MQSeries Integrator - IMS Message Handler Plug-in

Version 1.0

17 September 2001

David Arnold
IBM UK Labs Ltd
Hursley Park
Winchester

David_Arnold@uk.ibm.com

Take Note!

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

First Edition, September 2001

This edition applies to Version 1.0 of *MQSeries Integrator - IMS Message Handler 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:

- IBM
- MQSeries
- MQSeries Integrator
- MQSI

The following terms are trademarks of other companies:

- Windows NT, Visual Studio Microsoft Corporation

Summary of Amendments

Date	Changes
17 September 2001	Initial release

Preface

This SupportPac is an MQSeries Integrator V2 custom plug-in node. The node is designed for use in the transformation of messages moving to and from an IMS environment via MQSeries. The node is responsible for the addition or removal of the IMS segment headers, LLZZ(s). The SupportPac contains the functional details of the node, instructions for the build and install of the node and examples of how it can be used.

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. Overview

Functional Overview

This version of the node has been designed to work with a message parsed with the BLOB (Binary Large Object) parser only. Therefore, if MRM or XML formatted data is required, a ResetContentDescriptor node will be needed either before, after or either side of the node.

When the source of the message is IMS i.e. the node is configured to go *FROM IMS* the node will strip out the LLZZ (or LLZZs if multiple segments) and build a message containing raw data.

Example Message Data in:

IIH	LLZZ	TRANID	Message Data

Example Message Data out:

IIH	TRANID	Message Data

When the target of the message is IMS i.e. the node if configured to go *TO IMS* the node will take the data portion, calculate its length and add an LLZZ at the head of the message data to build a single segment IMS message.

Example Message Data in:

IIH	TRANID	Message Data

Example Message Data out:

IIH	LLZZ	TRANID	Message Data

There are two types of IMS message supported by MQSeries. It is not compulsory that IMS messages contain an IMS header. The IMSMSGHandler node can support IMS messages with or without the IMS header (that is the MQIIH structure) present in the message data. The routing description below details how both IMS message types are handled.

Node Destination Terminal Routing Description

The node can be configured to send a non-IMS message to the failure terminal or allow them to pass unaltered to the output terminal. The node can also be configured to differentiate between IMS messages with or without an IMS header and route them to either the output or failure terminal. This routing is done dependent on the setting in the MSGTYPES property found in the BASIC tab of the node.

Checking is done for the correct combination of message descriptor (MD) Format and presence of an IMS Header (IIH) in the message regardless of the direction that has been set. This version of the node doesn't manipulate or add/remove message headers it only checks them and routes the message accordingly.

If the node is defined with MSGTYPES = ALLIMS it checks for a message descriptor format of MQFMT_IMS or MQFMT_IMS_VAR_STRING. If the message descriptor format is found to be MQFMT_IMS it will expect to find an IIH present.

If the message descriptor format is found to be MQFMT_ IMS_VAR_STRING it will not check for an IIH.

If the node is defined with MSGTYPES = IMSIIH it checks for a message descriptor format of MQFMT_IMS. If this is not the case the message will be routed to the failure terminal. If the message descriptor format is found to be MQFMT_IMS the node will require that an IIH be present in order to go ahead and operate on the data and route to the output terminal.

The following table defines the destination terminal dependent on the type of message received and the setting in the MSGTYPES property.

Node Destination Terminal Routing Table

IP Message description **(note message length 4096 is for example only)**	**MSGTYPES**	**MAXMSGDATASIZE**	**OP TERM**
IIH Included, MDFormat = MQIMSVS and Length<4096	ANY	4096	Output
IIH Included, MDFormat != MQFMT_IMS and Length<4096	ANY	4096	Output
IIH Not Included, MDFormat = MQFMT_IMS_VAR_STRING and Length<4096	ANY	4096	Output
IIH Not Included, MDFormat != MQFMT_IMS_VAR_STRING and Length<4096	ANY	4096	Output
MD Format != MQFMT_IMS or MQFMT_IMS_VAR_STRING and Length<4096	ANY	4096	Output
IIH Included, MDFormat = MQIMSVS and Length<4096	ALLIMS	4096	Output
IIH Included, MDFormat != MQFMT_IMS and Length<4096	ALLIMS	4096	Failure
IIH Not Included, MDFormat = MQFMT_IMS_VAR_STRING and Length<4096	ALLIMS	4096	Output
IIH Not Included, MDFormat !=	ALLIMS	4096	Failure

MQFMT_IMS_VAR_STRING and Length<4096			
MD Format != MQFMT_IMS or MQFMT_IMS_VAR_STRING and Lengbbbbbbth<4096	ALLIMS	4096	Failure
IIH Included, MDFormat = MQFMT_IMS and Length<4096	IMSIIH	4096	Output
IIH not Included or MDFormat != MQFMT_IMS	IMSIIH	4096	Failure
Any message >4096 in length	*All options*	4096	Failure

DEBUG and MAXDATASIZE properties

Internal Node Debugging

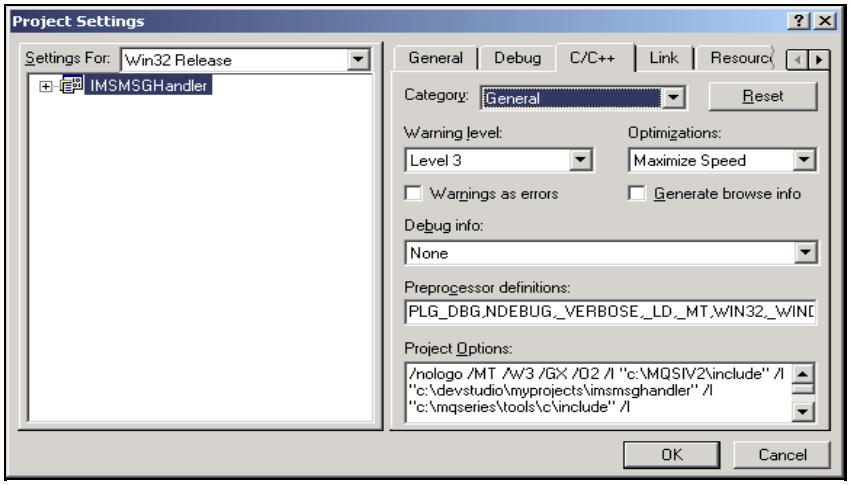
The node contains extensive internal tracing code that writes to a flat file. The file name and location are not configurable.

File Name: IMSMSG.txt

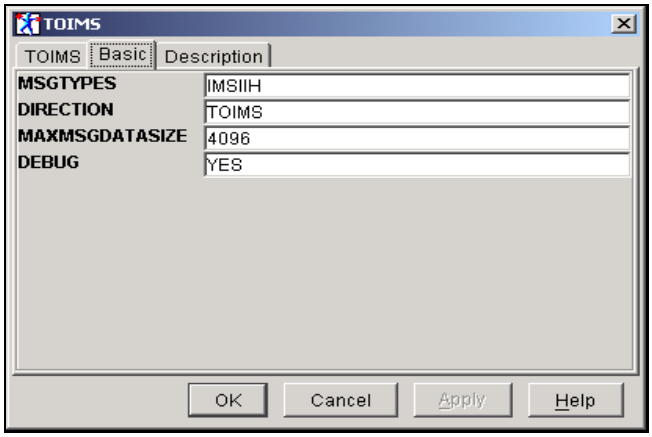
File Location: X:\<install path>\BIN (where x: is the MQSI v2 install drive)

Enabling debugging is a two-phase operation. To enable debugging the node must be conditionally compiled with flag PLG_DBG and have debugging enabled via the node properties. The tracing code and the checks for debugging enablement are hidden behind conditional compile to improved performance.

Add the conditional compile flag PLG_DBG to the preprocessor definitions in the project settings of the Microsoft C++ v6.0 or higher project.



Set the DEBUG property in the BASIC tag of the IMSMSGHandler node to YES.



An example of the how the node handles its internal tracing is as follows:

```
  outRootElement = cniRootElement(&rc, outMsg);
#if defined (PLG_DBG)  // Conditional Compile code
  if (DEBUGON == 1)     // Debug property set on for node check
     {fprintf(fptr, "\n outRootElement = cniRootElement(&rc, outMsg) rc = %d\n",rc);}
#endif
```

Maximum Message Data Size

Originally the IMSMSGHandler was coded to use the custom node function cniBufferSize to obtain the total size of the message buffer received by the node. This value is required to ensure that enough memory was allocated to build the datastream to be copied to the new output message. However testing showed that this function call was not 100% reliable in the BLOB domain and therefore the Max Message Data Size property was introduced. Users should set this property to the value of the largest piece of message data they expect to handle.

Chapter 2. Installing the Plug-in node

SupportPac contents

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

File Name	Description
IMSMSGHandler.c	Source file
IMSMSGHandler.h	Header file
IMSMSGHandler.mak	Microsoft C++ v6.0 exported make file
IMSMSGHandler.lil	Runtime library for Windows NT/2000
IMSMSGHandler.lil.dbg	Runtime library for Windows NT/2000 compiled for debug
IMSMSGHandler	XML interface definition file
IMSMSGHandler30.wdp	Web Dav properties file
IMSMSGHandler.gif	Tree view GIF
IMSMSGHandler30.gif	25% zoom GIF
IMSMSGHandler42.gif	50% zoom GIF
IMSMSGHandler58.gif	75% zoom GIF
IMSMSGHandler84.gif	100% zoom GIF
IMSMSGFlows.xml	Sample Message Flow export file
TOIMSiih.txt	TOIMS input IMS msg with IIH for use with MQSIPUT
FROMIMSiih.txt	FROMIMS input IMS msg with IIH for use with MQSIPUT
TOIMSnoiih.txt	TOIMS input IMS msg without IIH for use with MQSIPUT
FROMIMSiih.txt	FROMIMS IP IMS msg without IIH for use with MQSIPUT
TOIMSXML.txt	TOIMS input msg for end to end XML_TOIMS_BLOB
FROMIMSXML.txt	FROMIMS input msg for end to end BLOB_FROMIMS_XML

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 version of the IMS Message Handler was been written to support MQSI V2 on Windows NT/2000, AIX and Sun Solaris operating systems. It was developed and extensively tested on Windows NT/2000. An earlier version of this code was heavily tested on AIX. Since those tests support for IMS Messages without an IMS header included has been added. The node has not been retested on AIX since that support was added. The IMS Message Handler has not been compiled or tested on Sun Solaris.

General Installation

The XML Interface, Web Dav and GIF files have been supplied along with the nodes runtime LIL file. The MQSeries Integrator Programming Guide Chapter 7 “Installing a plug-in node or parser” should be followed for installation and deployment instructions.

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)

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
IMSMsgHandler	\Tool\repository\private\<hostname>\<CM QMName>\MessageProcessingNodeType
IMSMsgHandler.wdp	\Tool\repository\private\<hostname>\<CM QMName>\MessageProcessingNodeType
IMSMsgHandler.gif	\Tool\images
IMSMsgHandler30.gif	\Tool\images
IMSMsgHandler42.gif	\Tool\images
IMSMsgHandler58.gif	\Tool\images
IMSMsgHandler84.gif	\Tool\images
IMSMsgHandler.properties	\Tool\com\ibm\ivm\mqitool\extensions

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.

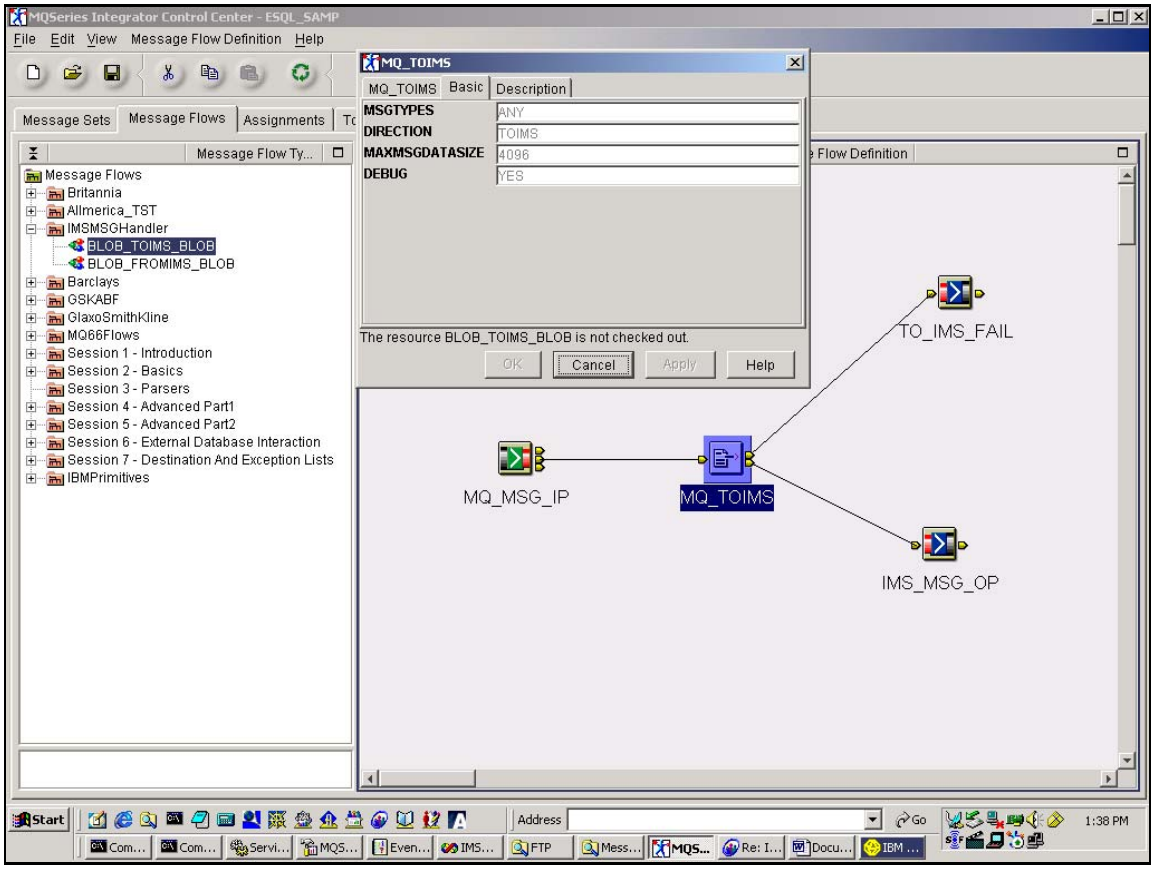
[1] You must take care if you change logon IDs to complete this task. Changing logon IDs can effect the operation of the Configuration Manager's queue manager if it is on this system, but not running as a Windows NT service. See the *MQSeries Integrator Administration Guide* for more information about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

Chapter 3. Using the plug-in node

Description

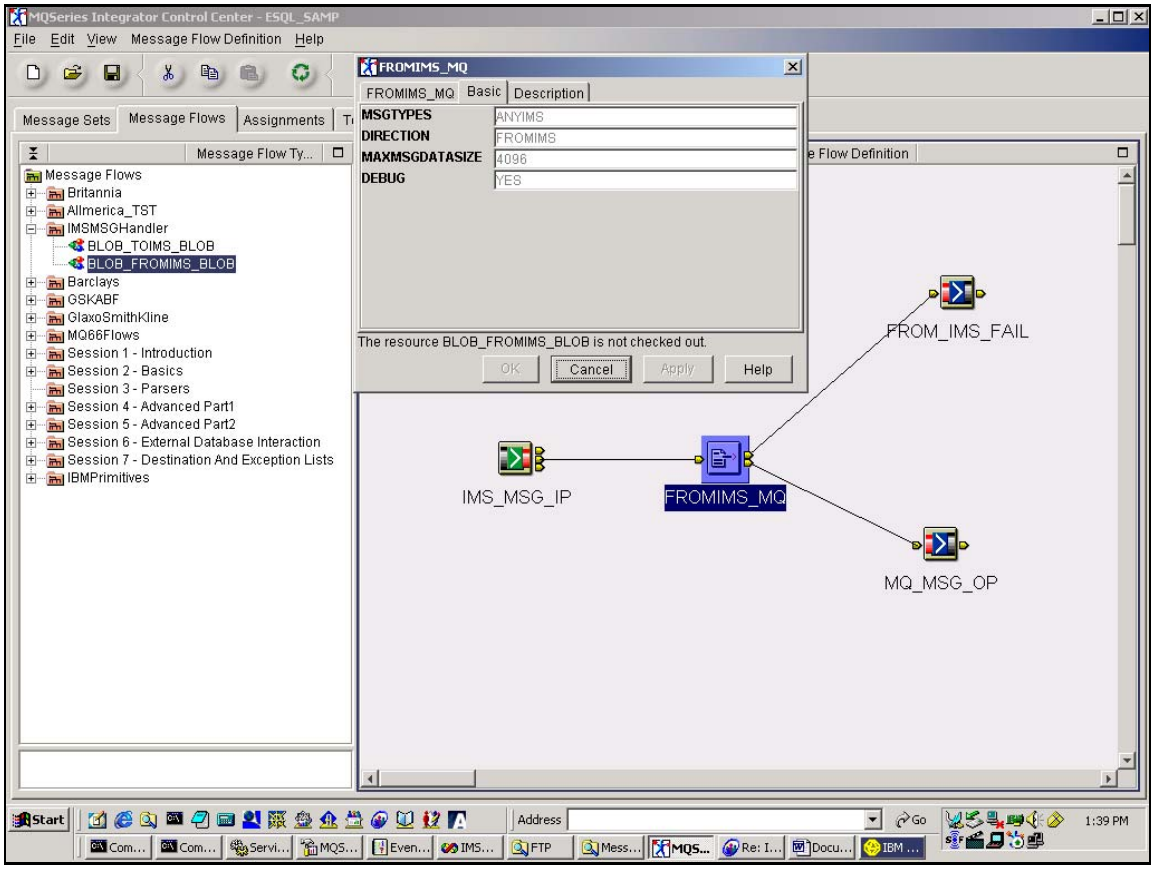
The following section details how the IMS Message Handler node can be employed in its most basic form, simply adding or removing IMS segment headers. The screen captures show the node properties and their values depending on the functionality required.

Message Direction from MQSeries to IMS



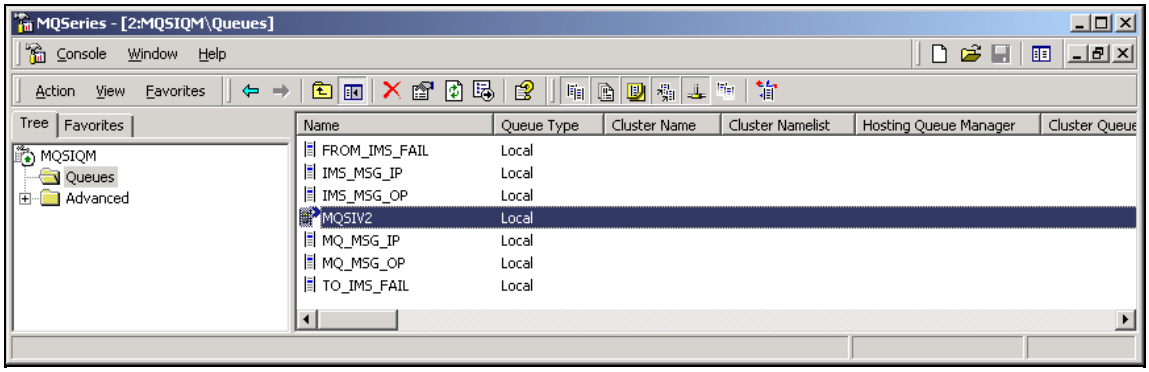
The above screen capture shows the IMSMSGHandler node in its most basic use simply adding an LLZZ to a message already prepared for delivery to IMS.

Message Direction from MQSeries to IMS



The above screen capture shows the IMSMSGHandler node in its most basic use simply configured to remove LLZZ(s) from a message containing IMS data.

MQSeries Queue definitions

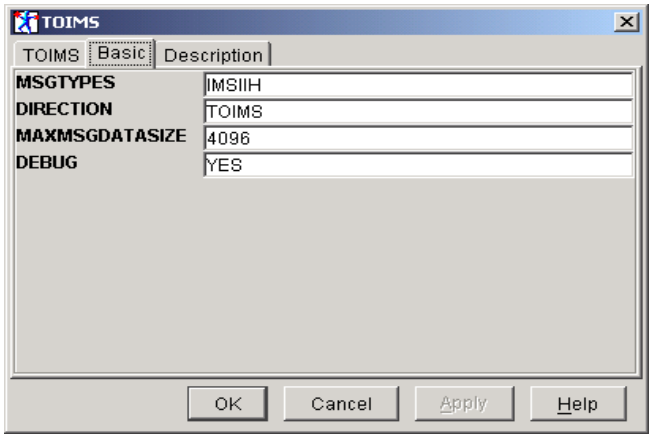


Plug-in node terminals

Terminal	Description
In	The input terminal that accepts a message for processing by the node
Out	The output terminal that outputs the original message
Failure	The output terminal which the message is routed if failure is detected during processing the message.

Plug-in node properties

Example



Initial Values

MSGTYPES: ANY/ALLIMS/IMSIIH

DIRECTION:TOIMS/FROMIMS

MAXMSGDATASIZE:0000

DEBUG:NO/YES

Chapter 4. Compiling the plug-in node

Windows NT, AIX and Sun Solaris

The IMSMSGHandler node should be compiled and linked in accordance with the instructions detailed in the MQSI v2.0.1or above programming guide section “Compiling a plug-in”.

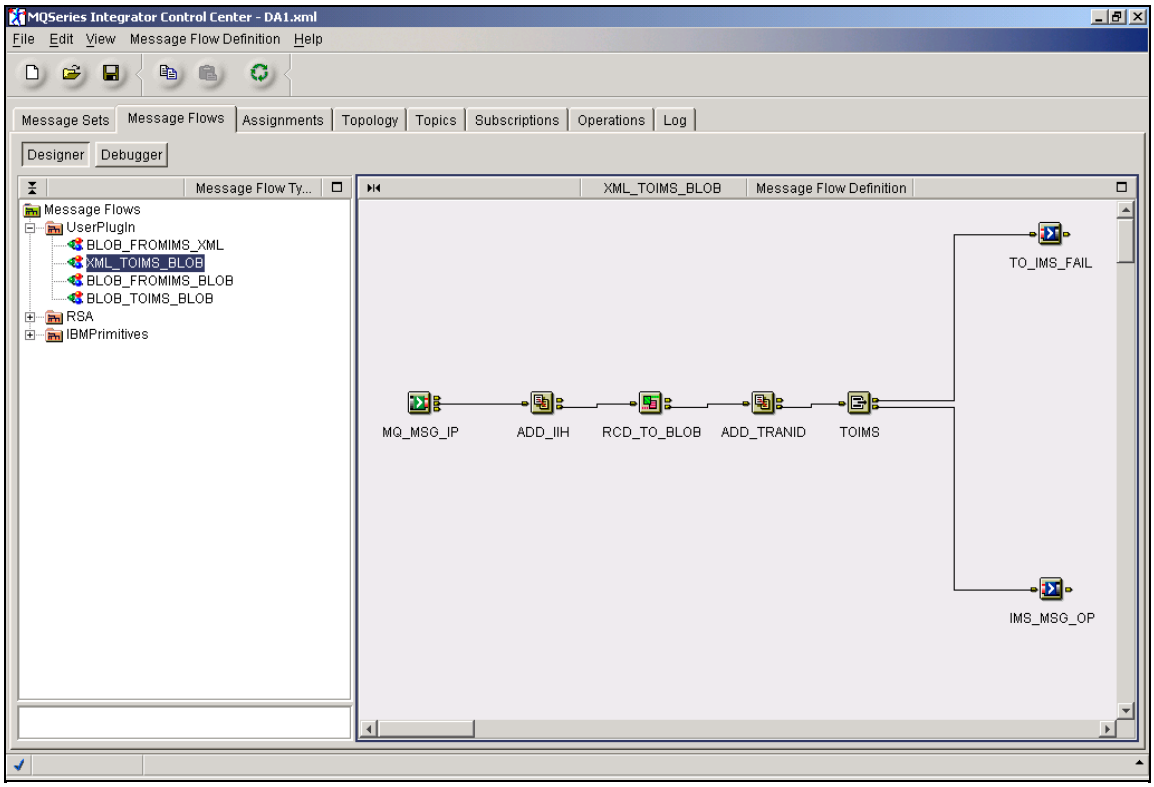
Chapter 5. Example using the plug-in node

This version of the IMSMSGHandler node does not add/strip the IMS header itself, handle the transaction code or manipulate the message descriptor itself. Developers will need to handle this functionality themselves using the ESQL language in compute nodes.

Target Environment is IMS

The steps required in a flow whose target is IMS and input message data is XML are:

1 MQInput Node.
    1.1 Set the Message Domain to XML
2 Compute Node.
    2.1 Add an MQIIH IMS header.
    2.2 Populate the IMS Header fields.
    2.3 Update the Message Descriptor Format field.
3 ResetContentDescriptor Node.
    3.1 Set Message Format to BLOB.
4 Compute Node.
    4.1 Insert the IMS transaction ID ahead of the data (at OutputRoot.BLOB.BLOB)
5 IMSMSGHandler node.
    5.1 Set with DIRECTION = TOIMS to add the llzz to build an IMS segment.
6 MQOutput node.



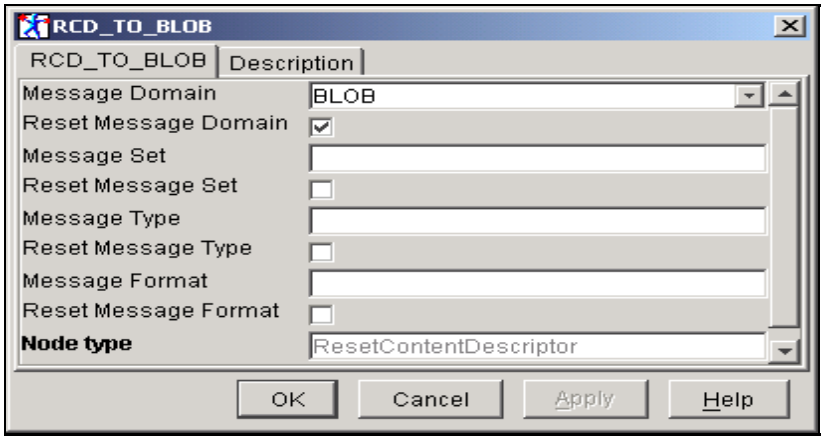
ADD_IIH – Compute Node ESQL

```
DECLARE C INTEGER;
SET C = CARDINALITY(InputRoot.*[]);
DECLARE I INTEGER;
SET I = 1;
WHILE I < C 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.MQIIH.Format = 'MQIMSVS ';
SET OutputRoot.MQIIH.Version = 1;
SET OutputRoot.MQIIH.Encoding = 273;
SET OutputRoot.MQIIH.CodedCharSetId = 437;
SET OutputRoot.MQIIH.Flags = 0;
SET OutputRoot.MQMD.Format = 'MQIMS   ';
SET OutputRoot.XML = InputBody;
```

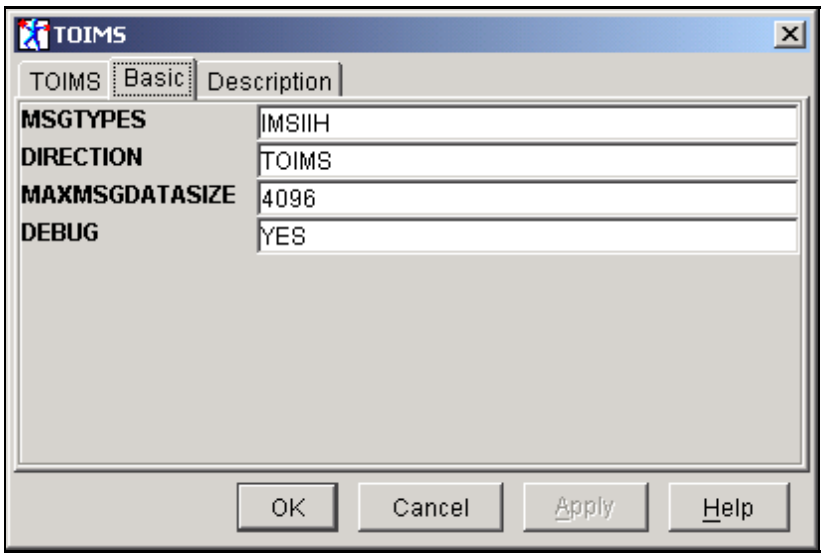
ADD_TRANID – Compute Node ESQL

```
DECLARE C INTEGER;
SET C = CARDINALITY(InputRoot.*[]);
DECLARE I INTEGER;
SET I = 1;
WHILE I < C 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.
DECLARE Tran BLOB;
DECLARE Tran_Data BLOB;
SET Tran = X'535545494f504342'; -- IMS Transaction ID is SUEIOPCB
SET Tran_Data = Tran||InputRoot."BLOB"."BLOB";
SET OutputRoot."BLOB"."BLOB"= Tran_Data;
```

Switch to BLOB – Reset Content Descriptor



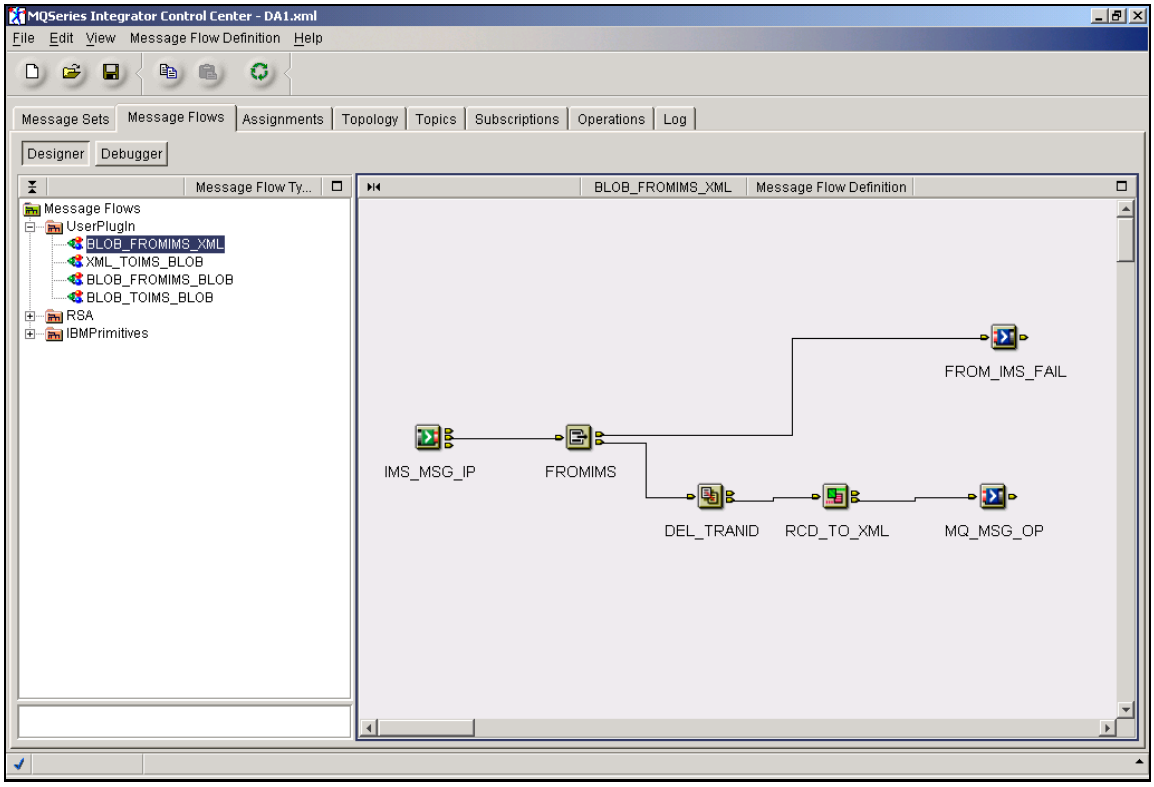
TOIMS – IMSMsghandler Node



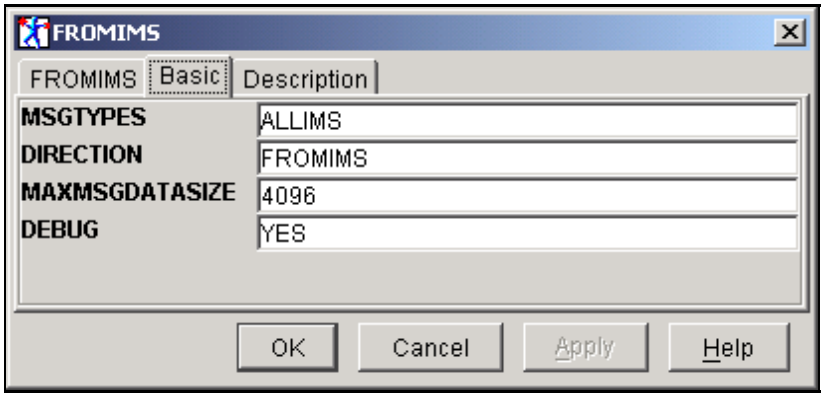
Originating Environment is IMS

The steps required in a flow whose target is IMS and output message data is XML are:

1 MQInput Node.
    1.1 Set the message domain to BLOB.
2 IMSMSGHandler node.
    2.1 Set with DIRECTION = FROMIMS to remove llzz(s) from IMS segment(s).
3 Compute Node.
    3.1 Remove the IMS Transaction ID
4 ResetContentDescriptor node
    4.1 Set the message format to XML
5 Continue to process message as required as XML.



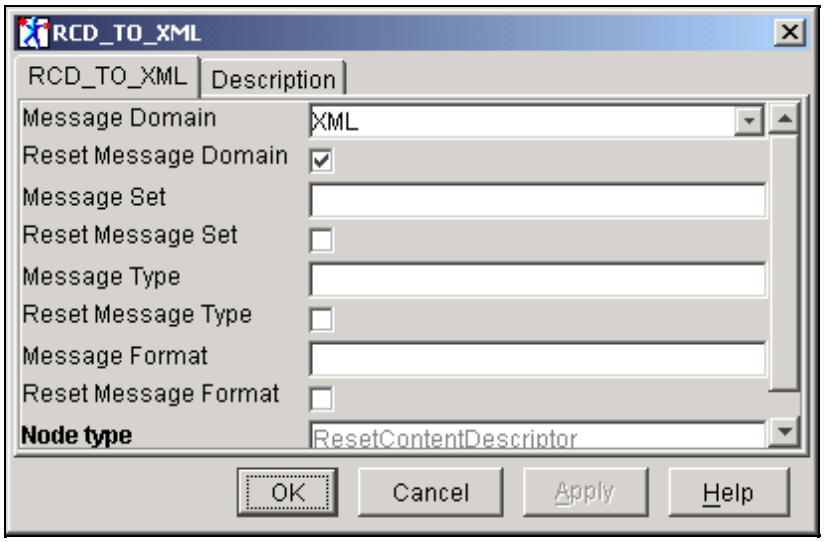
TOIMS – IMSMsghandler Node



DEL_TRANID – Compute Node ESQL

```
DECLARE C INTEGER;
SET C = CARDINALITY(InputRoot.*[]);
DECLARE I INTEGER;
SET I = 1;
WHILE I < C 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.
DECLARE NOTran_Data BLOB;
DECLARE DataLen INTEGER;
SET DataLen = LENGTH(InputRoot."BLOB"."BLOB");
SET DataLen = DataLen - 8;-- Minus the length (8) of a TRANID
SET NOTran_Data = SUBSTRING(InputRoot."BLOB"."BLOB" FROM 9 FOR DataLen);
SET OutputRoot."BLOB"."BLOB"= NOTran_Data;
```

Switch to XML - Reset Content Descriptor Node



Chapter 6. IMS Message Handler Sample Messages

The sample messages files are packaged with IMSMSGHandler ZIP file.

Simple Single node examples

These sample messages are for use in the simple case scenario detailed in sections 1.1 and 1.2 where there is no manipulation of the Message headers MD and IIH or the IMS Transaction ID. The basic action of the standalone node is demonstrated.

Input Message for use with MQSIPUT containing an IIH but no LLZZ

```
OPTIONS
DEBUGLEVEL         1
OPENOPTIONS        2064
MQPMOOPTIONS       2048
DELIMITER          %%
TESTSTART
MQMD
STRUCID            MD
VERSION            2
REPORT             0
MSGTYPE            8
EXPIRY             -1
FEEDBACK           0
ENCODING           546
CODEDCHARSETID     850
FORMAT             MQIMS
PRIORITY           0
PERSISTENCE        1
BACKOUTCOUNT       0
REPLYTOQ           REPL.OUTPUT
REPLYTOQMGR        MQSIQM1
MQIIH
STRUCID            IIH
VERSION            1
STRUCLENGTH        84
ENCODING           273
CODEDCHARSETID     437
FORMAT             MQIMSVS
FLAGS              0
STARTDATA
%%535545494f504342003230303030303230303030303530383334323534303830323230303131 36
323034363430303031202020202020202020202020202020202020202020202000303030303030303030
303030303030202020202020202020202020202020202020202020303030303030303030303030303030
303030303030303020203030303030303030302030303030303030303030303030303030303030303030
3030303131313135343030303030303030303030303030303030303030303030303030303030303030
303030303030303030303030303030303030303030303030303030303030303030303030303030%%

ENDDATA

TESTEND
```

Output Message with single LLZZ segment added

```
49 49 48 20 01 00 00 00  IIH ....
54 00 00 00 11 01 00 00  T.......
B5 01 00 00 4D 51 49 4D  µ...MQIM
53 56 53 20 00 00 00 00  SVS ....
20 20 20 20 20 20 20 20
20 20 20 20 20 20 20 20
20 20 20 20 20 20 20 20
20 20 20 20 20 20 20 20
00 00 00 00 00 00 00 00  ........
00 00 00 00 00 00 00 00  ........
20 30 43 20 00 EE 00 00   0C .î..
53 55 45 49 4F 50 43 42  SUEIOPCB
00 32 30 30 30 30 30 32  .2000002
30 30 30 30 30 35 30 38  00000508
33 34 32 35 34 30 38 30  34254080
32 32 30 30 31 31 36 32  22001162
30 34 36 34 30 30 30 31  04640001
20 20 20 20 20 20 20 20
20 20 20 20 20 20 20 20
20 20 20 20 20 20 00 30         .0
30 30 30 30 30 30 30 30  00000000
30 30 30 30 30 20 20 20  00000
20 20 20 20 20 20 20 20
20 20 20 20 20 20 20 20
30 30 30 30 30 30 30 30  00000000
30 30 30 30 30 30 30 30  00000000
30 30 30 30 30 30 30 20  0000000
20 30 30 30 30 30 30 30   0000000
30 20 30 30 30 30 30 30  0 000000
30 30 30 30 30 30 30 30  00000000
30 30 30 30 30 30 30 30  00000000
30 30 31 31 31 31 35 34  00111154
30 30 30 30 30 30 30 30  00000000
30 30 30 30 30 30 30 30  00000000
30 30 30 30 30 30 30 30  00000000
30 30 30 30 30 30 30 30  00000000
30 30 30 30 30 30 30 30  00000000
30 30 30 30 30 30 30 30  00000000
30 30 30 30 30 30 30 30  00000000
30 30 30 30 30 30 30 30  00000000
30 30                    00
```

Input Message for use with MQSIPUT containing an IIH and LLZZ(s)

```
OPTIONS
DEBUGLEVEL          1
OPENOPTIONS          2064
MQPMOOPTIONS       2048
DELIMITER           %%
TESTSTART
MQMD
STRUCID              MD
VERSION              2
REPORT               0
MSGTYPE              8
EXPIRY               -1
FEEDBACK             0
ENCODING             546
CODEDCHARSETID   850
FORMAT               MQIMS
PRIORITY             0
PERSISTENCE          1
BACKOUTCOUNT     0
REPLYTOQ             REPL.OUTPUT
REPLYTOQMGR      MQSIQM1
MQIIH
STRUCID              IIH
VERSION              1
STRUCLENGTH      84
ENCODING             273
CODEDCHARSETID   437
FORMAT               MQIMSVS
FLAGS                0
STARTDATA
%%00EE0000535545494f5043420032303030303032303030303035303833343235343038303232303
0303131363230343634303030312020202020202020202020202020202020202020202020003030303
0303030303030303030202020202020202020202020202020202020203030303030303030303030303
0303030303030303030303020203030303030303020303030303030303030303030303030303030303
0303030303030303131313135343030303030303030303030303030303030303030303030303030303
0303030303030303030303030303030303030303030303030303030303030303030303030303030303030%
%

ENDDATA
TESTEND
```

Output Message with LLZZ(s) removed

```
49 49 48 20 01 00 00 00  IIH ....
54 00 00 00 11 01 00 00  T.......
B5 01 00 00 4D 51 49 4D  µ...MQIM
53 56 53 20 00 00 00 00  SVS ....
20 20 20 20 20 20 20 20
20 20 20 20 20 20 20 20
20 20 20 20 20 20 20 20
20 20 20 20 20 20 20 20
00 00 00 00 00 00 00 00  ........
00 00 00 00 00 00 00 00  ........
20 30 43 20 53 55 45 49   0C SUEI
4F 50 43 42 00 32 30 30  OPCB.200
30 30 30 32 30 30 30 30  00020000
30 35 30 38 33 34 32 35  05083425
34 30 38 30 32 32 30 30  40802200
31 31 36 32 30 34 36 34  11620464
30 30 30 31 20 20 20 20  0001
20 20 20 20 20 20 20 20
20 20 20 20 20 20 20 20
20 20 00 30 30 30 30 30    .00000
30 30 30 30 30 30 30 30  00000000
30 20 20 20 20 20 20 20  0
20 20 20 20 20 20 20 20
20 20 20 20 30 30 30 30      0000
30 30 30 30 30 30 30 30  00000000
30 30 30 30 30 30 30 30  00000000
30 30 30 20 20 30 30 30  000  000
30 30 30 30 30 20 30 30  00000 00
30 30 30 30 30 30 30 30  00000000
30 30 30 30 30 30 30 30  00000000
30 30 30 30 30 30 31 31  00000011
31 31 35 34 30 30 30 30  11540000
30 30 30 30 30 30 30 30  00000000
30 30 30 30 30 30 30 30  00000000
30 30 30 30 30 30 30 30  00000000
30 30 30 30 30 30 30 30  00000000
30 30 30 30 30 30 30 30  00000000
30 30 30 30 30 30 30 30  00000000
30 30 30 30 30 30 30 30  00000000
30 30 30 30 30 30        000000
```

Multi-Node End to End Examples

These sample messages are for use with the end-to-end examples detailed in section 4 where the completed message flows link IMS and non-IMS systems together. Along with the manipulation of the message data, the message headers MD and IIH are managed and the IMS transaction ID is added or removed as required.

Input Message for use with MQSIPUT for flow XML_TOIMS_BLOB

```
OPTIONS
DEBUGLEVEL              1
OPENOPTIONS             2064
MQPMOOPTIONS            2048
TESTSTART
MQMD
STRUCID                 MD
VERSION                 2
REPORT                  0
MSGTYPE                 8
EXPIRY                  -1
FEEDBACK                0
ENCODING                273
CODEDCHARSETID          437
FORMAT                  xml
PRIORITY                0
PERSISTENCE             1
BACKOUTCOUNT            0
REPLYTOQ                IMS.REPLY
REPLYTOQMGR             MQSIQM

STARTDATA
<?xml version="1.0" encoding="UTF-8"?>
<IMS>
    <IMSdata>this is the IMS data</IMSdata>
</IMS>
ENDDATA
```

Output Message from flow XML_TOIMS_BLOB

```
49 49 48 20 00 00 00 01  IIH ....
00 00 00 54 00 00 01 11  ...T....
00 00 01 B5 4D 51 49 4D  ...µMQIM
53 56 53 20 00 00 00 00  SVS ....
20 20 20 20 20 20 20 20
20 20 20 20 20 20 20 20
20 20 20 20 20 20 20 20
20 20 20 20 20 20 20 20
00 00 00 00 00 00 00 00  ........
00 00 00 00 00 00 00 00  ........
20 30 43 20 00 64 00 00   0C .d..
53 55 45 49 4F 50 43 42  SUEIOPCB
3C 3F 78 6D 6C 20 76 65  <?xml ve
72 73 69 6F 6E 3D 22 31  rsion="1
2E 30 22 20 65 6E 63 6F  .0" enco
64 69 6E 67 3D 22 55 54  ding="UT
46 2D 38 22 3F 3E 3C 49  F-8"?><I
4D 53 3E 3C 49 4D 53 64  MS><IMSd
61 74 61 3E 74 68 69 73  ata>this
20 69 73 20 74 68 65 20   is the
49 4D 53 20 64 61 74 61  IMS data
3C 2F 49 4D 53 64 61 74  </IMSdat
61 3E 3C 2F 49 4D 53 3E  a></IMS>
```

Input Message for use with MQSIPUT for flow BLOB_FROMIMS_XML

```
OPTIONS
DEBUGLEVEL          1
OPENOPTIONS         2064
MQPMOOPTIONS        2048
DELIMITER           %%
TESTSTART
MQMD
STRUCID             MD
VERSION             2
REPORT              0
MSGTYPE             8
EXPIRY              -1
FEEDBACK            0
ENCODING            273
CODEDCHARSETID      437
FORMAT              MQIMSVS
PRIORITY            0
PERSISTENCE         1
BACKOUTCOUNT        0
REPLYTOQ            IMS.REPLY
REPLYTOQMGR         MQSIQM
```

STARTDATA
%%0064000055345494F5043423C3F786D6C2076657273696F6E3D22312E302220656E636F646
96E673D225554462D38223F3E3C494D533E3C494D53646174613E7468697320697320746865204
94D5320646174613C2F494D53646174613E3C2F494D533E%%

ENDDATA
TESTEND

Output Message from flow BLOB_FROMIMS_XML

```
3C 3F 78 6D 6C 20 76 65  <?xml ve
72 73 69 6F 6E 3D 22 31  rsion="1
2E 30 22 20 65 6E 63 6F  .0" enco
64 69 6E 67 3D 22 55 54  ding="UT
46 2D 38 22 3F 3E 3C 49  F-8"?><I
4D 53 3E 3C 49 4D 53 64  MS><IMSd
61 74 61 3E 74 68 69 73  ata>this
20 69 73 20 74 68 65 20   is the
49 4D 53 20 64 61 74 61  IMS data
3C 2F 49 4D 53 64 61 74  </IMSdat
61 3E 3C 2F 49 4D 53 3E  a></IMS>
```

End of Document