WebSphere Transformation Extender

IBM

for Message Broker

Version 8.1

Note

Before using this information, be sure to read the general information in "Notices" on page 11.

30 June 2006

This edition of this document applies to WebSphere Transformation Extender, 8.1 and to all subsequent releases and modifications until otherwise indicated in new editions.

To send us your comments about this document, e-mail DTX_doc_feedback@us.ibm.com. We look forward to hearing from you.

Contents

Chapter 1. Transformation Extender for Message Broker overview

Transformation Extender for Message Broker is a message broker solution. It is a plug-in that extends the basic messaging facilities supplied by MQSeries by enabling you to route messages and manipulate their content, according to a set of rules that are defined using a graphical interface.

The Transformation Extender for Message Broker parser starts a map to transform an incoming byte stream into an element tree, while the plug-in node transforms one element tree into another. These maps can use resource adapters in other cards not occupied by the transformation to access the numerous resources supported.

An incoming MQSeries message triggers a message flow that consists of message processing nodes that transform and/or enhance the message on its way to an MQSeries queue or database. The message first gets transformed from its original byte stream into a hierarchical structure called an **Element Tree** that can be understood by other nodes in the flow. This transformation is performed by a **Message Parser**. Before it is sent to a queue it is transformed into a byte stream using the parser to perform the inverse operation.

The integrator provides a set of ready-made message parsers and processing nodes. It also provides a programming interface for writing custom plug-ins to extend the set of formats and transformations.

Since the number of formats supported by WebSphere MQ Integrator is limited, it is a perfect place for using a map.

System requirements

The minimum system requirements and operating system requirements for the Transformation Extender for Message Broker are detailed in the release notes (*install_dir*/readme_mb.txt).

Chapter 2. Using Transformation Extender for Message Broker

The WebSphere Business Integration Message Broker Plug-in provides an extension to the integrator product allowing it to work with messages in virtually any format by using a map to perform the transformation.

In addition to this main function, any resource adapter can be used inside the transformation map to access a database or some other supported resource, for example, to do a database lookup or send a message to a message queue.

The Transformation Extender for Message Broker consists of two components:
- ″Plug-in Parser″
- ″Plug-in Node″

The Plug-in parser performs two functions:
- Converts an input byte stream into an element tree. Started from MQInput node.
- Converts an element tree into a byte stream. Started from MQOutput node.

The Plug-in node converts one element tree into another.

Data structures

There are four different data structures: Byte Stream, Element Tree, XML Data, and XML Tree.

Byte stream

This data structure represents raw data received from or sent to an MQSeries message queue. It can be in any format or have no format (BLOB). The plug-in parser uses a map to `understand' this format in both MQInput and MQOutput nodes. This format is not used in WebSphere MQ Integrator message flow directly - it is first transformed into an **Element Tree**.

Element tree

This data structure is used by the integrator as an internal format for representing messages in the message flow. It is a hierarchical structure of elements (parent-child relationship). Each element can have only one parent and any number of children.

There are three types of elements:
- **Name element**: Has only one property - name. The Name element can have any number of children.
- **Name-Value element**: Has two properties - name and value. Value always has two properties - type and actual value. The Name-Value element can have any number of children.
- **Value element**: Has two properties - type and actual value. The Value element cannot have children - it must be a leaf.

XML data

This data format is used in the plug-in only. It is the format used to exchange data with a map. In the plug-in parser the map transforms byte stream into this format in an input node and this format into byte stream in an output node. In the plug-in node this format is used as both input and output data for the map.

There is one-to-one correspondence between this format and the element tree. The mapping between the tree elements and XML is given in the following table:

Element Tree	XML Representation
Name	*<Name>...</Name>*
Name-Value	*<Name* `type="`*type-id*`"`>*Value</Name>*
Value	`<Value type="type-id">`*Value*`</Value>`

Note: XML words in **bold** font represent keywords. Words in *italic* font are variables. Words in ***bold-italic*** font are variables whose values come from a predefined set.

The following table shows defined type-ids used in the XML:

Type-id	Value Format
`boolean`	*true/false*
`integer`	*9999...*
`real`	*...999.999...*
`decimal`	*...999.999...*
`char`	*text*
`time`	*HH24:MI:SS.FF*
`gmttime`	*HH24:MI:SS.FF*
`date`	*CCYY-MM-DD*
`timestamp`	*CCYY-MM-DD HH24:MI:SS.FF*
`gmttimestamp`	*CCYY-MM-DD HH24:MI:SS.FF*

XML tree

This data structure is internal to the plug-in. It is neither visible nor accessible from outside. It represents a logical view of the XML character stream used in constructing and analyzing Element Trees.

Plug-in parser

The Transformation Extender for Message Broker parser component is started from an MQInput or MQOutput node - it does not have a node of its own. In an MQInput node, the parser is specified by setting the **Message Domain** field and the **Message Type** field to the format alias as defined in the **dtxwmqi.ini** plug-in initialization file. In an MQOutput node, the type of the incoming message determines the parser started. If the message specifies the parser then the format alias must be specified in the message **Properties** header in the **MessageType** field.

Plug-in node

The Transformation Extender for Message Broker node is a separate component in the message flow. It has its own GUI symbol in the Control Center. It has one input and two output terminals: out and failure.

The Plug-in node converts a message into XML and passes it to a map. The result of the map is also an XML stream that is then converted to a new message that is propagated to the out terminal. In case of a map error, the message is propagated to the failure terminal together with error details.

If the processing in the node was successful, the resulting message is routed to the out terminal. In cases of failure, the original message is routed to the failure terminal together with an additional error element added as a last child in the element tree.

A map can be created to format this into any format using a separate plug-in node. The input format for such a map looks like this:

```
<NodeErrorInfo>
    <ErrorCode type="integer">-1001
   </ErrorCode>
    <ErrorMessage type="char">Map failed - Could not open map (3)
   </ErrorMessage>
    <MapName type="char">c:\maps\map1\xml2xml.mmc
   </MapName>
</NodeErrorInfo>
```

XML words in **bold** font represent keywords. Words in *italic* font are variables. Words in ***bold-italic*** font are variables whose values come from a predefined set.

The following table shows possible error codes and their associated meanings.

Error Code	Error Message	Comment
-1001	Map failed - <message> (<code>)	1
-1008	Load library failed - runmer32.dll	Windows
-1008	Load library failed - libplatapi.so	Solaris & AIX
-1008	Load library failed - libplatapi.sl	HP-UX
-1010	No data to process - Failed to get Root element	
-1010	No data to process - Failed to get Body element	
-1010	No data to process - Empty Body element	
-1011	Map returned invalid XML - <message> (<code>) at pos <pos>	

[1]See Error and Warning Messages in the Map Designer documentation for more information.

A message is propagated to the error terminal only in cases where something related to a map is wrong, such as library not found, map file not found, or map failed.

In other cases, an exception is thrown back to the broker instructing it to handle it. Every error condition is written to the plug-in trace file (see ″Plug-in Configuration″). In addition to this log file, the broker has its own log files where a plug-in error is also recorded.

Click on the **Basic** tab to set the plug-in node properties.

The following properties can be set for the Transformation Extender for Message Broker node:

- **Map** - Specifies the path to the executable map file to be invoked.
- **OutputCard** - Integer representing the output card used to return data to the plug-in.
- **OutputDomain** - Specifies the domain of the output message.
- **OutputSet** - The message set to which the resulting message belongs.
- **OutputType** - Specifies the format alias.
- **OutputFormat** - Specifies the format of the message.

Configuration

There are two items to configure for the Transformation Extender for Message Broker:

- trace
- message formats

The configuration is stored in the **dtxwmqi.ini** XML file. The name is fixed. The file is required at run time and needs to exist on each machine that is running the message broker.

The file is optional for systems that use the plug-in node only, but is required for those that use the plug-in parser. The file format is shown here:

```
<PlugIn>
    <Trace>
        <Node    FilePath="c:\n.log" Level="normal" Enabled="yes"
               Mode="normal"
      />
        <Parser  FilePath="c:\p.log" Level="normal" Enabled="yes"
               Mode="normal"
      />
    </Trace>
    <MessageTypes>
       <Type Name="M4FMT1" InputMapPath="c:\maps\map1\bin2xml.mmc"
          InputMapCard="1" OutputMapPath="c:\maps\map1\xml2bin.mmc"
          OutputMapCard="1"
      />
        <Type Name="M4FMT2" InputMapPath="c:\maps\map1\bin2xml.mmc"
            InputMapCard="1" OutputMapPath="c:\maps\map1\xml2bin.mmc"
        OutputMapCard="1"
    />
  </MessageTypes>
</PlugIn>
```

Trace

Trace can be specified for both the node and the parser separately.

The following table shows the available Trace options and their descriptions:

Option	**Description**
FilePath	Full path name of the trace file. The Windows default name is **dtxwmqi.log**. The UNIX default name is <SYSTEM_TEMP>/**dtxwmqi.log** (usually /tmp or /var/tmp).
Level	Specifies level of details shown in the trace file. Can be one of the following. The default setting is *normal*: • *normal* - adapter activity and error log • *debug1* - *normal* plus data trees (element trees and XML trees) • *debug2* - *debug1* plus data passed to and from a map • *debug3* - *debug2* plus XML data is printed in binary format
Enabled	Switch can be *yes* or *no*. The default setting is *no*.
Mode	Set to *normal* or *append*. The default setting is *normal*.

Message types

The Plug-in parser supports many formats. Each format is supported by a set of two maps. One is used to convert raw data into XML format and the other to perform the inverse operation.

This section of the initialization file associates each set of maps with a unique alias, thus registering the format with the plug-in. The alias is used later in an MQInput node to refer to a certain set of maps.

The following table shows the available Message Type options and their descriptions:

Option	**Description**
Name	Message type name or alias specified in the Topic field in the MQInput node. They link an alias with a compiled map that is invoked to parse the given format. The default is *empty string*.
InputMapPath	Compiled map file path. The map converts byte stream into XML. The default is *empty string.*
InputMapCard	Output card number used to echo data from the input map back to the plug-in. The default is one (1).
OutputMapPath	Compiled map file path. The map converts XML into byte stream. The default is *empty string.*

OutputMapCard
: Output card number used to echo data from the output map back to the plug-in. The default is one (1).

Note: Data from the plug-in is always echoed to input card number one in a map.

Example files

Example files are located in the installation directory (../examples/dk/wmqi).

Chapter 3. Return codes and messages

Return codes and messages are returned when the particular activity completes. Return codes and messages may also be recorded as specified in the audit logs, trace files, execution summary files, and so on.

An input XML message is routed to theTransformation Extender for Message Broker node. *Out* and *failure* terminals are attached to Trace nodes. The map is forced to fail either by supplying the wrong format, or by removing the **.mmc** file. The purpose is to show that the message was propagated to the *failure* terminal and it contains the right information and format. This is verified by analyzing the log file generated by the trace node attached to the *failure* terminal. A message should not be propagated to the *out* terminal - the other trace file should not be generated.

Messages

The following is a listing of all the codes and messages that can be returned as a result of using the Transformation Extender for Message Broker.

The following error conditions are defined for the plug-in:

Return Code	Message
-1001	Map failed.
-1002	WMQI function failed.
-1003	Unknown type code.
-1004	Generating elements from XML failed.
-1005	Unknown message format.
-1006	Element navigation failed.
-1007	Unknown exception occurred.
-1008	Failed to load library.
-1009	Error in XML element tree.
-1010	No data to process.
-1011	Map returned invalid XML.

And the following warnings:

Code	Message
1001	Invalid atrributename.
1002	Invalid attribute index.

These error codes are propagated to the error terminal of the plug-in node, written to the plug-in log file, or reported back to the broker depending on the context and upon the error itself.

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan

The following paragraph does not apply to the United Kingdom or any other 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 NON-INFRINGEMENT, 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.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
577 Airport Blvd., Suite 800
Burlingame, CA 94010
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Programming interface information

Programming interface information, if provided, is intended to help you create application software using this program.

General-use programming interfaces allow you to write application software that obtain the services of this program's tools.

However, this information may also contain diagnosis, modification, and tuning information. Diagnosis, modification and tuning information is provided to help you debug your application software.

Warning: Do not use this diagnosis, modification, and tuning information as a programming interface because it is subject to change.

Trademarks and service marks

WebSphere Transformation Extender, Version 8.1

Index