# MQSeries Integrator - MQGet Plug-In
## Version 1.3.1

28th August 2001

Neil Kolban
MQSeries Technical Support
IBM Advanced Technical Support
Dallas, TX
USA

kolban@us.ibm.com

**Take Note!**

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

**Fifth Edition, August 2001**

This edition applies to Version 1.3.1 of *MQSeries Integrator – MQGet Plug-In* and to all subsequent releases and modifications unless otherwise indicated in new editions.

# Table of Contents

Notices ..... iv
Trademarks and service marks ..... iv
Summary of Amendments ..... v
Preface ..... vi
Bibliography ..... vii
Chapter 1. Introduction ..... 1
Chapter 2. Installing the plug-in node ..... 2
SupportPac contents ..... 2
Prerequisites ..... 2
Supported Platforms ..... 3
Installing the plug-in node on broker system ..... 3
Integrating the plug-in node into the Windows Control Center ..... 3
Defining the node to the configuration repository ..... 5
Chapter 3. Using the plug-in node ..... 6
Description ..... 6
Plug-in node terminals ..... 6
Plug-in node properties ..... 6
Chapter 4. Example using the plug-in node ..... 8
Request/Reply processing ..... 8
Request processing flow ..... 9
Reply Processing Flow ..... 10

# 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:

- Windows NT, Visual Studio — Microsoft Corporation
- Solaris — Sun Corporation

# Summary of Amendments

| Date | Changes |
|---|---|
| 9th March 2001 | Initial release |
| 19th March 2001 | Initial release on AIX, no functional changes |
| 16th April 2001 | Initial release on Sun Solaris, no functional changes |
| 25th July 2001 | Fixed for MQSI V2.0.2 changes |
| 28th August 2001 | Rebuild of Solaris and addition of MQSI AIX Message catalog |

# Preface

The MQSeries Integrator Version 2 (MQSI V2) product provides the capability for it to be extended with the creation of custom nodes and parsers. This SupportPac provides an extension node called **MQGet**. This node is intended to be used within an MQSI message flow to extract a message from an MQSeries queue during flow processing.

# Bibliography


- *IBM MQSeries Integrator for Windows NT Version 2 Installation Guide,* IBM Corporation. SC34-5600.
- *IBM MQSeries Integrator for Sun Solaris Version 2 Installation Guide,* IBM Corporation. SC34-5842
- *IBM MQSeries Integrator for AIX Version 2 Installation Guide,* IBM Corporation. SC34-5841
- *IBM MQSeries Integrator Version 2 Using the Control Center,* IBM Corporation. SC34-5602
- *IBM MQSeries Integrator Version 2 Programming Guide,* IBM Corporation. SC34-5603

# Chapter 1. Introduction

This document describes the **MQGet** add-on node for the MQSI V2 product. The purpose of this node is to provide the ability to obtain a message from an MQSeries queue in the middle of a message flow as opposed to simply at the start of the message flow. This capability has a wide variety of new uses, including:

- Handling MQSI V2 Request/Reply processing
- Providing additional message aggregation capabilities
- Maintaining state from one message flow to the next

# Chapter 2. Installing the plug-in node

## SupportPac contents

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

/source

NodeUtils.h
NodeUtils.c
MQGet.c
StdAfx.cpp
StdAfx.h

/source/NT_msg

mqget_msg.txt

/source/AIX_msg

MQSIV2_MQGET.msg

/source/SOLARIS_msg

msg.h
MQSIV2_MQGET.msg

/NT

MQGet
MQGet.gif
MQGet.lil
MQGet.properties
MQGet.wdp
MessageProcessingNodeType_MQGet.htm
MQGet30.gif
MQGet42.gif
MQGet58.gif
MQGet84.gif

/AIX

MQGET.lil
MQSIV2_MQGET.cat

/SOLARIS

MQGET.lil
MQSIV2_MQGET.cat

license2.txt

ia09.pdf

## Prerequisites

This SupportPac provides a plug-in node to be used with the IBM MQSeries Integrator Version 2.0.1 and above. For normal use, there are no other prerequisite products other than those required by IBM MQSeries Integrator Version 2.0.1 itself. If any changes are to be made to the plug-in node, an appropriate C++ compiler is required.

## Supported Platforms

This SupportPac has been developed and tested for

- Microsoft Windows NT environment.
- IBM AIX environment
- Sun Solaris environment

## Installing the plug-in node on broker system

The plug-in '**lil**' file should be installed by copying or moving the appropriate file to the following directory:

- <mqsi_root>\bin (Windows)
- <mqsi_root>\lil (AIX, Solaris)

On a Unix environment, the Message catalog file (MQSIV2_MQGET.cat) must be copied to the following directory

- <mqsi_root>\messages (AIX)
- /usr/lib/locale/*locale*/LC_MESSAGES (SOLARIS)
  where *locale* is the locale under which the machine is running or "C" if none is set.

You must stop and restart the broker to enable it to detect the existence of the new '**lil**'.

## Integrating the plug-in node into the Windows Control Center

The necessary files for integrating the plug-in into the Windows Control Center are provided in the /NT directory.

Use the following table to copy the files to their correct location. These locations should already exist providing you have deployed at least one message flow. Append your ***<MQSI V2 root install path>*** to the **Copy to location** value.

Use the following to replace the placeholders:

<hostname> - TCP/IP hostname

<CM QMName> - Configuration Manager's queue manager name

| Filename | Copy to location |
|---|---|
| MQGet | \Tool\repository\private\<hostname>\<CM QMName>\MessageProcessingNodeType |
| MQGet.wdp | \Tool\repository\private\<hostname>\<CM QMName>\MessageProcessingNodeType |
| MQGet.gif | \Tool\images |
| MQGet30.gif | \Tool\images |
| MQGet42.gif | \Tool\images |
| MQGet58.gif | \Tool\images |
| MQGet84.gif | \Tool\images |
| MQGet.properties | \Tool\com\ibm\ivm\mqitool\extensions |
| MessageProcessingNode Type_MQGet.htm | \Tool\help\com\isv |

Note: If any of the target folders or directories does not exist, create it before copying the file to the specific location.

## Defining the node to the configuration repository

When you have installed the files in the appropriate directories, as described in the previous section, you must make these definitions available to the Control Center.

1. Start the Control Center. The user ID you are using must be a member of the MQSeries Integrator group ***mqbrdevt***. You are recommended to use the superuser ***IBMMQSI2*** to complete this task[1]. This causes your new node to be locked under the same user ID as all the supplied IBM primitive nodes. If you do not use this user ID, the definition files in the configuration repository might be accidentally locked, and therefore open to unauthorized update.

2. Select the Message Flows view.

3. Select an existing Message Flow Category, or create a new one.

4. Right-click the selected category, and select *Add->Message Flow*.

   A list box is displayed showing all existing IBM-supplied primitive nodes and any defined message flows you have installed following the instructions provided.

5. Select the message flow (the node).

   This node now appears within the message flow category you selected in the tree view in the left-hand pane.

6. Select your new node, and right-click. Select *Check In*.

7. Right-click again, and select Lock. Then right-click again and select Check In for a second time. After this check, the interface and ****.wdp*** definition files disappear from the local directory and go into the shared repository, where they are available to all users of the Control Center. However, user can only use this new node if they have installed the additional files (icons, properties files, and so on) on their own system.

[1] You must take care if you change logon IDs to complete this task. Changing logon IDs can effect the operation of the Configuration Manager's queue manager if it is on this system, but not running as a Windows NT service. See the *MQSeries Integrator Administration Guide* for more information about queue manager operation (Chapter 2) and the superuser *IBMMQSI2* (Chapter 4).

# Chapter 3. Using the plug-in node

## Description

Once installed, the MQGet node can be added to any message flow. The node has the following primitives icon:

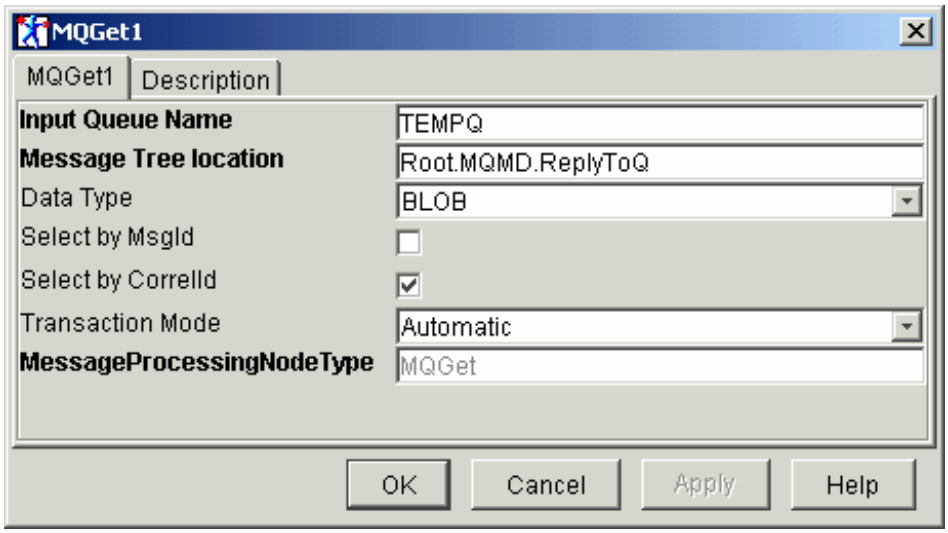
When displayed in a message flow, it appears as the following icon:

## Plug-in node terminals

| Terminal | Description |
|---|---|
| In | The input terminal that accepts the message being processed so far |
| Out | The flow continues from this terminal if a message was successfully obtained from the queue and added into the message tree. |
| No Message | The flow continues from this terminal if no message was available on the queue. The outgoing flow is identical to the incoming flow |
| Failure | The flow continues from this terminal if an error is encountered within the node while attempting to get a message from the queue. |

## Plug-in node properties

When the node's properties are selected, the following property dialog is displayed:




Within the property dialog, the attributes of this instance may be selected. The following properties are defined:

- **Input Queue Name** – The name of the MQSeries queue from which the message is to be obtained. This is a mandatory parameter. The queue must exist and be local to the queue manager associated with the broker.

- **Message Tree Location** – When a message is obtained from the queue, this mandatory parameter names the location within the message tree where the results will be stored. The message is saved as a single element in the tree. The location within the tree need not exist prior to adding the message and will be created if needed. The location is specified in ESQL path format. For example 'Root.XML.A.Name' would result in the message retrieved from the queue being assigned to the 'Root.XML.A.Name' location within the message tree. The path within the message tree must start with 'Root'. The ESQL suffix 'Body' is not permitted.

- **Data Type** – The message retrieved from the queue can be stored in the message tree as either a String or a BLOB (ByteArray). This option specifies which.

- **Select by MsgId** – This option allows the message to be selected from the queue by MsgId as opposed to simply the first message available in the queue. If checked, the MsgId to be matched is taken from the 'Root.MQMD.MsgId' of the message received in the 'in' terminal.

- **Select by CorrelId** – This option allows the message to be selected from the queue by CorrelId as opposed to simply the first message available in the queue. If checked, the CorrelId to be matched is taken from the 'Root.MQMD.CorrelId' of the message received in the 'in' terminal.

- **Transaction Mode** – This option allows the message to be obtained as part of the logical unit of work of the message flow itself. The possible values of this attribute are:

    - **Automatic** – If the message obtained from the queue is a persistent message, then it will be included in the logical unit of work. If the message is non-persistent, it will not be included.

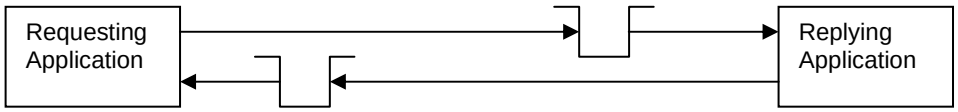
    - **Yes** – The message will be obtained from the queue as part of the logical unit of work.

    - **No** – The message will not be obtained from the queue as part of the logical unit of work.

# Chapter 4. Example using the plug-in node

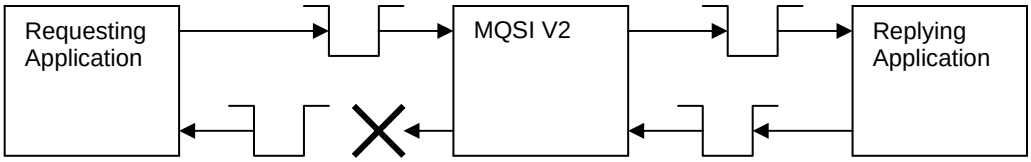
## Request/Reply processing

A common usage of MQSI is to transform data for request/reply based messages. The basic MQSeries Request/Reply paradigm is illustrated in the following diagram:




In this story, the requesting application places its request message on the input queue of the program processing requests and sending replies. The requesting program names a queue onto which the reply message should be saved. This is done by placing a queue name in the ReplyToQ field of the original request message.

When MQSI V2 is introduced to transform the message between the requestor and the replying application, the diagram looks as follows:




In this story, the requesting application places its request on a queue that MQSI V2 is watching. MQSI V2 retrieves the request message and performs reformatting upon it. It then places the resulting message on the queue of the replying application and names another MQSI V2 onto which the reply should return. When the replying application responds, it places the message onto the second MQSI V2 input queue. MQSI V2 then retrieves this second message, reformats it … and has a problem. The original ReplyToQ queue name contained within the original request message was lost and now MQSI V2 has no knowledge of which queue to send the actual reply message too.

The solution to this problem is to cache the original ReplyToQ message somewhere and, on receipt of the reply, lookup the ReplyToQ based on CorrelId or MsgId and send to that queue.

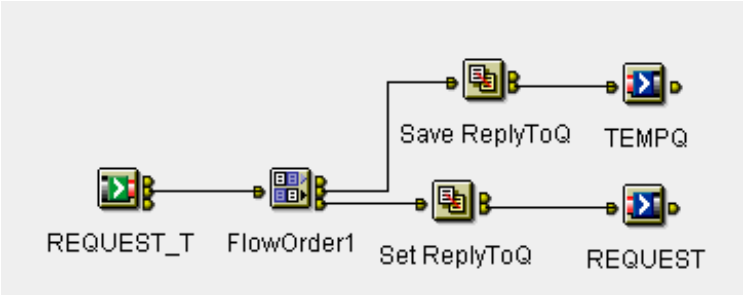
Prior to the introduction of the MQGet node, this was achieved by saving the ReplyToQ in a database and performing a database lookup on the response. This is a relatively expensive operation and does not handle the situation where a reply message never returns and thus stale ReplyToQ information starts to accumulate in the database.

With the MQGet node, the original ReplyToQ can be saved as a message on an MQSeries queue and retrieved from it when the reply arrives.

The following two message flows illustrate this technique for a request/reply-based application:

## Request processing flow

The following diagram illustrates a message flow that saves a copy of the ReplyToQ on an MQSeries queue called TEMPQ:




The original message is received on a queue called REQUEST_T. It is then sent in two directions using the flow control node. First, it passes into the 'Save ReplyToQ' compute node.

The ESQL in this node looks as follows:

```
DECLARE I INTEGER;
SET I = 1;
WHILE I < CARDINALITY(InputRoot.*[]) DO
        SET OutputRoot.*[I] = InputRoot.*[I];
        SET I=I+1;
END WHILE;
-- Enter SQL below this line.  SQL above this line might be
regenerated, causing any modifications to be lost.
SET OutputRoot.TREEASIS.ReplyToQ = InputRoot.MQMD.ReplyToQ;
```

**Note**: This node uses the TREEASIS parser; see MQSeries SupportPac IA05 available at:

*http://www.ibm.com/software/mqseries/txppacs/ia05.html*

Next, the same message is sent through the second path of the flow control node where it meets the 'Set ReplyToQ' compute node that has ESQL as follows:

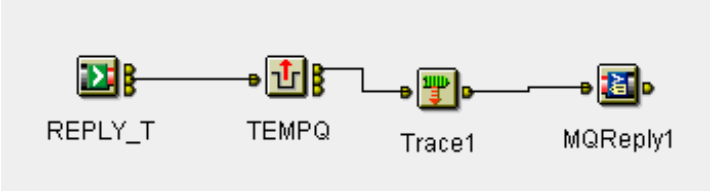
```
SET OutputRoot = InputRoot;
-- Enter SQL below this line.  SQL above this line might be
regenerated, causing any modifications to be lost.
SET OutputRoot.MQMD.ReplyToQ = 'REPLY_T';
```

This sets the ReplyToQ to be 'REPLY_T'.

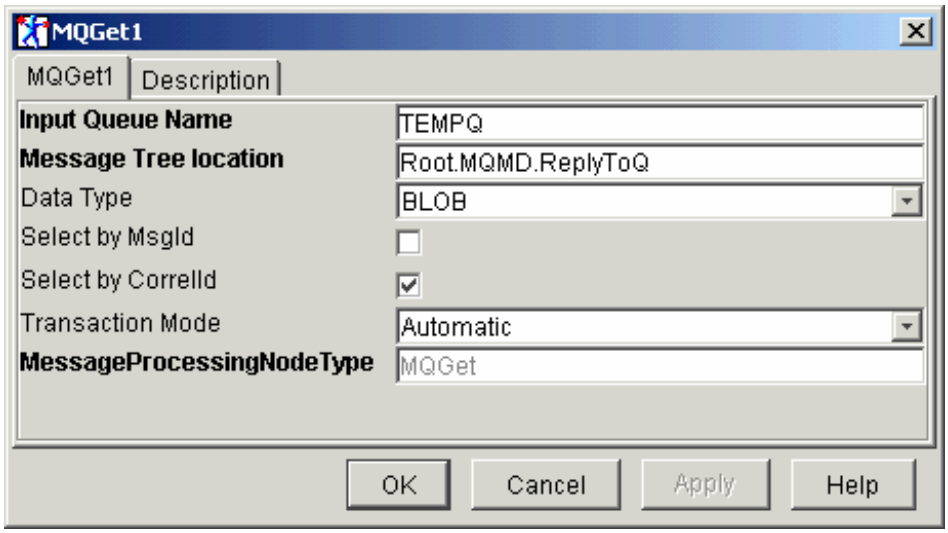
The message is the placed on the real REQUEST queue being watched by the replying application.

## Reply Processing Flow

The message flow for reply processing is illustrated next:




When the reply is received from the responding application, it is then submitted to the TEMPQ node, which is an instance of the MQGet node. It is configured with the following properties:




These properties instruct the node to get a message from the TEMPQ queue, which matches the CorrelId of the incoming message. When the message is retrieved, it is store in the 'Root.MQMD.ReplyToQ' field on the message and propagated through the 'out' terminal.

When the message finally reaches the 'MQReply1' node, the message is then sent to the queue named in the ReplyToQ field and reaches the original requesting application.

End of Document