MQSeries Integrator - DxPool Plug-In

Version 2.0

28th March 2003

Prafull Kumar
Sr. Software Engineer
IBM India Software Labs
Golden Enclave, Airport Road
Bangalore-560 017
India

kprafull@in.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 - DxPool Plug-in* and to all subsequent releases and modifications unless otherwise indicated in new editions.

Table of Contents

Notices

Trademarks and service marks

Summary of Amendments

Date	Changes
17th September 2001	Initial release
28th March 2003	Supported on AIX
	Supported on Solaris
	Supported on HP-UX
	Improved locking mechanism on Unix platforms

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 DxPool (Data eXchange Pool) Node. This node is intended to be used within an MQSI message flow to share the data between messages. A message can be passed through the same message flow or different message flows. These message flows can be of the same broker, within the same execution group or in different execution group or may be in different Brokers on the same machine.

This node is based on shared memory concept to share the data between messages. Through DxPool node you can store some of the data of the input message in the shared memory or you can insert the shared memory data into the output message by using the ESQL which is very similar to compute node.

Bibliography

- *IBM MQSeries Integrator for Windows NT Version 2 Installation Guide,* IBM Corporation. SC34-5600.
- *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

Description

DxPool node is intended to be used within an MQSeries Integrator message flow to share the data between messages. The message can be passed through the same message flow or different message flows. These message flow can be of the same broker, with in the same execution group or in different execution group or may be in different Brokers on the same machine.

When the business grows and the messages start becoming complex, user is often in need of sharing the data between messages. To achieve this, DxPool Node provides a simple but robust method of data sharing between messages. DxPool node is implemented using the Shared Memory concept. In that way all the processes running on the same system can use this shared memory segment. (Database Node can be used to achieve the same output in most of the cases but this node provide the simple, faster method to share data between messages).

ESQL is provided to store and retrieve the data from the DxPool. It is same as for the compute Node. This provides the user a simple and highly flexible way to store and retrieve the data from the Pool.

Facility is provided to retain the shared memory even after broker shut down by storing the shared memory data into a simple file.

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:

/code

- DxPool.c
- DxPoolUtil.c
- node_utils.c
- DxPool.h
- DxPoolUtil.h
- node_utils.h
- make.aix
- make.hpux
- make.solaris
- makeNT.bat

/config

- DxPool
- DxPool.properties
- DxPool.wdp

/images

- DxPool.gif
- DxPool30.gif
- DxPool42.gif
- DxPool52.gif
- DxPool84.gif

/lils

- DxPool.lil (for each platform)

/messages/NT

- PluginMsg.h
- PluginMsg.mc
- Make.bat
- PluginMsg.dll

/messages/<Unix>

- pluginMsg.h
- pluginMsg.msg
- MQSI-DxPool.cat

Ia79.pdf

DxPoolV1.zip [IA79 for the first release on 17Sept01]

license2.txt

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 in a Microsoft Windows NT/2000, AIX, Solaris and HPUX 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	(Unix)

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

- Add an entry for the message catalogue to the registry. Use regedit to add an entry to the registry under…(Windows)

```
HKEY_LOCAL_MACHINE
  SYSTEM
    ControlSet001
       Services
         EventLog
           Application
```

Create a new entry with the following details

MQSI-DxPool

(default)	(value not set)
EventMessageFile	<fully qualified name of directory containing **pluginMsg.dll**>
TypesSupported	0x00000007 (7)

- Copy the .cat file to <mqsi-root>\messages directory. (Unix)

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 /config, /images 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
DxPool	\Tool\repository\private\<hostname>\<CM QMName>\MessageProcessingNodeType
DxPool.wdp	\Tool\repository\private\<hostname>\<CM QMName>\MessageProcessingNodeType
DxPool.gif	\Tool\images
DxPool30.gif	\Tool\images
DxPool42.gif	\Tool\images
DxPool58.gif	\Tool\images
DxPool84.gif	\Tool\images
DxPool.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

To share the data between messages

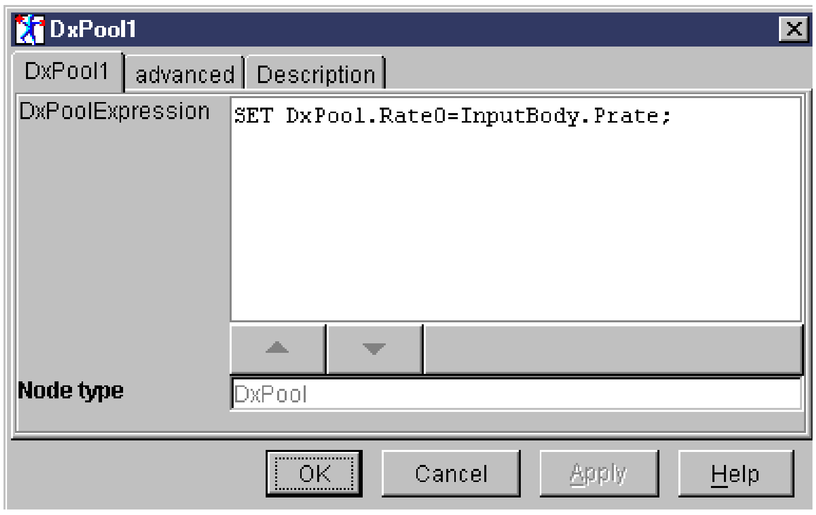
Plug-in node terminals

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

Basic Properties

DxPoolExpression: To write the script to be executed by the DxPool Node.



DxPoolExpression

```
SET DxPool.Rate0=InputBody.Prate;
```

Node type: DxPool

Programming

As of now, you are only allowed to save the data in the Pool and retrieve the data from the pool. Script language (ESQL) is provided to do the same. It is similar to compute node.

1) To save the data in the Pool you can use the DxPool.Key_Name as an argument

Example

SET DxPool.Tata.SharePrice=InputBody.CurrentPrice.Tata.SharePrice.PriceInPound;

cases:

- if InputBody.CurrentPrice.Tata.SharePrice.PriceInPound Element exists in message, it will be stored in the Pool and the Message will go to the out terminal.
- if InputBody.CurrentPrice.Tata.SharePrice.PriceInPound is not found in the message then the Message will go to the failure terminal and nothing will be saved in the Pool.
- if Pool Memory is full then an event will be generated, stating the same and message will go to failure terminal.

Pool Memory Size

DXPOOL_MEMORY_PAGES environment variable should be set according to the usage of the Pool Memory. by default this is set to 1(one) Page. each page size is 4096 bytes. You can set this environment variable according to the Memory usage of your messages.

To calculate the memory size required by your message is as follows:

Control Block size : 12 bytes

Each Message will require: 12 bytes+Key Size+Message Size.

2) To retrieve the data from the pool and insert it in the Message.

Example

SET OutputBody.Price.Script= DxPool.Tata.SharePrice;

cases:

- if DxPool.Tata.SharePrice exists in the Pool then the element OutputBody.Price.Script will be inserted in the Message and the Message will go to the out terminal.
- if DxPool.Tata.SharePrice will not be found by the Pool then the Message will go to the failure terminal.
- if the Node will not be able to create the Element due to any reason the message will propagate to failure terminal.

Note : you can write as many lines as you want.

PS: any other expression is not supported by DxPool Node. You can only store the data and retrieve it from the Pool. You can't use *, /, \ and other expressions.

Advanced Properties:

1) Store NULL Data in Pool even if Element Not found in Message

suppose the DxPoolExpression is:

SET DxPool.Share.Price=InputRoot.XML.Share.Price;

and in the input msg the Element Price does not exist. In that case if user checked this option, then the NULL value will be saved in pool message and it will go to OUT terminal. Without this option nothing will be saved in the Pool and the Message will go to failure terminal.

2) Insert NULL element in Message even if Data Not found in Pool

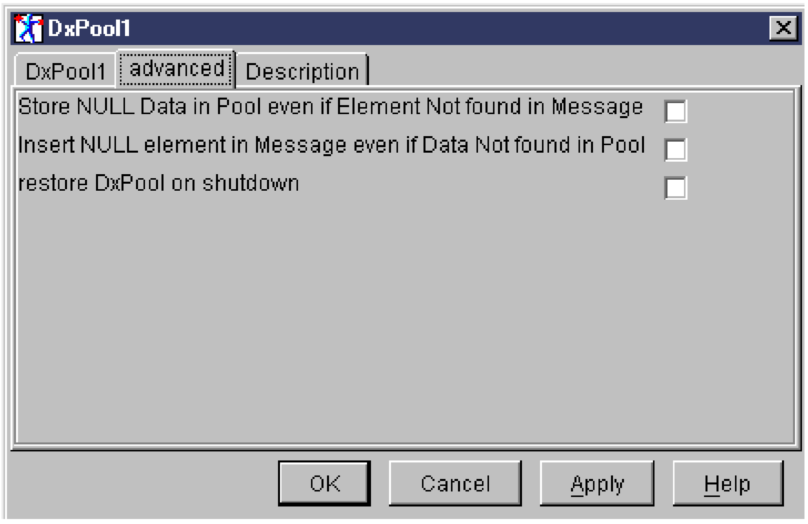
suppose the DxPoolExpression is:

SET OutputRoot.XML.Share.OldPrice=DxPool.Share.Price;

and the Element DxPool.Share.Price does not exist in Pool. In that case if user checked this option, then an element with NULL value will be inserted in message and the message will go to the out terminal. Without this option the message will go to failure terminal.

3) restore DxPool on shutdown

This option is used to restore the memory of DxPool node after shutdown of the Broker. The Pool Data will be saved in the c:\DxPoolData.bin and the Pool data will be filled with this data when the Broker starts again. You should check this option on all the DxPool node, If you want to restore the pool data.



Store NULL Data in Pool even if Element Not found in Message ☐
Insert NULL element in Message even if Data Not found in Pool ☐
restore DxPool on shutdown ☐

Debugging

To debug the DxPool node, messages will be logged in c:\DxPool.log file. You can check this file for any problem and this will help one to locate the problem, if at any time it occurs.

Chapter 4. Compiling the plug-in node

Windows NT

Create the message catalog using make.bat file in messages dir. This will create pluginMsg.dll and pluginMsg.h file.

Open the Project file DxPool.dsp in MS Visual C++ 6.0 and compile it.

Or use makeNT.bat file

UNIX

use the platform specific make file to build the source.

```
make -f make.<platform>
```

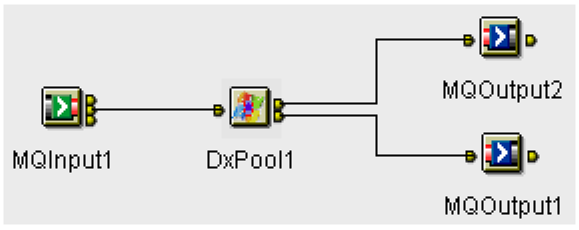
Chapter 5. Example using the plug-in node

Example 1 (between Message Flows)

Due to globalization, user may want to see the prices in different currencies (like $, £, Rs etc) in Stock exchange for instance and currently the Message contains the Price in Pounds(£) only.

To achieve this by DxPool Node, we can create two Message flows. The first Message flow can store the current conversion rate to the Shared Memory Pool. and the second Message flow can take this conversion rate from the Shared Memory Pool and inserted in the message. This information can be used by a compute node in the Message flow to calculate the Prices in other currencies.

Msg Flow 1

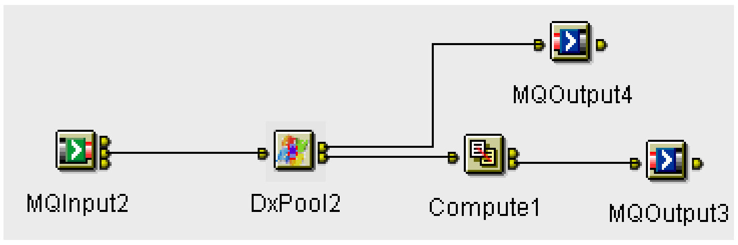


DxPool1 Script:

SET DxPool.ConvRate.PTD=InputBody.CurrentRate.PoundToDoller;

SET DxPool.ConvRate.PTR=InputBody.CurrentRate.PoundToRupee;

Msg Flow2



DxPool2 Script:

SET OutputBody.Share.Script.ConvRate.PoundToDoller=DxPool.ConvRate.PTD;

SET OutputBody.Share.Script.ConvRate.PoundToRupee=DxPool.ConvRate.PTR;

now the conversion rate will be inserted in the message as element OutputBody.Share.Script.ConvRate.PoundToDoller and OutputBody.Share.Script.ConvRate.PoundToRupee. these elements can be used by Compute Node to calculate the current price of Script in Dollars and Rupees.

Example 2 (with in the Same Message Flow)

To use the data of the previous message with the next message.

a) To calculate the % change of script Rate. The message contains the current rate of the scripts.

To achieve this you can put a DxPool Node in the message flow and store the rate of script. when the next message will come, you can insert the previous rate in the message. and can calculate the % change through a compute node in an message flow.

In DxPool

```
SET OutputRoot.XML.Rate.PreviousPrice=DxPool.Rate;

SET DxPool.Rate=InputRoot.XML.Rate.Price;
```

In Compute Node

```
SET OutputRoot.XML.Rate.PercentChange=((CAST(InputRoot.XML.Rate.Price as
INTEGER)-CAST(InputRoot.XML.Rate.PreviousPrice as
INTEGER))*100)/CAST(InputRoot.XML.Rate.Price as INTEGER);
```

Note: Advance Property "*Insert NULL element in Message even if Data Not found in Pool"* should be check on otherwise all Messages will go to failure Terminal.

b) To check the last processed MsgId by the Message flow. The message contains the messages MsgId.

```
SET OutputRoot.XML.Data.PriviousMsgId=DxPool.MsgId;

SET DxPool.MsgId=InputRoot.XML.Data.MsgId;
```

--- End of Document ---