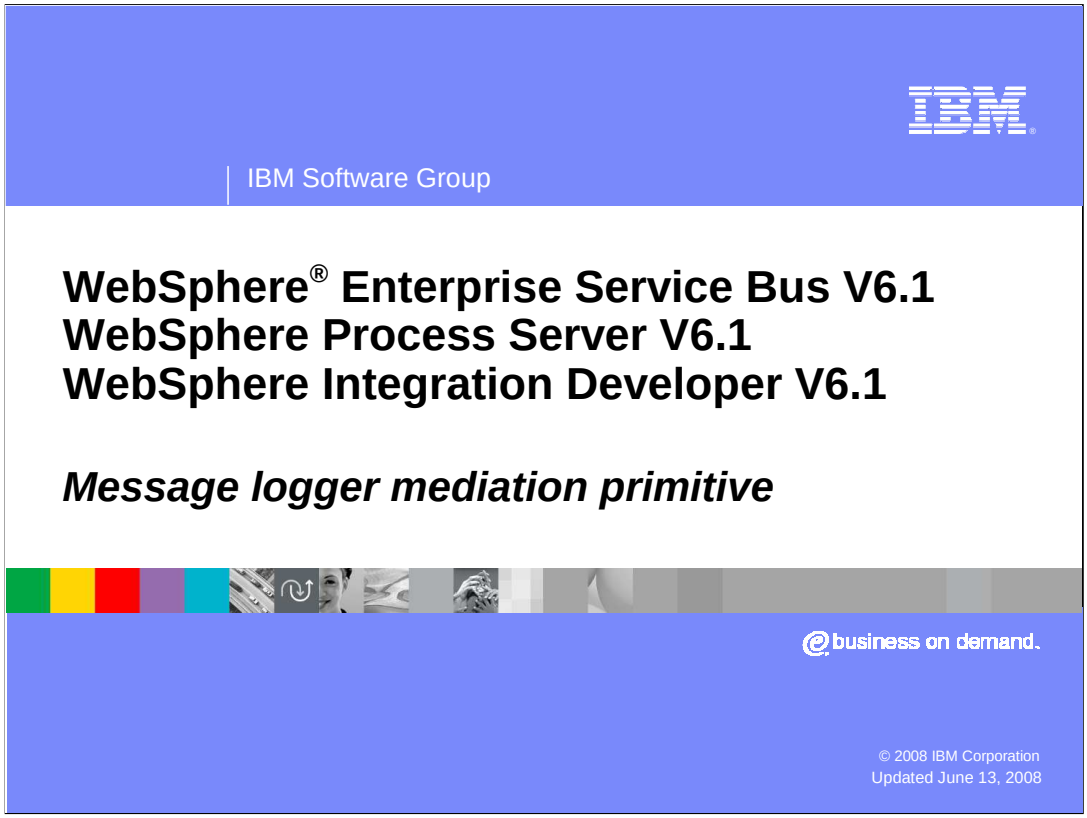


This presentation provides a detailed look at the message logger mediation primitive.

Goals

- Understand the message logger mediation primitive

Message logger

- Overview of function
- Use of terminals
- Definition of properties
- Message log database table
- Migration from previous releases
- Retrieval and usage of log data
- Error handling
- Example usage

The goal of this presentation is to provide you with a full understanding of the message logger mediation primitive.

The presentation assumes that you are already familiar with the material presented in the **Mediation primitive common details** presentation and the **Common details – Promoted properties** presentation. These two presentations serve as a base for understanding mediation primitives in general.

In this presentation, an overview of the message logger is presented along with information about the primitive's use of terminals and its properties. The message logger makes use of a message log database table, which is described, along with some configuration options available to you. Since there are some changes from prior releases, migration considerations are addressed. A discussion of how you can retrieve and make use of the log data is presented, followed by some error handling information, and finally an example use of a message logger primitive is presented.

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)
- Database usage
  - Default location is the common database used by the server
    - This is a change between V6.0.2 and V6.1, which used a unique database
  - A pre-configured data source points to the common database
  - Message logger defaults to JNDI name of pre-configured data source
  - Configuration options allow use of another database, multiple databases or multiple tables within a database
- The SMO is not updated

The purpose of a message logger primitive is to write a record to a database, containing some of the content of the service message object, or SMO. 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 which part of the SMO is logged. The default value for the property is the message payload or body.

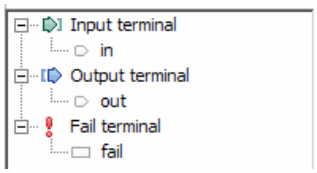
The common database, used by a WebSphere Process Server or WebSphere Enterprise Service Bus, is the default location for the database table used by the message logger. Note that this is a change in the V6.1 release. In V6.0.2 and earlier releases, there was a unique database created specifically as the default location for the database table used by the message logger primitive.

Similar to prior releases, there is a pre-configured data source in the server runtime environment that identifies the database containing the message log table. In V6.1, this data source continues to have the same JNDI name as previously, but has been changed to point to the common database. This JNDI name is used as the default JNDI name for the data source when a new message logger primitive is created. There are various configuration capabilities that allow for the use of other databases, multiple databases or multiple tables within a database.

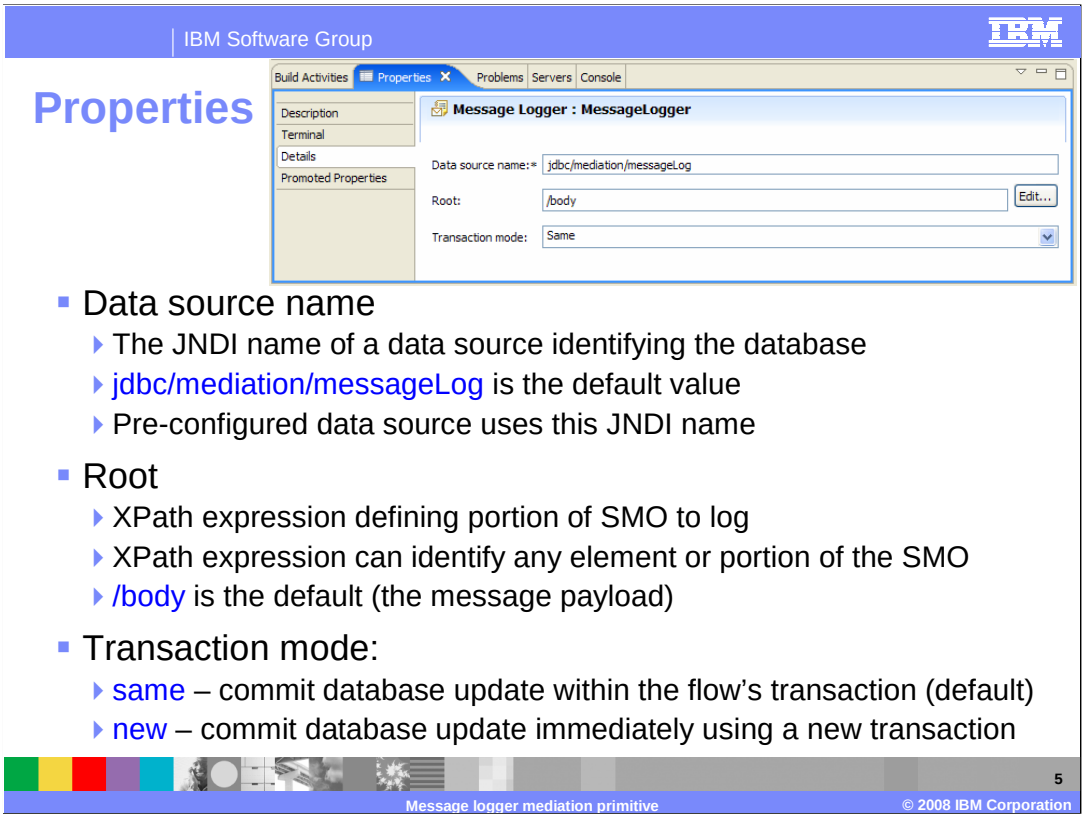
The message logger primitive does not update the contents of the SMO.

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.



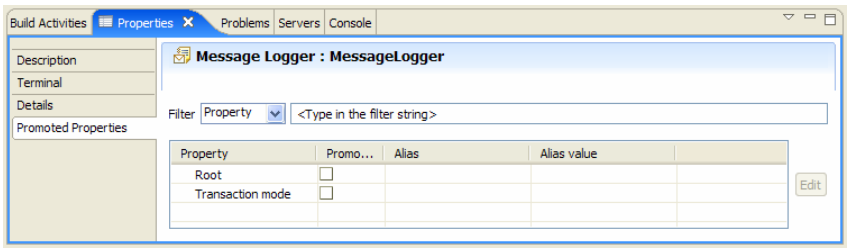
In the upper right corner of the slide is a screen capture showing the Details panel of the Properties view for a message logger primitive, showing the primitive's three properties.

The **Data source name** property is a JNDI name used to lookup the data source that identifies the database containing the table in which the messages are 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 data source identifying the common database.

The **Root** property contains an XPath expression that identifies the portion of the SMO that is included in the log message. 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, or just the context, just the headers or the body. In addition, the **Edit…** button opens an XPath Expression Builder which can be used to drill down and identify any portion or element within the SMO.

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

Promoted properties



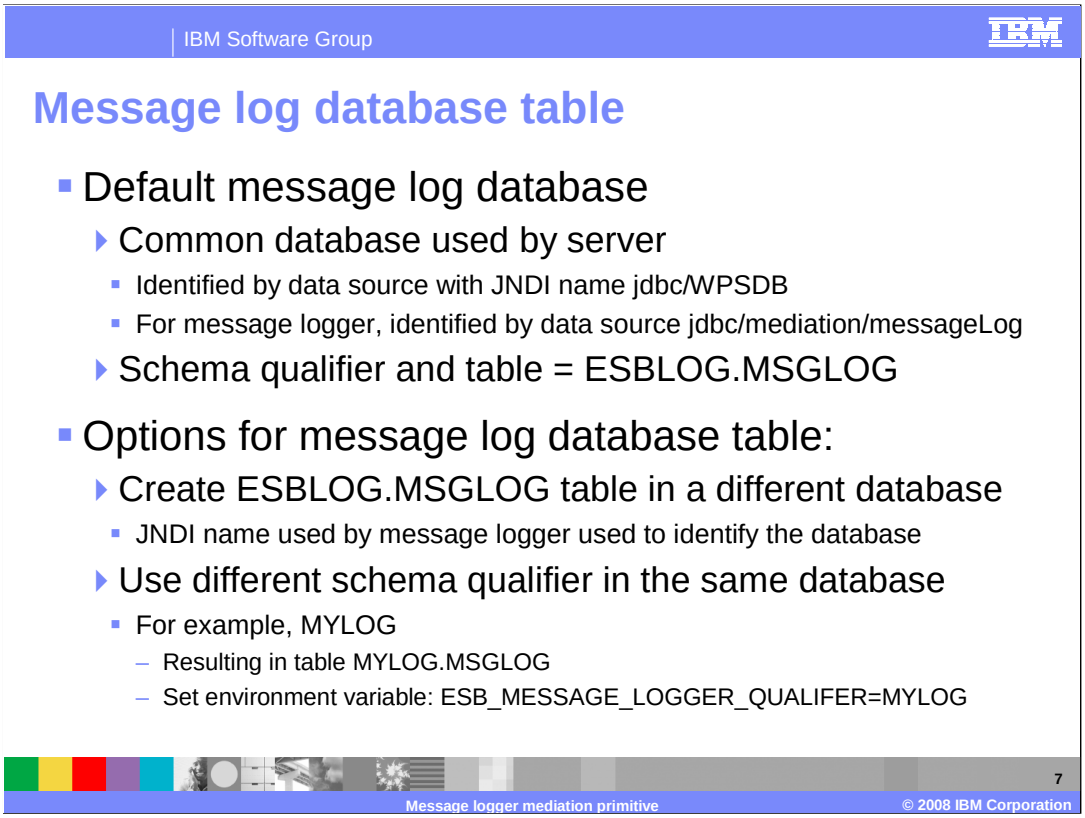
- Promotable
  - Root
  - Transaction mode
- Not Promotable
  - Data source name

This slide shows the Promoted Properties panel for the message logger.

The Data source name property contains the JNDI name of the database that is used for logging. The property is not promotable because the JNDI name gets associated with a resource reference in a generated EJB. Therefore, this requires a redeployment of the mediation application if it were to change.

The Root property and the Transaction mode property are both promotable.

Making the Root property promotable allows different portions of the SMO to be written to the log. This can be useful in a problem determination situation, where you might want to start writing the entire SMO to the log when trying to analyze a problem situation.



As indicated earlier, the message log table, by default, is contained in the common database used by a WebSphere Process Server or WebSphere Enterprise Service Bus. The server runtime environment provides a variety of options for the type of database used for the common database. In any case, whatever database is used, it is identified by a data source with a JNDI name of jdbc/WPSDB. The default data source use by the message logger identifies the same database and has the JNDI name of jdbc/mediation/messageLog. The message logger primitive writes into the database using the table named MSGLOG with a schema qualifier of ESBLOG.

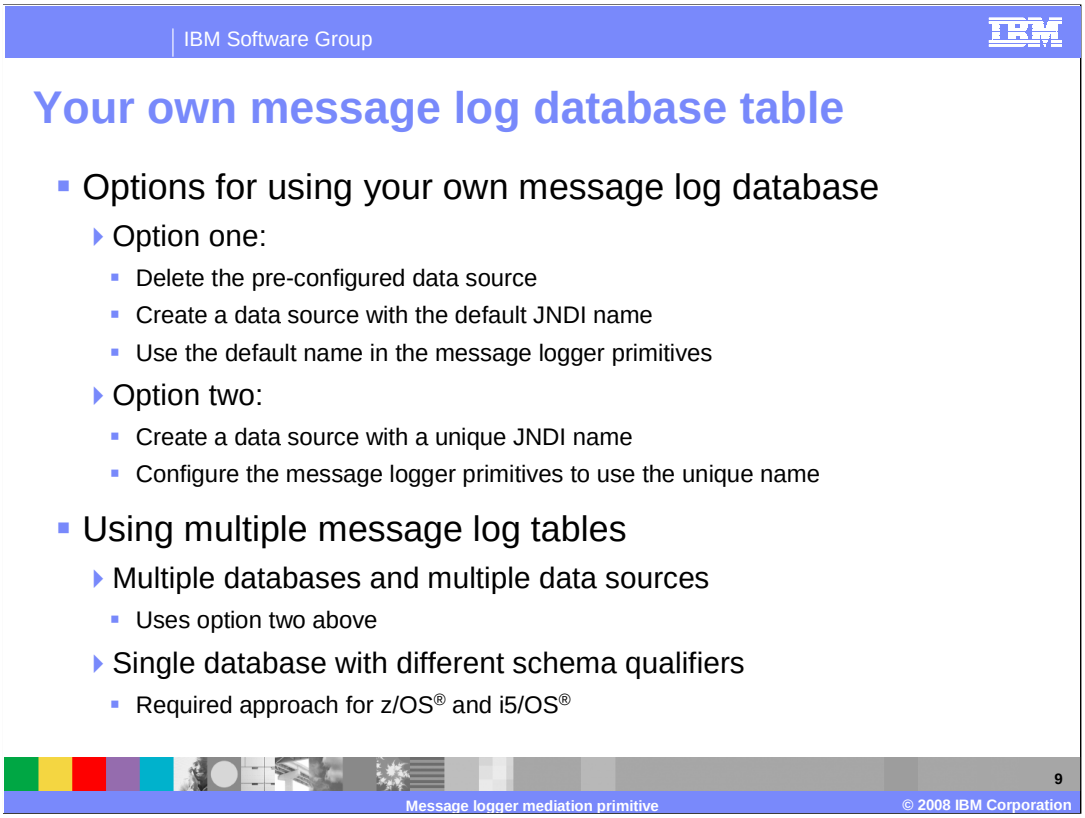
There are alternatives to the use of the ESBLOG.MSGLOG table in the common database. One approach is to use a different database that contains an ESBLOG.MSGLOG table. When configured this way, it is the use of the JNDI data source configured for the message logger that identifies the database to use. Another approach is to use the common database but write to a MSGLOG table that has a different schema qualifier, for example, MYLOG. When doing this, it is the use of the environment variable ESB_MESSAGE_LOGGER_QUALIFER that identifies the schema qualifier to use. Of course, the two approaches can be combined, using a different database in addition to different schema qualifiers.

Your own message log database table

- Supported databases:
  - Cloudscape®, DB2®, Derby, Informix®, Oracle, Sybase, Microsoft® SQL Server®
  - Data definition language (DDL) files are provided:
    - <install_root>/util/EsbLoggerMediation/<database_type>/Table.ddl
- Scripts for creating resources are provided:
  - <install_root>/util/createMessageLoggerResource.jacl
  - Can be used to create
    - Database table
    - Data source
    - Schema qualifier

You can configure the message logger primitive to use a database other than the common database. The data definition language (DDL) needed to create the message logger schema and table is provided. There are separate Table.ddl files for each of the supported databases, which include Cloudscape, DB2, Derby, Informix, Oracle, Sybase and Microsoft SQL Server. The DDL files are located in the server directory structure as indicated in the slide.

To help with the configuration of your environment, the script createMessageLoggerResource.jacl is provided. It can be used to create a database table, data source or schema qualifier.



When using your own database, it is best to have a strategy on how you plan to configure your data source and your message loggers. The first option is to delete the pre-configured data source and create a new data source 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 data source 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 because the JNDI name property does not need to be changed from its default value.

You might also want to have multiple tables used for logging. In this case, there are two approaches. The first is to follow option two, creating multiple databases and data sources, and then configuring each individual message logger to use an appropriate data source. The second approach is to use only one database containing multiple schema qualifiers, with a message log table contained with each schema qualifier. When this is done, the environment variable ESB_MESSAGE_LOGGER_QUALIFER must be set to indicate the appropriate schema qualifier to use. This approach is required when running on the z/OS and i5/OS platforms which provide the capability for only a single database.

Migration considerations

- Prior to V6.1
  - Message log table was not in the common database
  - Default database only usable for single server
- Migrating from releases before V6.1
  - The table schema has not changed, therefore:
    - Keep old and new messages in separate databases
    - Move old messages to new table in common database
    - Configure new environment to use the old database
- Cell with mixed nodes (V6.1 and prior versions)
  - Same options as above
  - Configure prior versions to point to the common database

The default database used for the message logger has changed in V6.1 from the implementation in prior releases. In V6.0.2 and earlier releases, the default message log database was a unique database and not part of the common database. Also, it was contained in a Cloudscape database and therefore was only usable in a single server environment. Because of this change, some consideration for migration from prior releases must be addressed. Because the actual table schema has not changed, the migration options are rather straight forward. One way to continue is to maintain the old and new message in separate databases. Another approach is to move the existing old messages into the new database. A third approach is that the new environment be configured to continue to use the old database. Another consideration is when you have a cell environment with nodes from varying versions. In this case, the same options can be considered, with the addition of an option for configuring the nodes with prior versions to use the common database.

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 is 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 are no tools 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 in the slide 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. Additional fields include the name of the mediation module, the message content in XML format as defined by the Root property, and finally the SMO version associated with this log message. You need to understand this schema in order to develop an application to retrieve and analyze the log message data.

Error processing

- MediationRuntimeException thrown for:
  - Incorrect JNDI name for data source
  - 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.

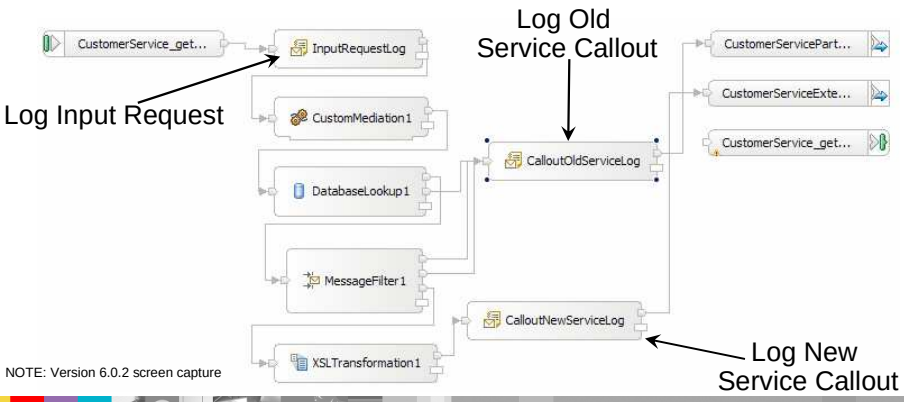
A MediationRuntimeException is thrown for an incorrect JNDI name for the data source. It is also thrown for the case where the root property has been specified as a null XPath value.

A MediationConfigurationException occurs for any kind of problems accessing the message log database. If the Fail terminal is wired, that flow is followed rather than the exception being thrown. When there is a failure in the mediation flow downstream from the message logger, the message normally remains logged in the database. However, if the transaction mode has been set to same and the flow has been configured to run in a global transaction, the message is removed from the log by 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 is still written but has an empty message field.

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 versus new service
  - Log the input request and log the callouts to the old and new services

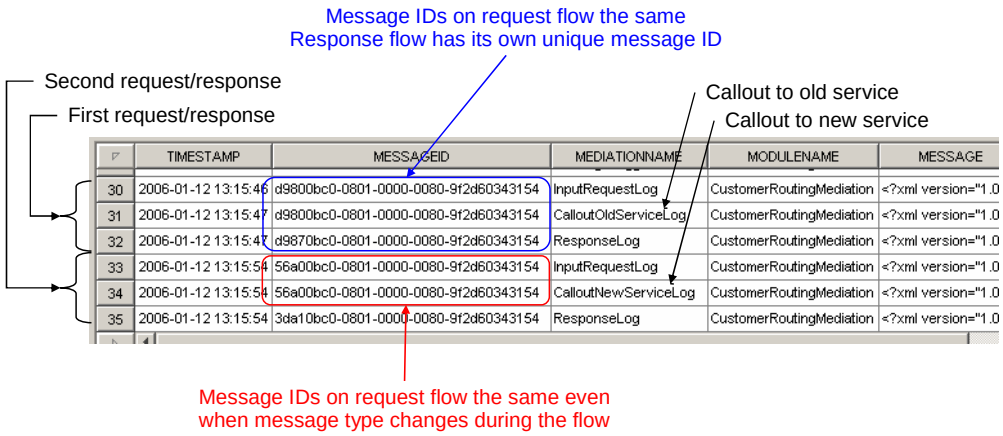


This slide illustrates a possible use of the message logger primitive. The screen capture is taken from the mediation flow editor using WebSphere Integration Developer version 6.0.2. You might notice some differences in the visual appearance from version 6.1, but the flow being described is the same between these versions.

In this example, 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 can see that there is a message logger at the beginning of the flow that records every request. There is also a message logger before the callout node to each of the service providers, so for any given request flow, there are two messages logged. Not shown in this screen capture is the message logger in the response flow 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.

Example usage

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



	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

This screen capture shows the contents of a message log produced from the example on the previous slide. 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 you can take note of in this screen capture. First, note that the columns for the database include the timestamp, message id, mediation name, module name and the message. There is also an SMO version column which doesn't show in this screen capture.

There are three logs for each request, which represent the incoming message, the callout to the old or new service provider and the response from the provider. In the mediation name column you can see 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. 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 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. Notice that 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 a primitive.

Summary

- Examined the message logger mediation primitive

Message logger

- Overview of function
- Use of terminals
- Definition of properties
- Message log database
- Migration from previous releases
- Retrieval and usage of log data
- Error handling
- Example usage

In summary, this presentation provided details regarding the message logger mediation primitive. It presented an overview of the message logger, along with information about the primitive's use of terminals and its properties. The message logger makes use of a message log database table which was described, along with some configuration options available to you. Since there have been some changes from prior releases, migration considerations were addressed. A discussion of how you can retrieve and make use of the log data was presented, followed by some error handling information, and finally an example use of a message logger primitive was presented.

Feedback

Your feedback is valuable

You can help improve the quality of IBM Education Assistant content to better meet your needs by providing feedback.

- Did you find this module useful?
- Did it help you solve a problem or answer a question?
- Do you have suggestions for improvements?

Click to send e-mail feedback:

mailto:iea@us.ibm.com?subject=Feedback_about_WBIV61_MessageLoggerPrimitive.ppt

This module is also available in PDF format at: ../WBIV61_MessageLoggerPrimitive.pdf

You can help improve the quality of IBM Education Assistant content by providing feedback.

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:

Cloudscape DB2 i5/OS Informix WebSphere z/OS

A current list of other IBM trademarks is available on the Web at http://www.ibm.com/legal/copytrade.shtml

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

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 or changes in the products or programs 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 (for example, 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.

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.