *USE authority to all objects in the object library. This profile is given *CHANGE authority to all collections created with DBINSTALL.

These two user profiles can be used as group profiles to control access to MQSeries Integrator.

For example, to add user JOE to the MQSIUSER group, enter the command:

```
CHGUSRPRF USRPRF(JOE) GRPPRF(MQSIUSER)
```

For example, to add user MIKE to the MQSIADMIN group so that he can administer the MQSeries Integrator product, enter the command:

```
CHGUSRPRF USRPRF(MIKE) GRPPRF(MQSIADMIN)
```

AS/400, DB2, IBM, MQSeries, and OS/400 are trademarks of the IBM Corporation in the United States or other countries, or both. Windows NT is a registered trademark of Microsoft Corporation.

GC34-5646-01