



IBM Software Group

WebSphere® Process Server V6.0.1
WebSphere® Integration Developer V6.0.1
WebSphere® Enterprise Service Bus V6.0.1

Message Logger Mediation Primitive



@business on demand.

© 2006 IBM Corporation
Updated May 1, 2006

This presentation will provide a detailed look at the Message Logger mediation primitive.

Goals

- Understand the Message Logger mediation primitive



Message Logger

- ▶ Overview of function
- ▶ Use of terminals
- ▶ Definition of properties
- ▶ Message log database
- ▶ Error handling
- ▶ Example usage

The goal of this presentation is to provide you with a full understanding of the Message Logger mediation primitive. It is assumed that you are already familiar with the material presented in the **Mediation Primitive Common Details** presentation, which serves as a base for understanding mediation primitives in general. An overview of the Message Logger is also presented along with information about the primitive's use of terminals, its properties and error handling characteristics. Finally, Information about configuring a message log database and an example usage of a Message Logger is provided.

Message Logger – Overview of Function

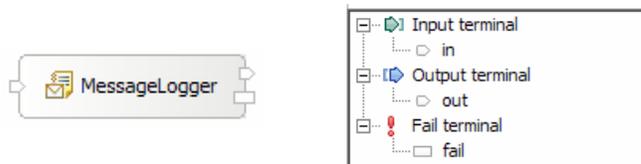
- Writes content of the message to a database
 - ▶ Message written in XML format
 - ▶ All or part of the Service Message Object (SMO) can be written
 - Configured using XPath to identify that portion of the message to write
 - Default is the message payload (/body)
- Pre-configured cloudscape database
 - ▶ Usable for single server installations
 - ▶ Database location: <profile_dir>/esb/databases/EsbLogMedDB
 - ▶ Includes a pre-configured datasource
- The SMO is not updated



The purpose of a Message Logger is to write a record to a database containing some content from the service message object. This content is written in XML format and can consist of all or part of the SMO. The message logger has a property, which is an XPath expression identifying what part of the SMO should be logged. The default value for the property is the message payload or body. A pre-configured cloudscape database, which is found at the location shown in the slide, is used for logging messages. There is also a pre-configured datasource in the server runtime environment that identifies this database. This pre-configured database is only suitable for use in a single server environment. Considerations for the network deployment environment are addressed later in this presentation. The SMO is not updated by the message logger.

Message Logger – Terminals

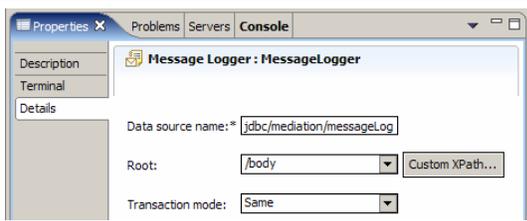
- Terminals:
 - ▶ Input terminal
 - ▶ One Output terminal
 - ▶ Fail terminal
- All terminals must be for the same message type



The Message Logger primitive has one input terminal, one output terminal and a fail terminal. The output terminal must be for the same message type as the input terminal because the message logger primitive does not modify the message body. Shown here is a message logger primitive with its terminals and the terminals as seen in the properties view.

IBM Software Group IBM

Message Logger – Properties



- Data source name
 - ▶ The JNDI name of the datasource identifying the log database
 - ▶ `jdbc/mediation/messageLog` is the default value
 - ▶ Pre-configured datasource uses this JNDI name
- Root
 - ▶ XPath expression defining portion of SMO to log
 - ▶ XPath expression can identify any element or portion of the SMO
 - ▶ `/body` is the default (the message payload)
- Transaction mode:
 - ▶ `same` – commit database update within the flow's transaction (default)
 - ▶ `new` – commit database update immediately using a new transaction

5

Message Logger Mediation Primitive ©2006 IBM Corporation

In the upper right is a screenshot of the Details tab from the Properties view for a Message Logger showing the following properties:

Data source name property is a JNDI name used to lookup the datasource that identifies the database in which the message will be logged. When creating a new Message Logger, this property is set to a default value of `jdbc/mediation/messageLog` which also happens to be the JNDI name for the pre-configured datasource identifying the pre-configured database.

Root contains an XPath expression that identifies the portion of the SMO that is to be logged. When creating a new Message Logger, this property is set to a default value of `/body` indicating the message payload should be logged. Using the dropdown box, this value can be set to indicate logging of the entire SMO, the context, the headers or the body. In addition, the **Custom...** button opens an XPath Expression Builder which can be used to identify any portion or element within the SMO.

Transaction mode determines when the update to the message log database will be committed. The default value is **same** which means that the update will be committed as part of the transaction configured for the mediation flow, whereas the value of **new** indicates the update should be committed immediately using a new transaction.

Message Logger – User Configured Database

- The pre-configured database is not usable when:
 - ▶ In a Network Deployment (multiple server) environment
 - ▶ A database other than cloudscape is required
- Creating your own message log database
 - ▶ Data definition language files have been provided:
 - <install_root>/util/EsbLoggerMediation/<database_type>/Table.ddl
 - ▶ Supported Databases: Cloudscape, DB2®, Informix, MS-SQL, Oracle, Sybase
- Approaches to using your own message log database
 - ▶ Option 1:
 - Delete pre-configured datasource
 - Create datasource with default JNDI name
 - Use default name in Message Logger primitives
 - ▶ Option 2:
 - Create datasource with its own unique JNDI name
 - Configure Message Logger primitives to use the unique name



You can configure the Message Logger primitive to use a database other than the pre-configured database, and this is required in the network deployment environment where you have multiple servers using the same database. It might also be applicable to some single server situations where you have requirements to use a particular database such as DB2 or Oracle. The Data Definition Language needed to create the message logger database has been provided and can be found in the directory indicated on this slide. There are separate Table.ddl files for Cloudscape, DB2, Informix, MS-SQL, Oracle and Sybase. When using your own database, it is best to have a strategy on how you plan to configure your datasource and your Message Loggers. The first option is to delete the pre-configured datasource and create a new datasource for your database that uses the default JNDI name. This approach allows you to continue to use the default JNDI name for each of your Message Logger primitives. The second option is to create a new datasource with its own unique JNDI name and configure your Message Loggers to use the new JNDI name. The first approach makes the configuration of your Message Loggers easier and less prone to error, while the second approach is useful if you would like to have more than one message logger database in your environment.

Message Logger – Retrieval of Log Data

- Possible message log data usage:
 - ▶ Audit trails of enterprise service bus message handling
 - ▶ Data mining of business data contained in messages
 - ▶ Statistics of service usage
- No tools or applications are provided
 - ▶ Users must create their own tools
- Message log table schema:

Column	Type	Key	Description
TimeStamp	TIMESTAMP	Y	UTC timestamp of when message was logged
MessageID	VARCHAR	Y	Message ID from the Service Message Object
MediationName	VARCHAR	Y	Mediation primitive that logged the message
ModuleName	VARCHAR	N	Mediation module containing mediation primitive
Message	CLOB	N	Requested portion of Service Message Object in XML
Version	VARCHAR	N	The Service Message Object version

There are many possible uses of the log data contained in the message log database. For example, the log might be used to maintain an audit trail of the message handling within the enterprise service bus. Another possibility would be to do some data mining of business data that is contained in the messages. A third possibility might be to compute statistics about service usage through the bus. Although there are these and many other possible uses of the log data, there is no tooling provided to extract or analyze the data contained in the log. You must provide your own applications for extracting and analyzing the data based on your own requirements. The table shows the schema for the message log database. There is a timestamp containing the time the message was logged, a unique message ID, and the name of the message logger primitive that wrote the log, which together form the key. The additional fields are the name of the mediation module within which the message logger primitive was running, the message content in XML format as defined by the Root property of the message logger, and finally the SMO version associated with this log message. You will need to understand this schema in order to develop an application to retrieve and analyze the log message data.

Message Logger – Error Processing

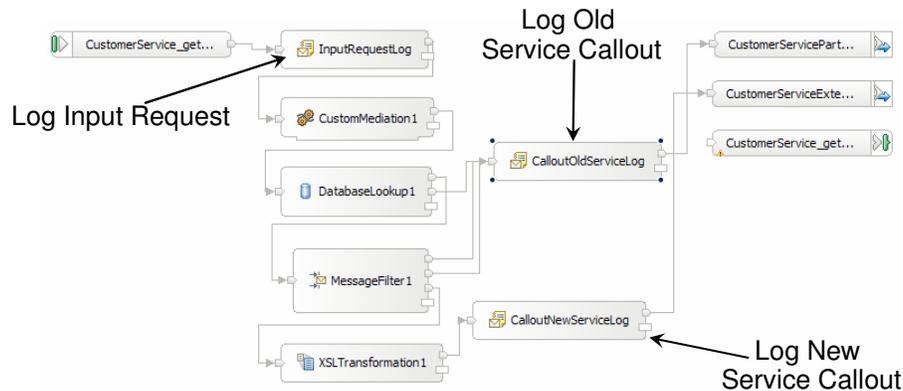
- **MediationRuntimeException** thrown for:
 - ▶ Incorrect JNDI name for datasource
 - ▶ Root XPath value of “null”
- **MediationConfigurationException** (Fail terminal flow)
 - ▶ Problems accessing database
- **Failure of flow downstream of Message Logger primitive**
 - ▶ Transaction mode = new – message is logged
 - ▶ Transaction mode = same - message is rolled back if the mediation flow has been configured to run in a global transaction
- **Root XPath value not found in Service Message Object**
 - ▶ Not considered an error condition
 - ▶ The log is written with the Message field empty



The error processing details and considerations are examined in this slide. A `MediationRuntimeException` will be thrown for an incorrect JNDI name for the datasource and also for the case where the root property has been specified as a null XPath value. A `MediationConfigurationException` occurs for any kind of problems accessing the message log database. If the Fail terminal is wired, that flow will be followed rather than the exception being thrown. When there is a failure in the mediation flow downstream from the message logger, the message will remain logged to the database unless the transaction mode has been set to same and the flow has been configured to run in a global transaction. When that is the case, the log of the message will be rolled back as part of the global transaction rollback. It is not considered an error condition when the root property contains an XPath expression that is not found in the Service Message Object. The log message will still be written but the message field will be empty.

Message Logger – Example Usage

- Example – Use log to enable statistics of service usage
 - ▶ Mediation routes requests to an old or new service
 - ▶ Want to be able to track usage of the old vs. new service
 - ▶ Log the input request and log the callouts to the old and new services



This slide illustrates a possible use of the message logger primitive. The requirement is to enable the keeping of statistics about service usage as requests flow through the enterprise service bus. The scenario involves a flow where the requestor uses an interface that is for the original service provider but there is now also a new service provider with a new interface. Based on some criteria involving the values in the message body a decision is made to use the old or the new provider. To meet this requirement, appropriate log messages are written so that statistics can be computed from the log database regarding usage of the old and new services. Looking at the flow diagram, you will see that there is a message logger at the beginning of the flow that records every request. There is also a message logger prior to the callout to each of the service providers, so for any given request, there will be two messages logged. Not shown in this flow is a message logger on the response side which logs every response as it goes back to the requestor. Given this set of logs, it is possible to write an application that computes service usage statistics for the old and new versions of the service.

Message Logger – Example Usage

- Example log data
 - Input request, service callout (old or new) and response all logged

Message IDs on request flow the same
Response flow has its own unique message ID

Second Request/Response
First Request/Response

Callout to Old Service
Callout to New Service

P	TIMESTAMP	MESSAGEID	MEDIATIONNAME	MODULENAME	MESSAGE
30	2006-01-12 13:15:46	d9800bc0-0801-0000-0080-9f2d60343154	InputRequestLog	CustomerRoutingMediation	<?xml version="1.0
31	2006-01-12 13:15:47	d9800bc0-0801-0000-0080-9f2d60343154	CalloutOldServiceLog	CustomerRoutingMediation	<?xml version="1.0
32	2006-01-12 13:15:47	d9870bc0-0801-0000-0080-9f2d60343154	ResponseLog	CustomerRoutingMediation	<?xml version="1.0
33	2006-01-12 13:15:54	56a00bc0-0801-0000-0080-9f2d60343154	InputRequestLog	CustomerRoutingMediation	<?xml version="1.0
34	2006-01-12 13:15:54	56a00bc0-0801-0000-0080-9f2d60343154	CalloutNewServiceLog	CustomerRoutingMediation	<?xml version="1.0
35	2006-01-12 13:15:54	3da10bc0-0801-0000-0080-9f2d60343154	ResponseLog	CustomerRoutingMediation	<?xml version="1.0

Message IDs on request flow the same even when message type changes during the flow

Message Logger Mediation Primitive ©2006 IBM Corporation 10

This screenshot from CView shows the contents of the message log as it appears in the cloudscape database. It shows two service requests, the first of which used the old service provider and the second of which used the new service provider. There are several things to take note of in this screenshot:

- Notice the columns for the database including the timestamp, message id, mediation name, module name, message and SMO version, the last of which is not shown.
- There are three logs for each request, which represent the incoming message, the callout to the old or new service and the response. You can see by the mediation name column that the first request went to the old service and the second request went to the new service.
- The message ids are also interesting to examine. You can see that on the first request, the first and second logs, both of which are on the request flow, have the same message id, whereas the third log for the response flow has a different id. From this you see that the request and response will have unique message ids. On the second request, the one that uses the new service, the SMO body was changed during the request flow by an XSLT primitive to match the new service interface. However, the message ids are still the same, showing that the unique message id is associated with an SMO throughout a flow even if the structure of the SMO body is modified by an XSL Transformation.

Summary

- Examined the Message Logger mediation primitive



Message Logger

- ▶ Overview of function
- ▶ Use of terminals
- ▶ Definition of properties
- ▶ Message log database
- ▶ Error handling
- ▶ Example usage

In summary, this presentation provided details regarding the Message Logger mediation primitive and an overview of the Message Logger along with information about the primitive's use of terminals, its properties and error handling characteristics. Finally, information about configuring a message log database was provided and an example usage of a Message Logger was presented along with a look at an example message log.

Trademarks, Copyrights, and Disclaimers

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

IBM	CICS	IMS	MQSeries	Tivoli
IBM (logo)	Cloudscape	Informix	OS/390	WebSphere
e(logo)business	DB2	iSeries	OS/400	xSeries
AlX	DB2 Universal Database	Lotus	pSeries	zSeries

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are registered trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel, ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a registered trademark of Linus Torvalds.

Other company, product and service names may be trademarks or service marks of others.

Product data has been reviewed for accuracy as of the date of initial publication. Product data is subject to change without notice. This document could include technical inaccuracies or typographical errors. IBM may make improvements and/or changes in the product(s) and/or program(s) described herein at any time without notice. Any statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business. Any reference to an IBM Program Product in this document is not intended to state or imply that only that program product may be used. Any functionally equivalent program, that does not infringe IBM's intellectual property rights, may be used instead.

Information is provided "AS IS" without warranty of any kind. THE INFORMATION PROVIDED IN THIS DOCUMENT IS DISTRIBUTED "AS IS" WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IBM EXPRESSLY DISCLAIMS ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. IBM shall have no responsibility to update this information. IBM products are warranted, if at all, according to the terms and conditions of the agreements (e.g., IBM Customer Agreement, Statement of Limited Warranty, International Program License Agreement, etc.) under which they are provided. 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 in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. IBM makes no representations or warranties, express or implied, regarding non-IBM products and services.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents or copyrights. Inquiries regarding patent or copyright licenses should be made, in writing, to:

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

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.

© Copyright International Business Machines Corporation 2004,2005. All rights reserved.

Note to U.S. Government Users - Documentation related to restricted rights-Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract and IBM Corp.