MQSeries Integrator – GetSQLCode Plug-In Version 1.1

11th September 2002

Mike Brady
Senior Consulting IT Specialist
IBM
Australia

mjbrady@au1.ibm.com

Take Note!

Before using this report be sure to read the general information under "Notices".

Second Edition, September 2002

This edition applies to Version 1.1 of *MQSeries Integrator – GetSQLCode Plug-In* and to all subsequent releases and modifications unless otherwise indicated in new editions.

Table of Contents

Notices

The following paragraph does not apply in any 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 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.

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates.

Any reference to an IBM licensed program or other IBM product in this publication is not intended to state or imply that only IBM's program or other product may be used. Any functionally equivalent program that does not infringe any of the intellectual property rights may be used instead of the IBM product.

Evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, is the user's responsibility.

IBM may have patents or pending patent applications covering subject matter 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 the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, New York 10594, USA.

The information contained in this document has not be submitted to any formal IBM test and is distributed AS-IS. The use of the information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item has been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Trademarks and service marks

The following terms, used in this publication, are trademarks of the IBM Corporation in the United States or other countries or both:

- AIX
- IBM
- MQSeries
- MQSeries Integrator
- MQSI

The following terms are trademarks of other companies:

• HP-UX	Hewlett Packard Corporation
• Solaris	Sun Corporation
• Windows NT	Microsoft Corporation

Summary of Amendments

Date	Changes
14th July 2001	Initial internal release
11th August 2001	Added support for HP-UX platform
3rd September 2001	Release as a SupportPac
11th September 2002	• Add preprocessor directives to allow source to build correctly against v2.1, v2.0.2 and v2.0.1 releases • Shipped v2.0.1 and post v2.0.1 binaries

Preface

When building message flows with MQSeries Integrator V2 that incorporate database nodes, it is often desirable to be able to interrogate the SQL return code following a failure in order to control subsequent logic. One way of doing this is to use a compute node and/or a filter node to parse the message Exception List, however this involves a degree of fairly complex ESQL that is unlikely to be within the ability of the average MQSI programmer.

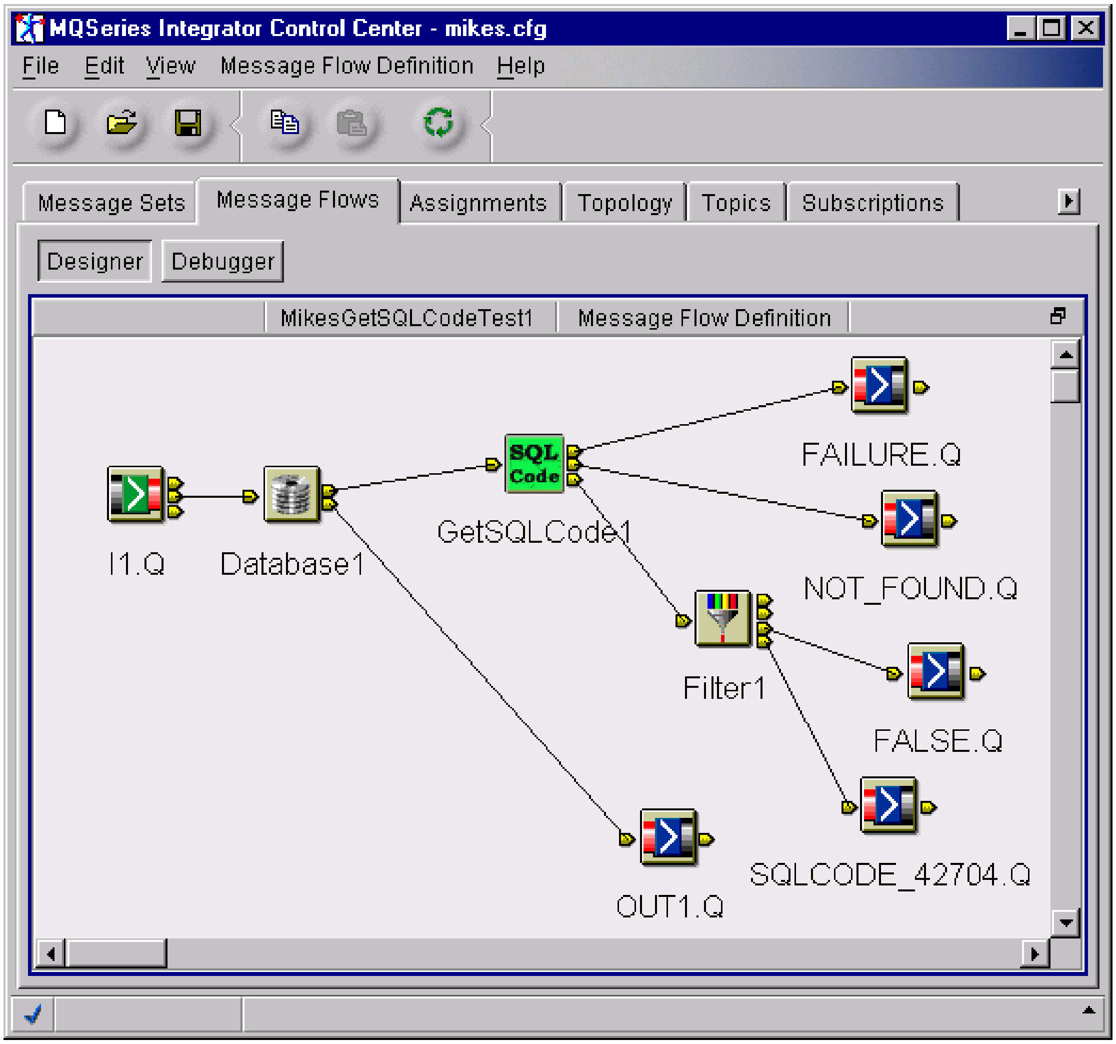
The MQSI V2 GetSQLCode function provides an alternative mechanism for extracting the SQLCODE value. If an exception containing a SQLCODE value exists in the message being processed, the value is extracted and stored in the message at a location specified by the message flow programmer. The GetSQLCode implementation is thread-safe and will work equally well for deployments where Message Flows have been configured with multiple worker threads.

Chapter 1. Overview

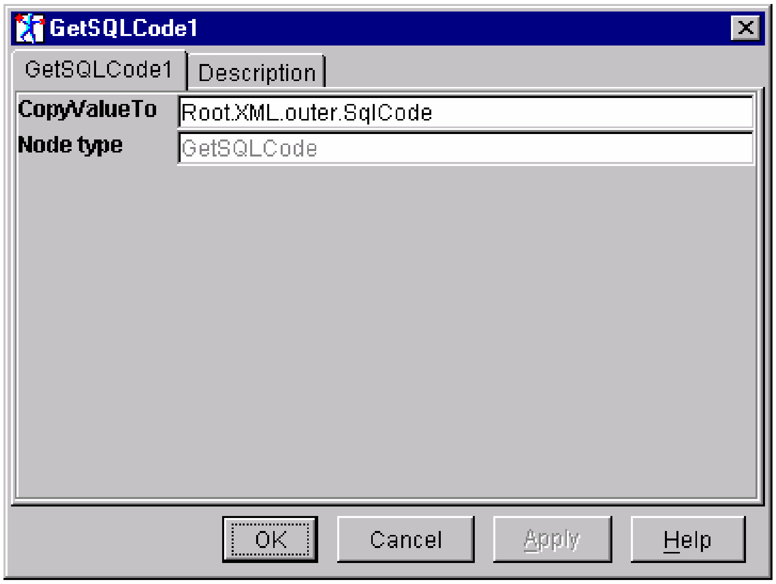
The GetSQLCode node is responsible for extracting the SQLCODE value from the current message's Exception List. If the SQLCODE value is successfully located and extracted, a copy of the input message is made and the value is copied into the new message at the specified syntax element location.

The following simple message flow shows an instance of a GetSQLCode node being used. Here, a database node has its failure terminal wired to an instance of the GetSQLCode node. In this example, assuming the SQLCODE value is extracted ok, the modified message (containing the value) is then propagated to the *out* terminal. If the input message does not contain a SQLCODE value, the message is propagated to the *notFound* terminal (shown in the example wired to the NOT_FOUND.Q). Any other errors result in the original message being propagated to the 'failure' terminal and a message being logged to the system log.

The Filter node is shown as an example of the type of processing that could be carried out on the message containing the extracted SQLCODE. For example, different logic paths might be taken depending on the setting of the SQLCODE.



The properties window for the GetSQLCode node (GetSQLCode1) is shown in the following image. The CopyValueTo attribute specifies the message location into which the extracted SQLCODE value should be stored. In this case, the SQLCODE will be stored in the XML body at the location referred to by ‘outer.SqlCode’. If the location does not exist, a syntax element is created with an element type of NAME-VALUE and the SQLCODE value is stored as CHAR for XML and INTEGER for non-XML messages.



CopyValueTo: Root.XML.outer.SqlCode

Node type: GetSQLCode

It should be noted that the location specified for CopyValueTo must be compatible with the integer data type of the SQLCODE.

The way in which the plug-in copies the SQLCODE value into the message depends upon whether the message is an XML message or not.

For XML messages, the SQLCODE value is converted to a CHAR data type and then copied into the specified element.

For non-XML messages, if the location specified already exists in the message, the plug-in always copies the SQLCODE as an INTEGER type value.

When specifying a location in an XML message, the ‘XML’ qualifier can be omitted. For example, the CopyValueTo attribute settings ‘Root.XML.outer.SqlCode and ‘Root.outer.SqlCode will locate/create the same syntax element in the message.

Chapter 2. Installation

SupportPac contents

The supplied zip file should be unzipped in a temporary directory. The following files and sub-directories will be created.

/source

Makefile.NT

Makefile.AIX

Makefile.Solaris

Makefile.HPUX

getSQLCode.c

extractCode.c

plugin.c

retcodes.c

utilities.c

getSQLCode.h

getSQLCode_retcodes.h

getSQLCode_constants.h

plugin.h

unicode.h

threads.h

rc.h

trace_publ.h

trace_defs.h

trace_func.h

trace_data.h

getSQLCodeTest.c

samples.mak.NT

samples.mak.AIX

samples.mak.Solaris

/NT/help

MessageProcessingNodeType_GetSQLCode.htm

/NT/bin

getSQLCode.lil
v201/getSQLCode.lil
getSQLCodeTest.exe

/NT/messages

MQSIV2_getSQLCode.msg
MQSIV2_ getSQLCode _msg.h
MQSIV2_ getSQLCode.dll
Makefile

/NT/objects

traceinit.obj
trace.obj
error.obj
unicode.obj
threads.obj

/NT/config

GetSQLCode
GetSQLCode.wdp

/NT/images

GetSQLCode.gif
GetSQLCode 30.gif
GetSQLCode 42.gif
GetSQLCode 58.gif
GetSQLCode 84.gif

/AIX

/AIX/bin

getSQLCode.lil
v201/getSQLCode.lil
getSQLCodeTest

/AIX/messages

MQSIV2_getSQLCode.cat
MQSIV2_ getSQLCode.msg
MQSIV2_ getSQLCode _msg.h

/AIX/objects

traceinit.o
trace.o
error.o
threads.o
unicode.o

/Solaris

/Solaris/bin

getSQLCode.lil
v201/getSQLCode.lil
getSQLCodeTest

/Solaris/messages

MQSIV2_getSQLCode.cat
MQSIV2_ getSQLCode.msg
MQSIV2_ getSQLCode _msg.h

/Solaris/objects

traceinit.o
trace.o
error.o
threads.o
unicode.o

/HPUX

/HPUX/bin

getSQLCode.lil
v201/getSQLCode.lil
getSQLCodeTest

/HPUX/messages

MQSIV2_getSQLCode.cat

MQSIV2_ getSQLCode.msg

MQSIV2_ getSQLCode _msg.h

/HPUX/objects

traceinit.o

trace.o

error.o

threads.o

unicode.o

license2.txt

ia0j.pdf

Prerequisites

This SupportPac provides a Plug-in node to be used with the IBM MQSeries Integrator for Windows NT or 2000 - V2.0.1, IBM MQSeries Integrator for AIX - V2.0.1, IBM MQSeries Integrator for Solaris - V2.0.1 and above and IBM MQSeries Integrator for HP-UX - V2.0.2. For normal use, there are no other pre-requisite products other than those required by MQSeries Integrator V2.1 itself.

Supported platforms

This SupportPac has been developed for and tested on

- Microsoft Windows NT environment
- IBM AIX environment
- Sun Solaris environment
- HP-UX environment

Installing the plug-in node on Windows NT

1. Unzip the packaged files into a temporary directory.
2. Copy the message catalogue file **NT\messages\MQSIV2_getSQLCode.dll** to a directory of your choice, e.g. <MQSI root>\messages.
3. Add an entry for the message catalogue to the registry. Use regedit to add an entry to the registry under…

```
HKEY_LOCAL_MACHINE
    SYSTEM
      CurrentControlSet
        Services
          EventLog
             Application
```

   Create a new entry with the following details

```
MQSIV2_getSQLCode
```

```
(default)              (value not set)
EventMessageFile   <fully qualified name of MQSIV2_getSQLCode.dll>
TypesSupported        0x00000007 (7)
```

4. Copy the file **NT\bin\getSQLCode.lil** to the MQSeries Integrator bin directory, e.g. <MQSI_root>\bin.

5. Restart the broker and check for plug-in initialisation messages in Event log

Installing the plug-in node on AIX

6. Unzip the packaged files into a temporary directory.

7. Copy the message catalogue file **AIX\bin\MQSIV2_getSQLCode.cat** to a directory specified by the NLSPATH setting, e.g. <MQSI_root>/messages.

8. Copy the file **AIX\bin\getSQLCode.lil** to the MQSeries Integrator lil directory (<MQSI_root>/lil).

9. Restart the broker and check for plug-in initialisation messages in the syslog.

Installing the plug-in node on Solaris

1. Unzip the packaged files into a temporary directory.

2. Copy the message catalogue file **Solaris\messages\ MQSIV2_getSQLCode.cat** to a directory specified by the NLSPATH setting, e.g. /usr/lib/locale/<*locale*>/LC_MESSAGES where <*locale*> is the locale under which the machine is running or 'C' if none is set.

3. Copy the file **Solaris\bin\getSQLCode.lil** to the MQSeries Integrator lil directory (<MQSI_root>/lil).

4. Restart the broker and check for plug-in initialisation messages in the syslog.

Installing the plug-in node on HP-UX

1. Unzip the packaged files into a temporary directory.

2. Copy the message catalogue file **HPUX\messages\ MQSIV2_getSQLCode.cat** to a directory specified by the NLSPATH setting, e.g. /usr/lib/locale/<*locale*>/LC_MESSAGES where <*locale*> is the locale under which the machine is running or 'C' if none is set.

3. Copy the file **HPUX\bin\getSQLCode.lil** to the MQSeries Integrator lil directory (<MQSI_root>/lil).

4. Restart the broker and check for plug-in initialisation messages in the syslog.

Integrating the plug-in node into the Windows Control Center

1. Unzip the packaged files into a temporary directory.

2. Change to the **NT\images** directory and copy its contents to <MQSI_root>\Tool\images

3. Change to the **NT\config** directory and copy its contents to <MQSI_root>\tool\repository\private\<machine name>\<Queue Manager name>\MessageProcessingNodeType

4. Change to the **NT\help** directory and copy its contents to <MQSI_root>\tool\help\com\isv

5. Start the MQSeries Integrator Control Center and display the MessageFlows panel. Right click on IBM Primitives and select **Add to Workspace**, then **Message Flow**. Select the GetSQLCode node from the displayed list and add it to the palette.

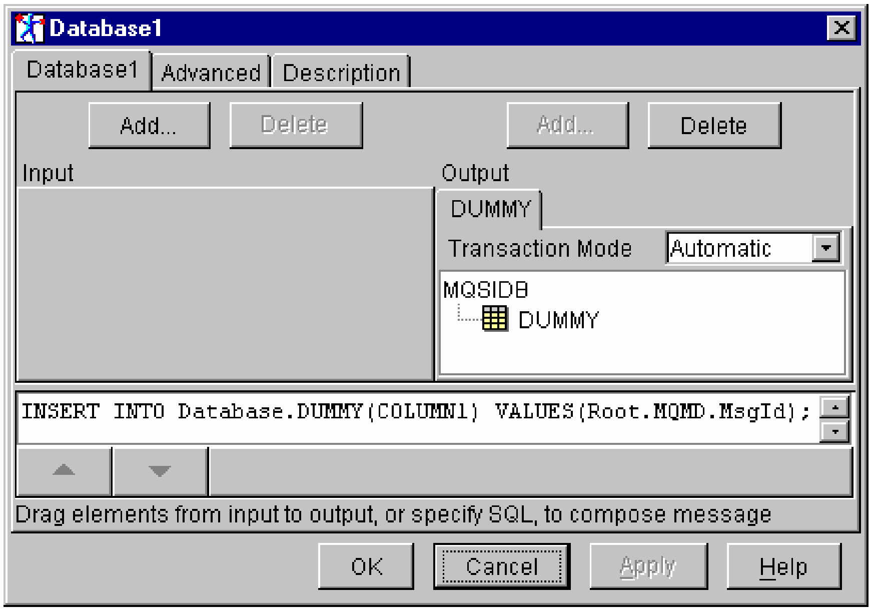
6. Check in the new node type

Installation verification

Create a message flow that is similar to the one shown in the Overview section.

The sample program getSQLCodeTest puts a simple message on the I1.Q and reads the modified message (containing the extracted SQLCODE) from the SQLCODE_42704.Q. The database node can consist of any ESQL that will generate a SQL error, however, in the example a non-existent table (DUMMY) in a valid database (MQSIDB) is used to generate the SQL error 42704 on the NT platform.

Examples of the database and filter node configurations are shown below.

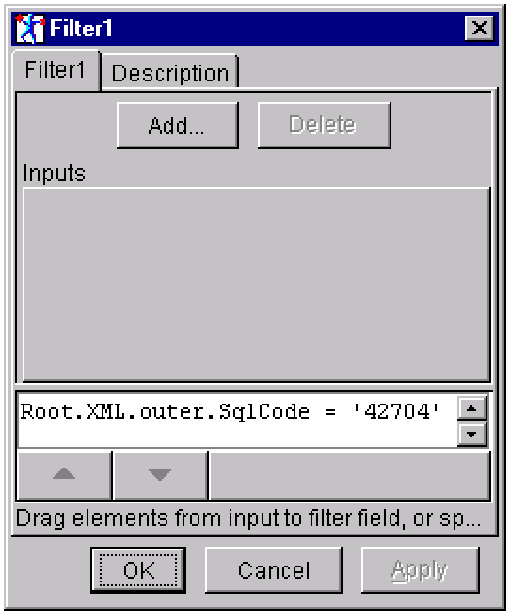


Transaction Mode: Automatic

MQSIDB
DUMMY

```
INSERT INTO Database.DUMMY(COLUMN1) VALUES(Root.MQMD.MsgId);
```

Drag elements from input to output, or specify SQL, to compose message



```
Root.XML.outer.SqlCode = '42704'
```

Drag elements from input to filter field, or sp...

By default, the getSQLCodeTest program will use the queues I1.Q and SQLCODE_42704.Q, however, you can specify your own queue manager and queues by invoking the application in the following way

```
timerTest <queueManagerName> <putQueue> <getQueue>
```

The source for the GetSQLCodeTest program is supplied along with an appropriate makefile for the target Operating System, e.g. samples.mak.NT to enable you to easily rebuild it.

Chapter 3. Nodes Reference

GetSQLCode Node

The GetSQLCode node is responsible for extracting the SQLCODE value from the current message's Exception List structure and storing it in a new message at the location specified by the CopyValueTo attribute.

Each message processed by a GetSQLCode node will cause the GetSQLCode node to perform the following actions:

- Locate and extract the SQLCODE value from the Exception List.
- Locate the syntax element specified by CopyValueTo property. If the element is not found, create it.
- Store the SQLCODE at the located (or created) syntax element ensuring that a CHAR data type is used for XML messages and an INTEGER data type for non-XML messages.

If the node fails to find or extract the SQLCODE, the input message is routed to the **notFound** queue.

If the node detects any other errors, the input message will be routed to the **failure** terminal.

GetSQLCode node terminals	
Terminal	**Description**
in	The input terminal that accepts a message for processing by the node
out	The output terminal to which the message is normally routed
failure	The output terminal to which the message is routed if there is an error during the node processing
notFound	The output terminal to which the message is routed if the SQLCODE cannot be found in or extracted from the Exception List.

GetSQLCode node properties

These properties are displayed when you right click a GetSQLCode node entry in the Message Flow Types pane, and click **Properties**. The values displayed are the default properties for this instance of the node. They cannot be edited when displayed from the Message Flow Types pane.

CopyValueTo

> This mandatory attribute identifies the location in the message which will hold the SQLCODE value. It must begin with '**Root.**' and if it references an existing element, then that element must be a suitable type to contain data. For example, it cannot be a NAME type element.

Configuring the GetSQLCode node

For a description of the properties of the GetSQLCode node and their possible values, see"GetSQLCode node properties" above.

To configure a GetSQLCode node:

1. In the Message Flow Definition pane, right click the symbol of the GetSQLCode node you want to configure and click **Properties**. The **GetSQLCode node** dialog is displayed.

2. In the **GetSQLCode node** dialog, type values for those properties that you want to set.

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

4. Click **OK** to finish configuring this GetSQLCode node.

Tracing the GetSQLCode plug-in node

To trace execution of the plug-in nodes, set the GETSQLCODE_PLUGIN_TRACE environment variable (system variable on NT) and reboot machine **before** restarting broker. Settings for trace are as follows.

GETSQLCODE_PLUGIN_TRACE =

-f *traceOutputFileName* - name of file to write trace to

-t - include time stamp on trace entries.

-i - include process and thread id on entries

-c - commit (flush) entries to file after every write

-l **-** trace level to output (see trace values below)

-a - append trace to existing trace file

Valid trace level settings are

- TRACE_NONE
- TRACE_ENTRY_EXIT
- TRACE_ERROR
- TRACE_WARNING
- TRACE_INFO
- TRACE_SYSTEM_ALL
- TRACE_ALL

The following setting results in comprehensive tracing and will be sufficient in most cases.

```
GETSQLCODE_PLUGIN_TRACE=-f <name of output file> -i –c -l TRACE_SYSTEM_ALL
```

End of Document