Exstream
Dialogue™

WebSphere MQ Connector

Guide

Version 5.0

exstream software

2424 Harrodsburg Road, Suite 200
Lexington, KY 40503

(859) 296.0600
info@exstream.com :: www.exstream.com

Contents

Release Notes

:: Chapter 1

Release Notes

Release Notes

This document includes the following updated information. This list begins after release 1.5 of the connector.

VERSION 1.5.008

Addition of the -MSGFORMAT= option to set the message format field of the global message descriptor.

VERSION 1.5.007

-DDAFILEMAP for using the MQ connector on AIX platform.

VERSION 1.5.005

The code in the MQ Connector only terminates the buffer when a MQGET is successful.

VERSION 1.5.002

Added a wait interval parameter to MQ Connector -RECONNECT parameter.

VERSION 1.5.001

Addition of the -RECONNECT parameter.

MQ Connector can now handle "roll forward" and "roll back" transaction management operations with -TRANSMANAGE switch in the .ini file.

WebSphere MQ Connector

:: Chapter 2

WebSphere Solutions

IBM WebSphere MQ Server

The IBM WebSphere MQ Server provides heterogeneous any-to-any connectivity across platforms. This server is an IBM MOM (Message Oriented Middleware) package that:

- Stores and forwards messages (similar to e-mail).
- Features asynchronous or synchronous operation, with synchronous operation providing a return action, such as a message receipt.
- Provides load balancing for multiple Engines, such as multiple instances of Dialogue running one large job.

Exstream Software WebSphere MQ Connector

The Exstream Software WebSphere MQ Connector enables Dialogue to support the IBM WebSphere MQ system. The MQ (Message Queue) Connector compiles with client libraries to supports cross-platform delivery of messages to remote queues. As a multi-threaded module, the connector can be accessed by multiple Engines simultaneously.

WebSphere MQ Connector Installation on OS/390 and z/OS

Requirements

On the target system, you must have the Dialogue Engine and the IBM WebSphere MQ already installed. Also, at least one queue manager is set up and running.

Verify Disk Space

Be sure you have plenty of room to download and unzip the files. Additionally, be sure you have three times the memory available as your largest data file to run in production. If you use more than one data file in a production run, adjust your memory requirements accordingly.

Upload Terse

If you do not already have IBM's Terse application on your system, upload this program to your mainframe using the following IBM-provided directions.

Edit and Install

1. Allocate a PDS (Partitioned Data Set; PACKLIB) with the following characteristics:

– Space units	BLOCK
– Primary quantity	24
– Secondary quantity	3
– Directory blocks	1
– Record format	U
– Record length	0
– Block size	13030

2. Edit the sendtrsmain.txt file, which is called by the sendtrsmain.bat file. The sendtrsmain.bat file uses an ftp program to transfer the trsmain to mainframe.
3. Edit the following in sendtrs.txt:
   - Change IBM6000 to the URL or IP address of your mainframe.
   - Change p390a to your FTP username.
   - Change p390a6 to your FTP password.
4. Open a DOS window, and change the directory to the directory where sendtrsmain.txt and sendtrsmain.bat reside.
5. Type sendtrsmain at the DOS prompt. This transfers the terse program to the mainframe.
6. Issue TSO RECEIVE INDSN (your_local_dsn) from your TSO session to convert the TRSMAIN program back into its load module format.
   The following messages display after the above command.
   - INMR901I Dataset PTFLCG.TERSE409.LOADLIB from NHAN on PLPSC
   - INMR902I Members: TRSMAIN
   - INMR906A Enter restore parameters or 'DELETE' or 'END' +
7. Reply to INMR906A with the PDS name, which you allocated earlier, and member name - TRSMAIN. (i.e. DA(PACKLIB(TRSMAIN)).

8. Edit the sendtrs.txt file, which is called by the sendtrs.bat file. The sendtrs.bat file uses an ftp program to transfer the terse files, test data, JCL, required files and load module to the mainframe.
9. Change the following in sendtrs.txt:
   - Change IBM6000 to the URL or IP address of your mainframe.
   - Change p390a (userid) to your FTP username.
   - Change p390a6 (password) to your FTP password.
   - Change SCPMV3 to the volume where you want the terse files to be placed.
10. Open a DOS window, change the directory to the directory where sendtrs.txt and sendtrs.bat reside and type sendtrs at the DOS prompt. This transfers the terse files to the mainframe.
11. In the mainframe, edit the following file: MQSERIES.VERUNP
    - Add or edit your job card as appropriate.
    - In the definition of the UNPACK proc: Change userid to your high level qualifier where you loaded your Terse files. Example:
      `C P390A <new high level qualifier> all`
    - Change the VOLSER symbolic parameter default from SCPMV3 to your desired Volser.
12. Submit MQSERIES.VERUNP and check the results.

Verify Installation

1. Edit MQSERIES.JCL(VEREXMQ)
   - Change HLQ=P390A to indicate the high level qualifier where you loaded your Terse files.
   - Change EHLQ=P390A to indicate the high level qualifier where the Exstream Dialogue Engine is installed.
   - Change MQHLQ=CSQ520 to indicate the high level qualifier where MQ Series is installed.
   - Change QMGR='CSQ1' to indicate the name of your MQ Queue Manager.
   - Change the STEPLIB for the VERIFY step (by default: CSQ520.SCSQANLE) to match the location of your IBM MQ Series installation.
2. Edit MQSERIES.VERDATA(EXMQSINI)
   - Change QUEUEMGR=CSQ1 to indicate the name of your MQ Queue Manager.
   - Submit MQSERIES.JCL(VEREXMQ) and check the results.

WebSphere MQ Connector Installation on Other Platforms

Download the Zipped File

From the Exstream web site, download the MQ Connector zip file for your platform. Other than OS/390 and z/OS, the remaining prefix/platform choices are Windows, AIX, HP, Solaris, and SuSE LINUX.

The naming convention is:
XXX_MQConnector_X_X_XXX.zip

As an example: 'AIX_MQConnector_1_5_003.zip' is the MQ Connector version 1.5.003 for the AIX environment.

Extract Files

For the Windows installation, the connector DLL is ExMQConnect.dll. The ZIP file includes a test database with two sample applications and all the necessary files to run the test cases. The readme text file explains each sample file in detail.

The UNIX installation procedure is similar to installing a Dialogue Engine for any UNIX platform. Inside the zip file you find a tar file that contains the connector DLL for that platform (libExMQConnect.so) as well as several files needed to run two sample applications.

On a LINUX (SuSE) installation, the connector DLL is libExMQConnect.so. The ZIP file includes a test database with two sample applications.

WebSphere MQ Connector Specifications

:: Chapter 3

Specifications

The WebSphere MQ Connector is currently supported on Windows, Solaris, AIX, HPUX, OS/390, z/OS, and Linux (SuSE) platforms. This connector module for Driver files, Reference files, Report files, and print files has the following file name, default initialization file, and function name.

Specifications for MQ Connector by platform

	FILE NAME	FUNCTION NAME	DEFAULT INIT FILE
Windows	ExMQConnect.dll	processRec	EXMQS.INI
Unix and Linux	libExMQConnect.so	processRec	EXMQS.INI
OS/390 and z/OS	EXMQCONN	processRec	DD:EXMQSINI

As an option to using the default Initialization file, you can create your own .ini file (see *Open Parameters*, below).

Define in Dialogue Design Manager

Using the previous table as a guideline, here are the properties needed to define the MQ Connector on the *Test Data Source* and *Production Data Source* tabs for data files in the Property Panel.

Dynamic Link Library

In the *Dynamic link library* box you type the MQ Connector's file name for the particular platform shown in the File name column in the table.

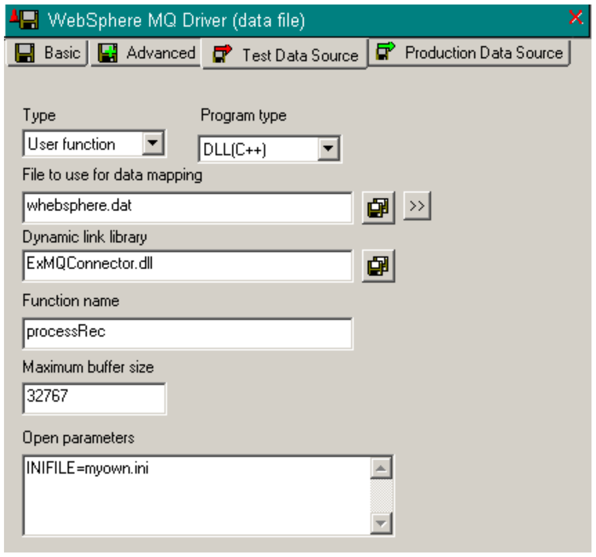
Function Name

In the *Function name* box you type the function for the particular platform shown in the 'Function name' column in the table.

Maximum Buffer Size

You can use 32767 as the *Maximum buffer size* on all platforms.

Example data file (for Windows use) - Test Data Source tab



Type: User function
Program type: DLL(C++)
File to use for data mapping: whebsphere.dat
Dynamic link library: ExMQConnector.dll
Function name: processRec
Maximum buffer size: 32767
Open parameters: INIFILE=myown.ini

Open Parameters

As with Driver files, you can type parameters you want the Engine to pass to DDA modules in the *Open parameters* text box.

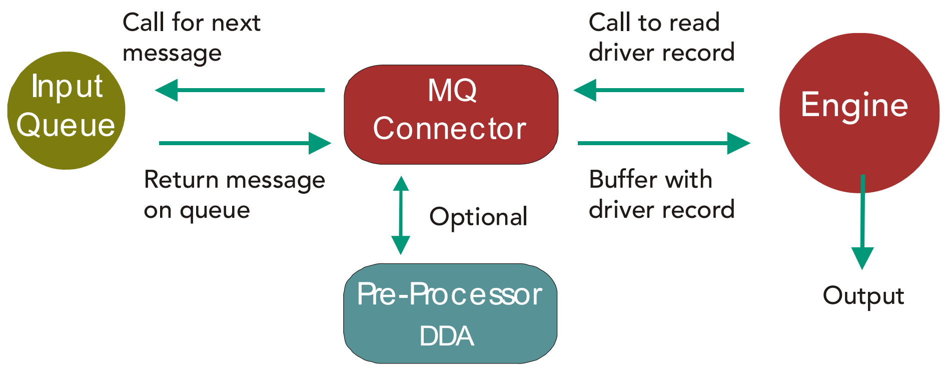
Options for Open Parameters

PARAMETER	DESCRIPTION	DEFAULT
INIFILE=anyfile.INI	Identifies an initialization file other than the default. Supply name and, if not located in the Exstream directory, the full path to the .ini file.	(see table at beginning of chapter for default)
PROCESSORSWITCH	Open parameters for the Pre/Post processor	No default

Overview of Engine Interaction

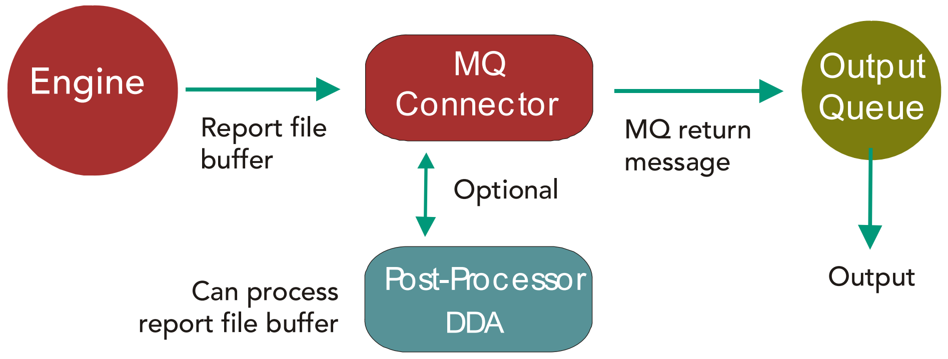
The following diagrams illustrate the interaction of the Dialogue Engine with the WebSphere MQ Connector.

View 1: Instance of WebSphere MQ Connector DDA between Input Queue and Engine



If the Engine specifies a SHUTDOWN, the message returns to the input queue for other Engines.

View 2: Instance of WebSphere MQ Connector DDA to support Report file



WebSphere MQ Connector Engine Initialization

:: Chapter 4

Engine Startup

On startup, the system reads an Initialization file you create. By default:

- This ini file must be in the current directory, but a path is supported.
- On Windows and UNIX the default file is EXMQS.INI.
- On OS/390 and z/OS the default file is DD:EXMQSINI.

Example Initialization File

```
QUEUEMGR=(Your name)
INPUTQUEUE=(Your queuename)
OUTPUTQUEUE=(Your queuename)
INPUTBUFFERSIZE=400000
MESSAGEFILE=EXMSG.txt
CONVERT=n
MSGFORMAT=NONE
SYNCPOINT=y
EXPIRYSHUTDOWN=20
TRACE=y
```

EXMQS.INI Parameter Options

Use the following parameters in your INI file.

Initialization parameter options

PARAMETER	DESCRIPTION	DEFAULT
CONVERT=Y/N	Message is converted into the format required by the receiving system before transmission.	MQGMO_CONVERT
CORRELATE=MSG/CID	Correlates input and output messages based on IDs. **MSG** - Passes the input MSGID to the CORRELID field of the output message. **CID** - Passes the input CORRELID to the CORRELID field of the output message. NOTE: Only valid when creating one output file per input record	NONE — optional
DDAPOSTPROCESSOR= ProgramName, FunctionName, ProgramType [,MaxBufferSize]	Identifies the processor program to call after receiving an input message from the Engine before sending to output queue. If the DDA returns a buffer, then the buffer is returned to the Engine. If it returns an EOF return then MQGET is called on the input buffer prior to returning to the Engine. Example: `DDAPOSTPROCESSOR= PostProcessor.so.1, procBuffer,DLL, 50000`	NONE — optional
DDAPREPROCESSOR= ProgramName, FunctionName, ProgramType [,MaxBufferSize]	Identifies the processor program to call after receiving an output message from the Engine. Processing occurs before sending an MQ report message to the OUTPUT QUEUE. Example: `DDAPREPROCESSOR= PreProcessor.so.1,procBuff,D LL`	NONE — optional
EXITON= #[,#,#,...]	Enables you to specify certain WebSphere MQ reason codes returned from MQGET calls on which they want MQ Connector to exit and shutdown.	NONE
EXITON= WRITEERROR	If MQPUT returns any failure reason code then the MQ Connector shuts down the Dialogue Engine.	

Initialization parameter options

Parameter	Description	Default
EXPIRYSHUTDOWN=#	Number of seconds the DDA SHUTDOWN message stays in queue. When the MQ Connector receives a shutdown message (MQ message with a priority of9 and a body of "SHUTDOWN"), the DDA posts a copy of that shutdown message back onto the queue so other Engines reading from the same queue also receive the shutdown request. This parameter indicates the expiry property of the copied shutdown message. If you are running only one Engine, set to zero. Otherwise, you have to wait for the copy that was placed in the queue to expire before you can restart the Engine.	60
FILETYPE=LOOKUP/ DRIVER	Identifies input file type. When using DDA Reference files, you must specify FILETYPE=LOOKUP.	DRIVER
INPUTBUFFERSIZE= ######	Specifies the size of the input buffer to be used in getting the MQ input message.	1/2 MB (524288 bytes)
INPUTQUEUE= ANYQUEUE1	Identifies the MQ Series input queue.	NONE — required
LOOKUP= CONFIRM/CORREL	Identifies lookup option for retrieving data.	CONFIRM
LOOKUPWAITINTERVAL =#	Number of seconds the MQ Connector waits for a lookup message to be available on the reply-to-queue.	10
MAXINACTIVITYCOUNT =#	Number of "get message" sequential fails before shutting down the transaction Engine.	0 (unlimited)
MESSAGEFILE=anyfile. txt	Name of message file for DDA messages. Output is limited to every 100th message after 100 consecutive messages. When the MQ message file exceeds 100,000,000 bytes, it is re-initialized. This prevents the file from growing too quickly and taking up available disk space.	exmqsmsg.txt

Initialization parameter options

Parameter	Description	Default
MSGFORMAT= formatname/NONE	Applies an IBM reserved or user-defined format name to the message descriptor format field to inform the receiver of the type of data in the message. This option affects all messages the connector sends. Alphanumeric characters known to the queue manager's character set can be used, up to eight letters (the connector pads any that are less than eight characters with spaces and truncates any format name longer than eight characters). Without this option in the INI file, the data type defaults to MQSTR (string). You can enter NONE (signifies no format name) to set the data type as undefined. One use of this option is to avoid changing output file messages from EBCDIC to ASCII characters.	MQSTR
OUTPUTQUEUE= ANYQUEUE2	Identifies the MQ Series output queue.	NONE — **required**
PERSISTENCE=Y/N	Specifies message persistence option to be used on "put".	MQPER_PERSISTENCE_AS_Q_DEF
QUEUEMGR= ANY.QUEUE.MANAGER	Identifies the MQ Series queue manager.	NONE — **required**

RECONNECT=n,m,i	Defines how the system responds to broken connections.
	
	n - System waits for these many MQRC_CONNECTION_BROKEN errors to occur in a session before taking action. The default is zero. (By default, the system makes no attempt to reconnect.)
	
	m - Number of times the system attempts to reconnect. The system stops after this number.
	
	i - Number of seconds between each reconnection attempt.
		n=0
		m=0
		i=10

Tip

Refer to the *WebSphere MQ Application Programming Reference Manual* from IBM for a list of the queue manager's built-in format names.

Initialization parameter options

Parameter	**Description**	**Default**
RECORD=FIXED, #/ TERMINATED, HEX	Signals the MQ Connector to buffer input messages according to a record format. **FIXED** - Buffer fixed records of specified length in # bytes. **TERMINATED** - Buffer terminated records according to specified hex termination string. Example: ODOA is hex code for CR/LF for native ASCII platforms. **NOTE**: This option is only necessary when sending multiple records per message through the input queue.	**NONE — optional**
REPLYTOQUEUE= ANYQUEUE	Identifies the MQ Series reply-to queue for lookup-type messages. NOTE: Only available with LOOKUP=CORREL	**NONE — optional**
SYNCPOINT=Y/N	Get message with/without syncpoint control.	**MQGMO_NO_SYNC POINT**
THROTTLEPROC=anypr ogram	Program to call to determine if DDA should sleep. Return value is zero to continue or a number of seconds to sleep. Unless you are using an external program to pace the Engine, do not include this parameter in the INI file. If you specify **Throttleproc=0** in the INI file, it attempts to run a process called **0**. Alternatively, you can just put **Throttleproc=**.	**NONE — optional**
TRACE=Y/N/L	Enable output trace messages to be written to the debug file. NOTE: L creates a log file of the trace ('MQConn_Log.dat').	**N**
TRACEFILE=FILENAME	Enables you to specify a trace log filename. NOTE: Only used with **TRACE=L**.	**MQConn_Trace.dat**

Initialization parameter options

Parameter	Description	Default
TRANSMANAGE	Enables Dialogue to perform transaction management. When included in the .ini file, Dialogue verifies, at the end of customer, if there are any MQ Connector calling errors. If there are none, then the connector commits the unit of work. If there is an error, then the MQ Connector backs out of the unit of work (the original set of customer information remains at the data source).	None - optional
WAITINTERVAL=#	Amount of time to wait for a message to be available on the input queue (in seconds). This indicates the polling interval for the MQGET. Each time the MQ Connector reads, it calls MQGET with this Waitinterval. MQGET waits for a message to arrive for up to Waitinterval seconds before returning failure. NOTE: This parameter is more useful if you are not in transaction mode.	10

WebSphere MQ Connector Queues

:: Chapter 5

Queue Specifications

Control Messages

All control messages have a priority of 9. Only control messages use that priority. All print messages must have a priority less than 9. The following control messages can be placed on the PRINTOUTPUT queue for the Exstream DDA module to process:

- **`STOP`** - Instructs the DDA to shutdown. The DDA is responsible for placing the message back on the queue, so other active DDA can receive the STOP message. The message is added to the queue with an expiration of 60 seconds. The STOP Operator clears the message before restarting the Exstream Engines, if the Engines need to be restarted within 60 seconds of the last Engine shutting down.
- **`FLUSH`** - Instructs the Exstream Engine via the DDA Module to flush the message and report file.
- **`SLEEP #ss`** - Where ss is the number of seconds to sleep. The DDA module is required to place a SLEEP control message back on the queue so other currently running Exstream Engines can receive the same SLEEP message. The MsgID is appended to the SLEEP control message before the message is placed back on the queue, if this is the first DDA to receive the message. The Expiry time is be extracted from the Original Sleep Message if non-zero, decremented by a second and used as the Expiry time for the message added to the queue. If Expiry time is zero, the number of seconds in the SLEEP control statement is used as the Expiry time. All control messages are placed back in queue with a timeout so that all Engines reading from queue can receive them. For example, when multiple instances of the Engine are running, all must receive the shut-down command to stop processing.
- **`SHUTDOWN`** - Instructs the DDA to perform adelayed shutdown. The shutdown does not process until all other messages are cleared from the input queue. This occurs when the call to MQGET returns a 2033 code meaning there are no messages available on the queue. At that point, the shutdown occurs. As with a STOP, the DDA is responsible for placing the SHUTDOWN message back on the queue.

:: Technical Note

Expiry time is MQ-specific. A message expires after a default time of 60 seconds so that SHUTDOWN messages do not persist after the Engine restarts.

General Queue Operation

QUEUEMGR

The QUEUEMGR opens once and remains open for the execution of the currently running Exstream Engine. It closes on SHUTDOWN or STOP.

INPUTQUEUE

The INPUTQUEUE queue enables the Exstream MQ Connector Module to receive print requests and control messages. In addition, it is used for lookup messages and requests. The INPUTQUEUE queue opens if specified in the INI file. The queue remains open for the execution of the currently running Engine. It closes on SHUTDOWN or STOP.

For lookup requests, the MQ Connector Module sends a lookup key to the input queue in one of two ways. If lookup is set to CONFIRM, then the message is sent and the module waits for lookup data to be returned from the input queue with confirmation. If lookup is set to CORREL (correlation ID), then the MQ Connector Module can wait for data to be returned from a specified reply-to queue or wait for the return from the same input queue. In this case, its return data is identified because its message ID is that of the lookup request message correlation ID.

OUTPUTQUEUE

The OUTPUTQUEUE queue enables the MQ Connector to send messages (return MQ messages). This queue opens for output if specified in the INI file and remains open for the execution of the currently running Engine. It closes on SHUTDOWN or STOP of the currently running Exstream Engine.

Reference Files and Dynamic Data Access (DDA)

:: Chapter 6

Reference File Sequence and Methods

Use the following description as a guide to how to create DDA routines to use the MQ queue manager to support Reference file lookups.

DDA Reference files perform dynamic lookups based on given key data, in the following sequence:

1. The DDA module receives an initial open call. At this point, the open parameters are parsed and the options are set according to the given Initialization file. Then the input queue opens for get/put access.
2. For each lookup, the DDA module is called with an access type of SEEK and passed the lookup key. At this point, there are two options for the lookups:
   – Default Method - The key is put on the input queue with the delivery confirmation option specified. This confirmation message tells the MQ Series queue manager to create a report message and place it on the reply to the queue. Based on a passed flag, this message has the message ID copied into the correlation ID field so it is easily recognizable by the MQ Connector input module (flag MQRO_COPY_MSG_ID_TO_CORREL_ID). After putting the key on the queue, the input module waits for a confirmation message by searching the reply to queue for a message with the known correlation ID. This “confirmation” message is only available when the key has been read from the queue by another application. After delivery is confirmed by the report message, the input module then grabs the next message off the queue (the returned data message). At this point, the input module returns the data length to the Dialogue Engine. If unsuccessful, an End of File (EOF) status returns.

   – Correlation ID Method - The key is put on the input queue as a message of type REQUEST, with option set to MQRO_COPY_MSG_ID_TO_CORREL_ID, and “reply-to-queue” set to either the specified REPLYTOQUEUE from the INI file or the same queue (INPUTQUEUE by default if no REPLYTOQUEUE is specified). After putting the key in the queue, the input module attempts to get the return message based on lookup by the known correlation ID with a wait interval of 10 seconds, (or the number of seconds specified with LOOKUPWAITINTERVAL=#). At this point, the input module returns the data length to the Dialogue Engine. If unsuccessful, an EOF status returns. You specify this method by the option LOOKUP=CORREL in the INI file, or in the *Open parameters* property in Dialogue.
3. If the seek is successful, the DDA module is called repeatedly with an access type READ. The lookup data is returned in the buffer and the length in the buffersize field, both of which returns to the Engine with a return code 0 (indicating data has been returned).
4. When there is no more data to be returned, the DDA module returns a code EOF to the Engine to signify that the lookup data is complete.
5. After all lookups have completed, the DDA module then passes the final close call (access type 5). The input queue closes.

Copyright

April 2005