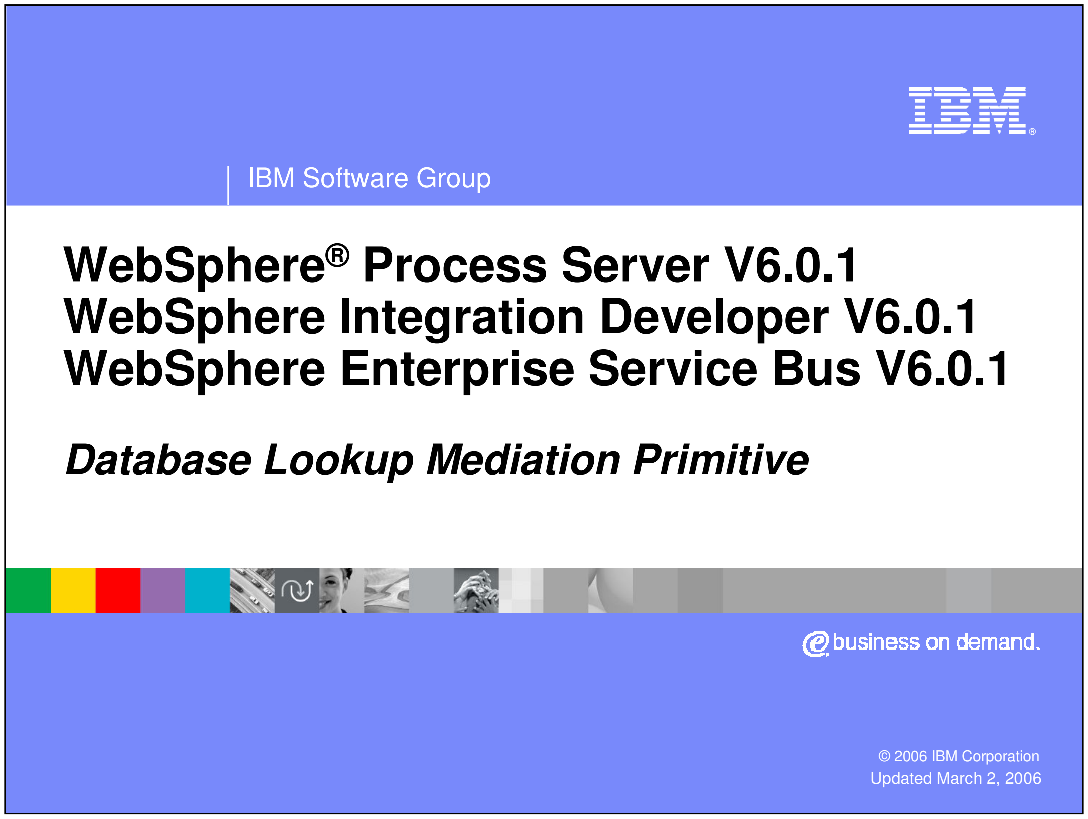


IBM Software Group

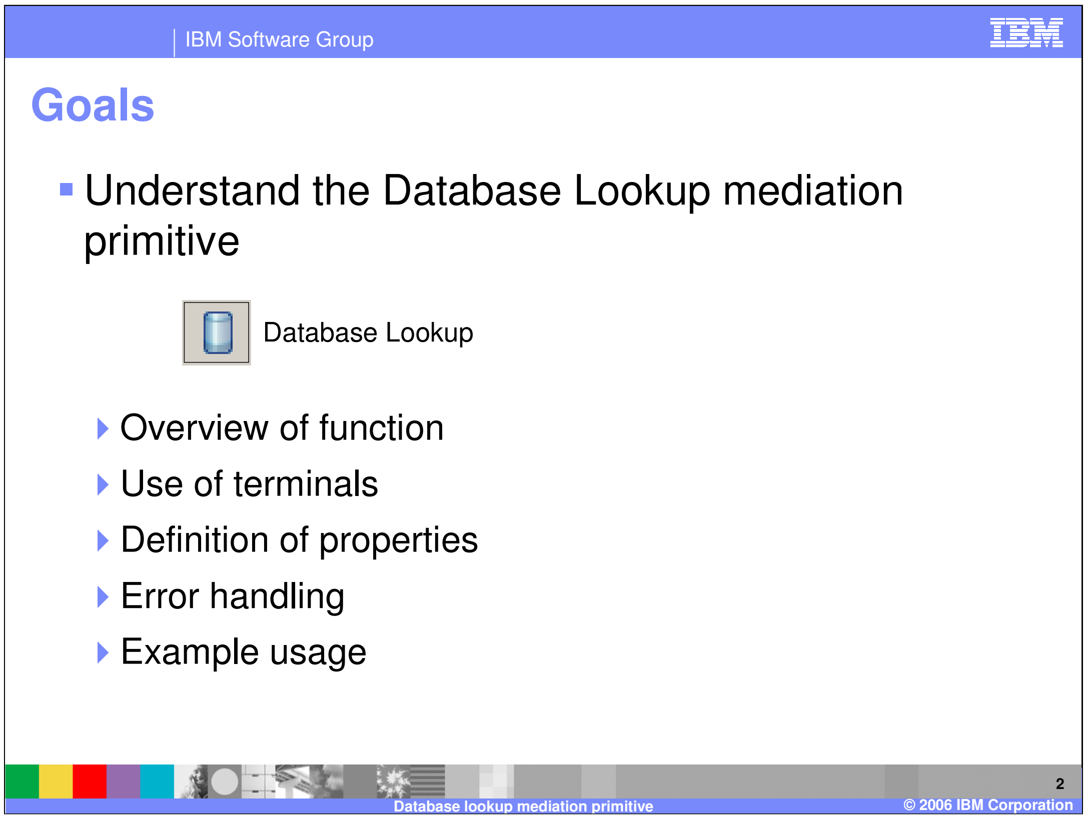
WebSphere® Process Server V6.0.1 WebSphere Integration Developer V6.0.1 WebSphere Enterprise Service Bus V6.0.1

Database Lookup Mediation Primitive

business on demand.

Updated March 2, 2006

This presentation will provide a detailed look at the Database Lookup mediation primitive.



Goals

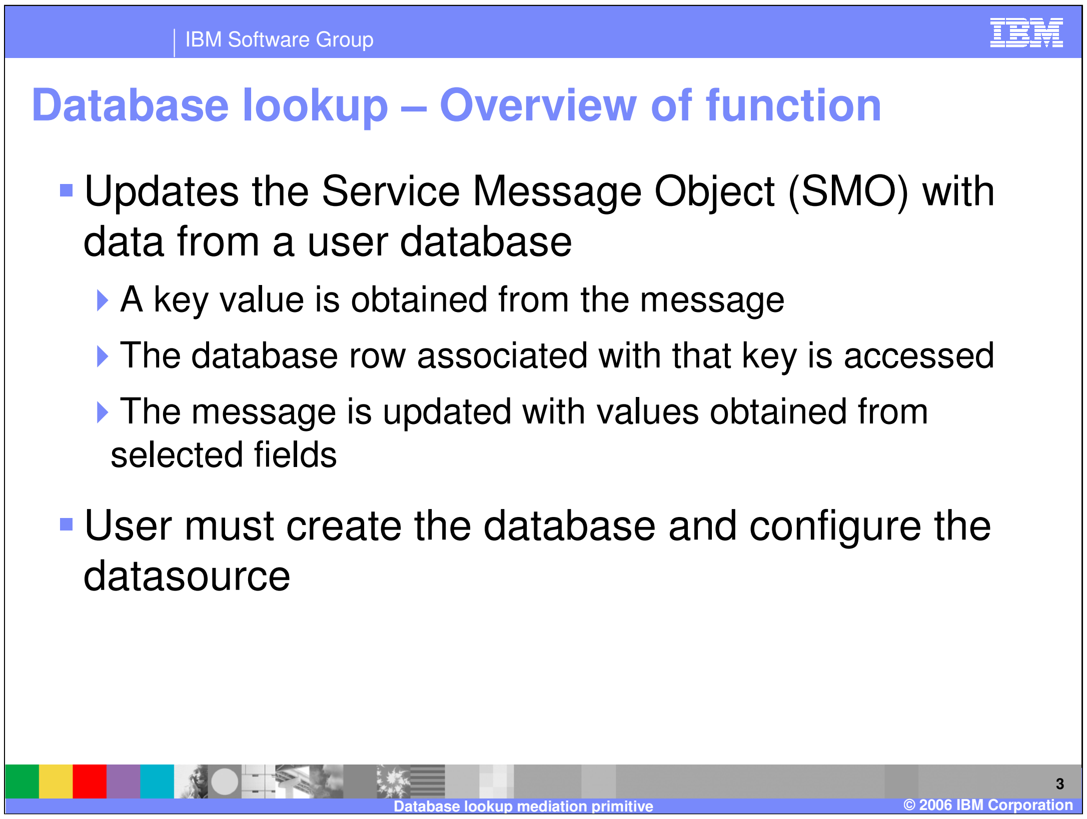
- Understand the Database Lookup mediation primitive

  Database Lookup

  - Overview of function
  - Use of terminals
  - Definition of properties
  - Error handling
  - Example usage

The goal of this presentation is to provide you with a full understanding of the Database Lookup mediation primitive. The presentation assumes that you are already familiar with the material presented in the **Mediation Primitive Common Details** presentation because that serves as a base for understanding mediation primitives in general.

In this presentation an overview of the Database Lookup is presented along with information about the primitive's use of terminals, its properties and error handling characteristics. Finally an example usage of a Message Logger is presented.

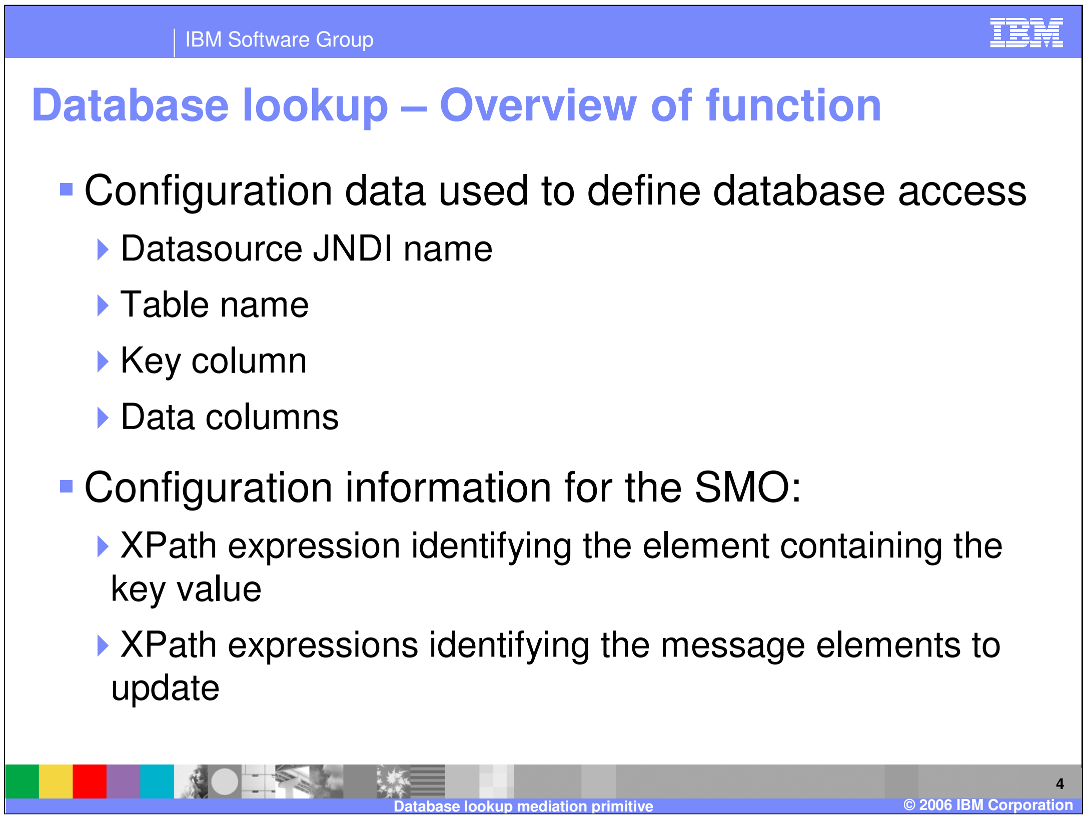


Database lookup – Overview of function

- Updates the Service Message Object (SMO) with data from a user database
  - A key value is obtained from the message
  - The database row associated with that key is accessed
  - The message is updated with values obtained from selected fields
- User must create the database and configure the datasource

The purpose of the Database Lookup is to update the Service Message Object with data obtained from a user database.

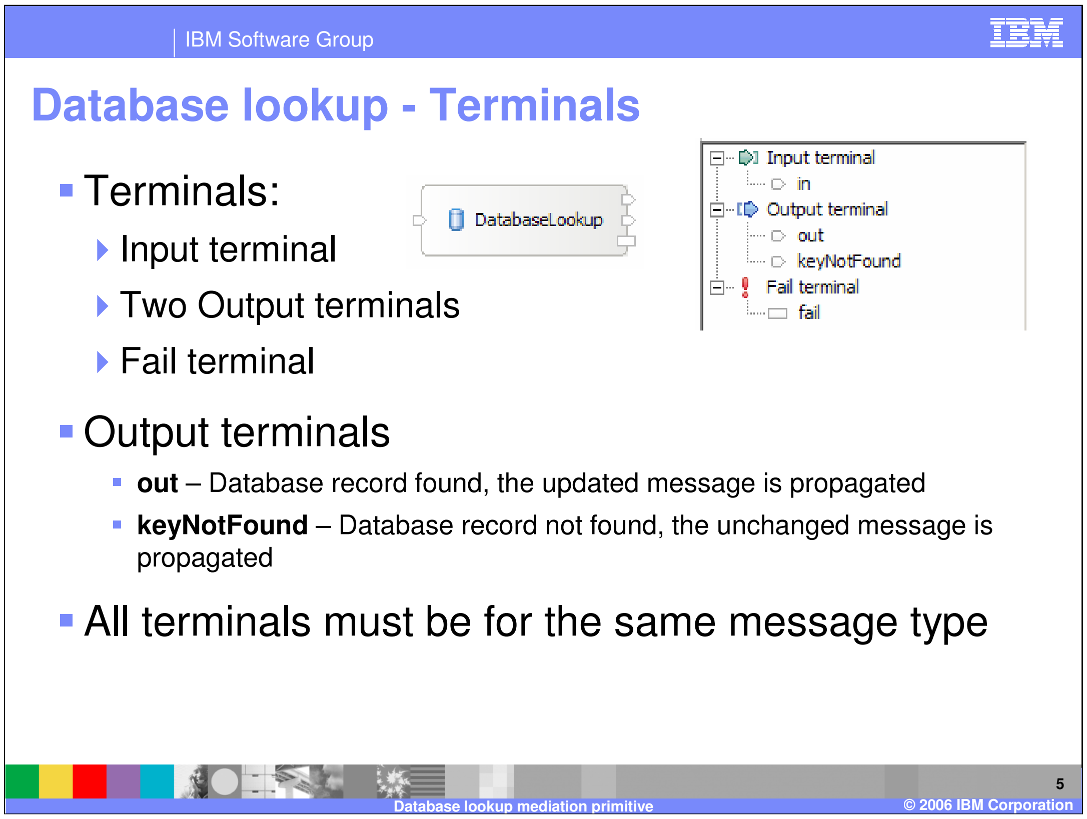
The Database Lookup primitive first obtains a key value from the SMO and uses that key to access a row from a database table. Values from selected fields in that row are then used to update the SMO. Depending upon your situation you must either create or use an existing database and you must also create a datasource that can be used to identify that database.



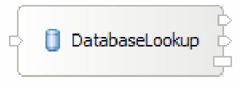
Database lookup – Overview of function

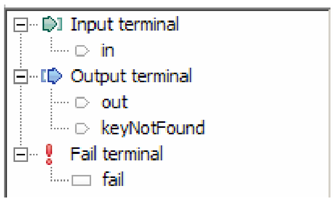
- Configuration data used to define database access
  - Datasource JNDI name
  - Table name
  - Key column
  - Data columns
- Configuration information for the SMO:
  - XPath expression identifying the element containing the key value
  - XPath expressions identifying the message elements to update

There is quite a bit of configuration data associated with a Database Lookup primitive. The configuration data associated with the database includes the JNDI name for the datasource, the name of the table on which to do the lookup, the column used for the key and the columns from which to extract data. The configuration data associated with the SMO includes the XPath expression that identifies the element in the SMO containing the key value and XPath expressions which identify the elements in the SMO to be updated with values from the database.


IBM Software Group

Database lookup - Terminals




Input terminal
in
Output terminal
out
keyNotFound
Fail terminal
fail

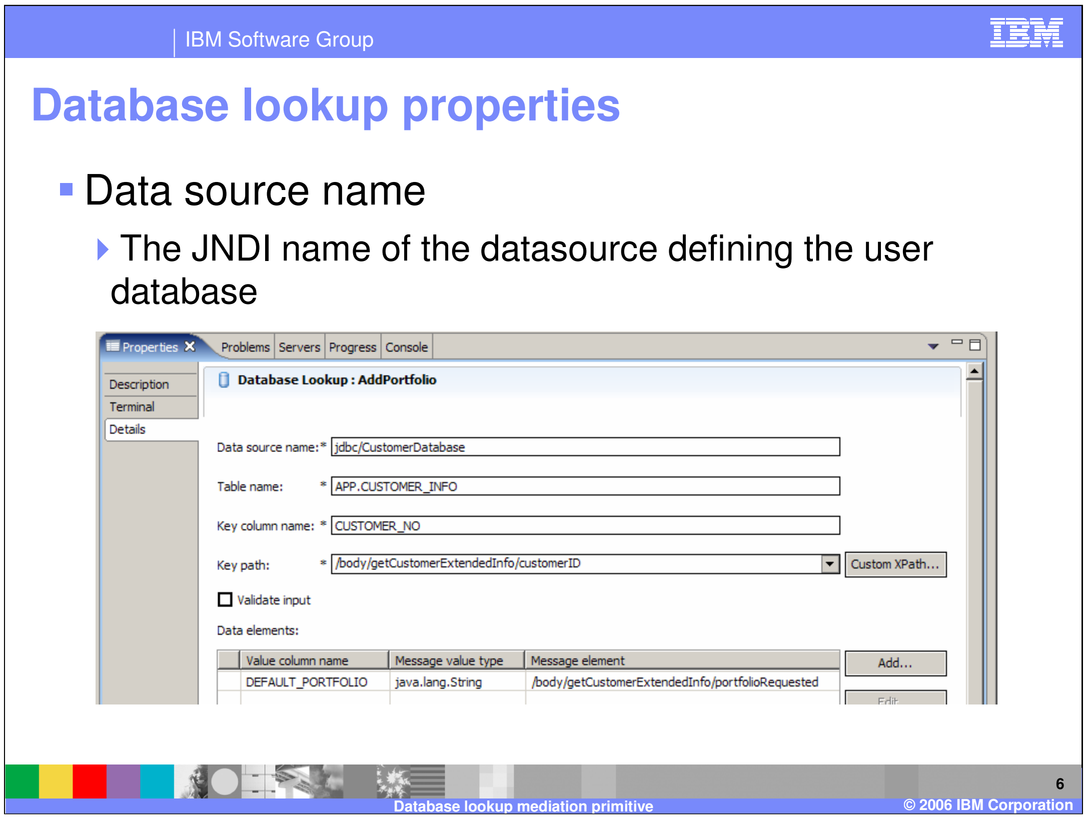
- Terminals:
  - Input terminal
  - Two Output terminals
  - Fail terminal
- Output terminals
  - **out** – Database record found, the updated message is propagated
  - **keyNotFound** – Database record not found, the unchanged message is propagated
- All terminals must be for the same message type

5

Database lookup mediation primitive

© 2006 IBM Corporation

The Database Lookup primitive has one input terminal, two output terminals and a fail terminal. There is one output terminal named **out** used when the database lookup is successful and another output terminal named **keyNotFound** used when there is no row in the database for the key value obtained from the SMO. The output terminals must be for the same message type as the input terminal as the database lookup primitive does not modify the message body structure although it may update the message body contents. The slide shows a database lookup primitive with its terminals and also the terminals as seen in the properties view.



Database lookup properties

- Data source name
  - The JNDI name of the datasource defining the user database

Properties | Problems | Servers | Progress | Console

Description
Terminal
Details

Database Lookup : AddPortfolio

Data source name: * jdbc/CustomerDatabase

Table name: * APP.CUSTOMER_INFO

Key column name: * CUSTOMER_NO

Key path: * /body/getCustomerExtendedInfo/customerID

Custom XPath...

☐ Validate input

Data elements:

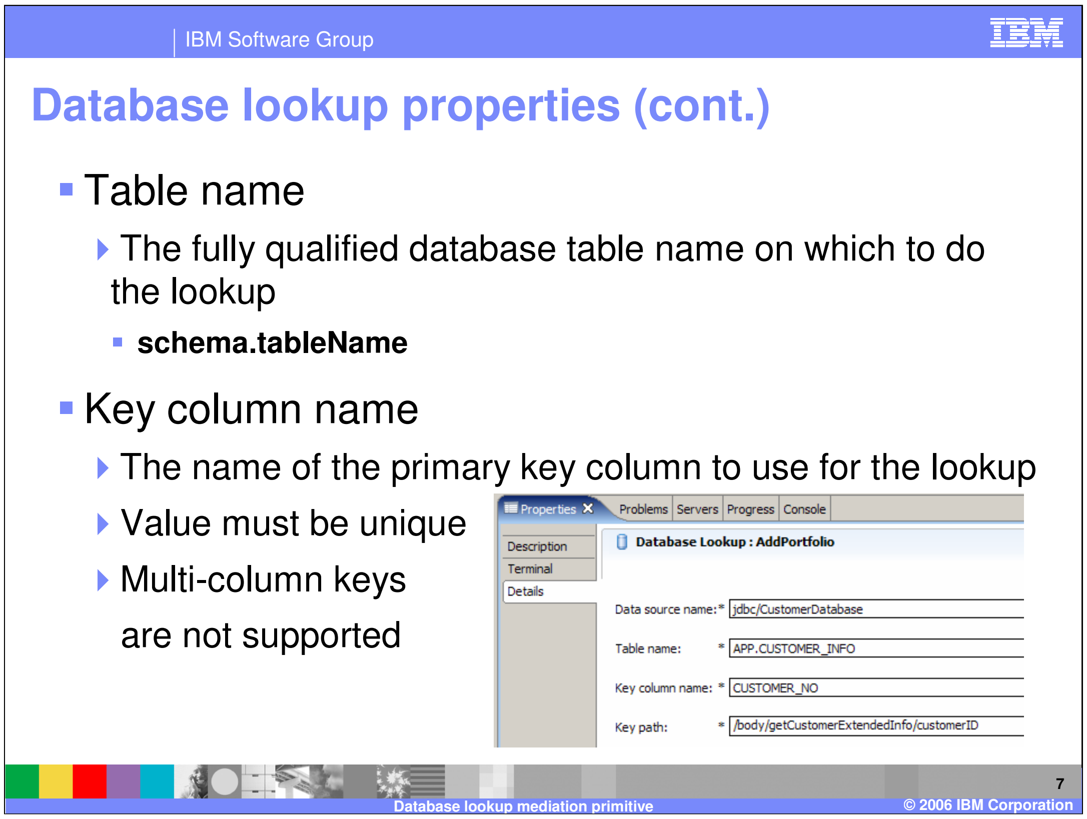
Value column name	Message value type	Message element
DEFAULT_PORTFOLIO	java.lang.String	/body/getCustomerExtendedInfo/portfolioRequested

Add...

Edit

This screen capture shows the Details tab from the Properties view for a Database Lookup.

The **Data source name** property is a JNDI name used to lookup the datasource which identifies the database containing the table to be used for the lookup. When creating a new Database Lookup this property does not have a default and must be set.


IBM Software Group

Database lookup properties (cont.)

- Table name
  - The fully qualified database table name on which to do the lookup
    - **schema.tableName**
- Key column name
  - The name of the primary key column to use for the lookup
  - Value must be unique
  - Multi-column keys are not supported

Properties | Problems | Servers | Progress | Console

Description | Terminal | Details

Database Lookup : AddPortfolio

Data source name:*	jdbc/CustomerDatabase
Table name: *	APP.CUSTOMER_INFO
Key column name: *	CUSTOMER_NO
Key path: *	/body/getCustomerExtendedInfo/customerID

7

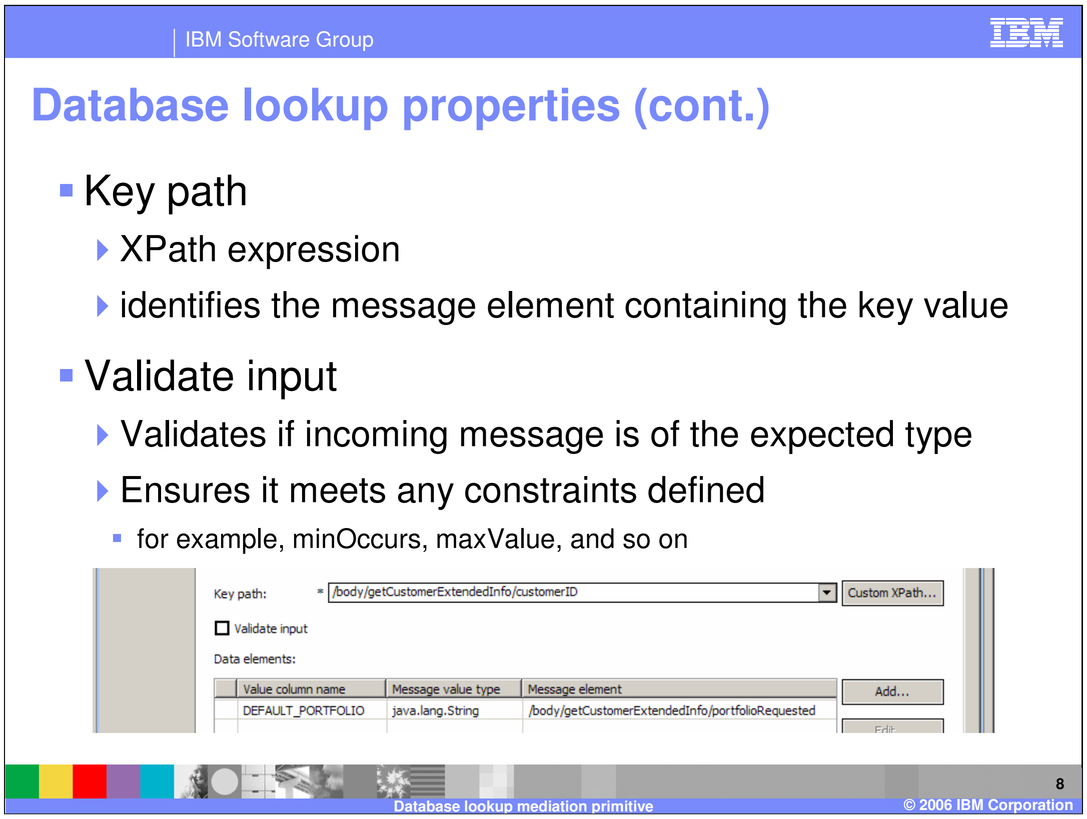
Database lookup mediation primitive

© 2006 IBM Corporation

The **Table name** property contains the fully qualified database table name on which to do the lookup. In some cases simply using the unqualified table name will work but it is recommended that you use the fully qualified schema.tableName value for this property as that will always yield the expected results.

The **Key column name** property identifies the name of the primary key column to be used for the lookup. The key value must be unique within the database table and the use of multi-column keys is not supported.

The following slide will continue to examine the properties of the Database Lookup primitive.



IBM Software Group

Database lookup properties (cont.)

- Key path
  - XPath expression
  - identifies the message element containing the key value
- Validate input
  - Validates if incoming message is of the expected type
  - Ensures it meets any constraints defined
    - for example, minOccurs, maxValue, and so on

Key path: * /body/getCustomerExtendedInfo/customerID [Custom XPath...]

☐ Validate input

Data elements:

Value column name	Message value type	Message element
DEFAULT_PORTFOLIO	java.lang.String	/body/getCustomerExtendedInfo/portfolioRequested

8

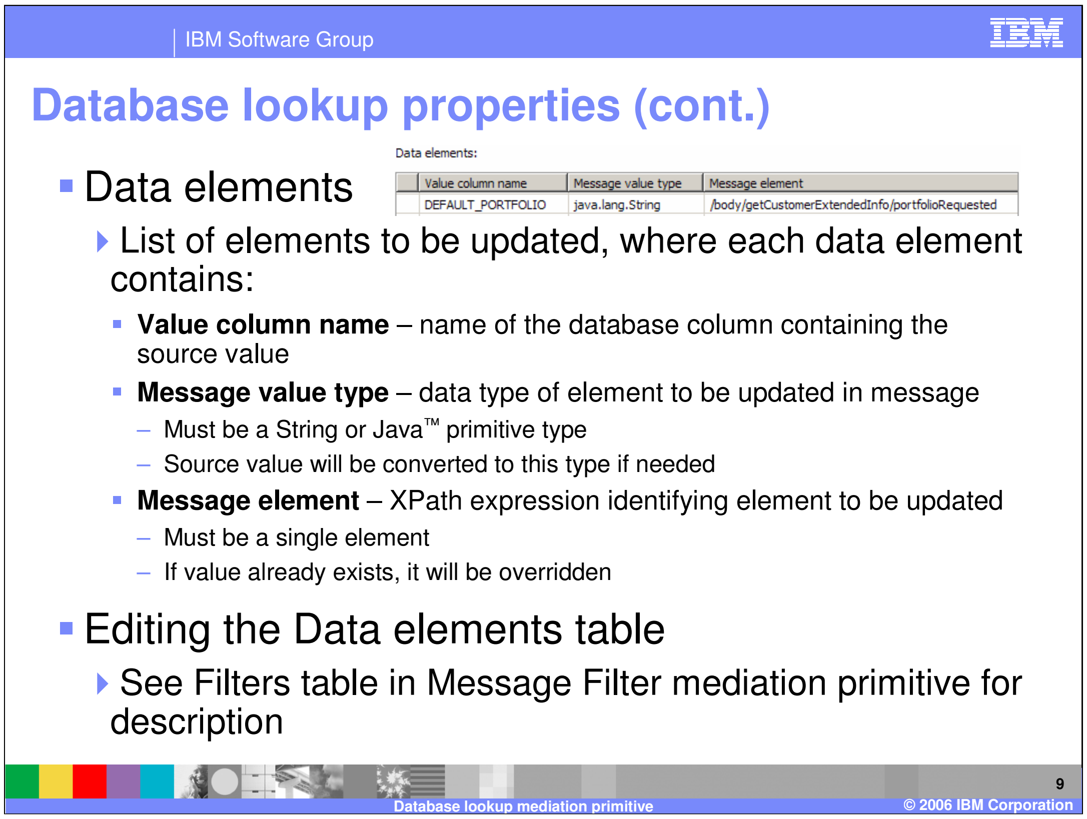
Database lookup mediation primitive

© 2006 IBM Corporation

The **Key path** property contains an XPath expression which identifies the element within the SMO whose value is to be used for the key in the lookup. This expression must identify a single element. The **Custom XPath…** button can be used to access the XPath Expression Builder dialog to construct the XPath expression to be used.

The **Validate input** property is a checkbox use to indicate if incoming messages to the Database Lookup primitive are to be validated prior to processing. This will ensure that the incoming message is of the expected type and that any constraints defined are not violated.

The following slide will continue to examine the properties of the Database Lookup primitive.



IBM Software Group

Database lookup properties (cont.)

Data elements:

	Value column name	Message value type	Message element
	DEFAULT_PORTFOLIO	java.lang.String	/body/getCustomerExtendedInfo/portfolioRequested

- Data elements
  - List of elements to be updated, where each data element contains:
    - **Value column name** – name of the database column containing the source value
    - **Message value type** – data type of element to be updated in message
      - Must be a String or Java™ primitive type
      - Source value will be converted to this type if needed
    - **Message element** – XPath expression identifying element to be updated
      - Must be a single element
      - If value already exists, it will be overridden
- Editing the Data elements table
  - See Filters table in Message Filter mediation primitive for description

9

Database lookup mediation primitive © 2006 IBM Corporation

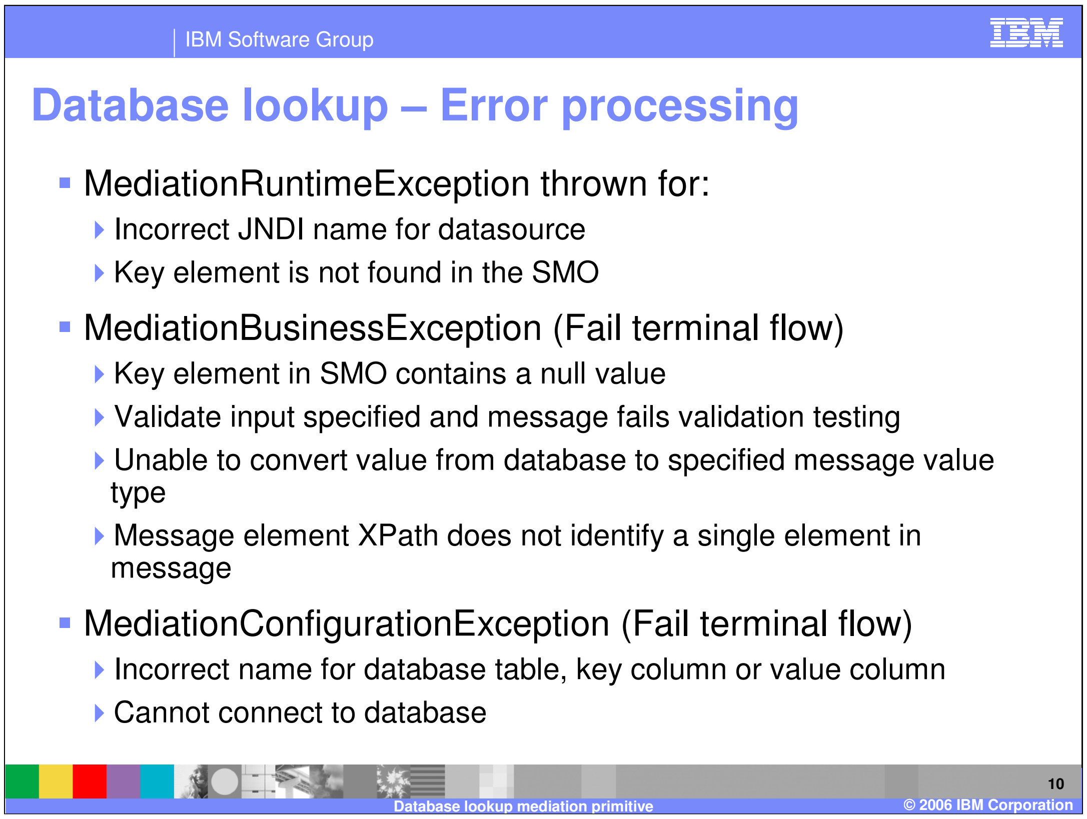
The final property for the Database Lookup primitive is the **Data elements** property which is a table consisting of three columns. Each row in the table identifies a specific element within the SMO which is to be updated with data obtained from the database.

The first column is the **Value column name** which identifies the column in the database from which the source value for the update is obtained.

The next column is the **Message value type** which defines the data type within the SMO for the element to be updated. This must be either a String or a primitive Java type. The source value obtained from the database column will be converted to this type prior to being inserted into the SMO.

The third column is the **Message element** which identifies the element in the SMO to be updated. This is specified with an XPath expression and must identify a single element within the SMO. If a value for that element already exists in the SMO it will be overwritten.

The Data elements table is edited and manipulated much the same as the Filters table use with the Message Filter primitive. The presentation on the Message Filter primitive provides details about editing the table.


IBM Software Group

Database lookup – Error processing

- MediationRuntimeException thrown for:
  - Incorrect JNDI name for datasource
  - Key element is not found in the SMO
- MediationBusinessException (Fail terminal flow)
  - Key element in SMO contains a null value
  - Validate input specified and message fails validation testing
  - Unable to convert value from database to specified message value type
  - Message element XPath does not identify a single element in message
- MediationConfigurationException (Fail terminal flow)
  - Incorrect name for database table, key column or value column
  - Cannot connect to database

10

Database lookup mediation primitive

© 2006 IBM Corporation

The error processing details and considerations are examined in this slide.

A MediationRuntimeException will be thrown for an incorrect JNDI name for the datasource. It is also thrown if the key element does not exist in the SMO which indicates a problem with the configuration XPath expression for the key element.

A MediationBusinessException occurs for several different problems including the following:

•The element for the key value in the SMO contains a null value. This is different than the situation causing the MediationRuntimeException in that the element does exist in this case, the problem is that it has a null value and therefore cannot be used as a key in the lookup.

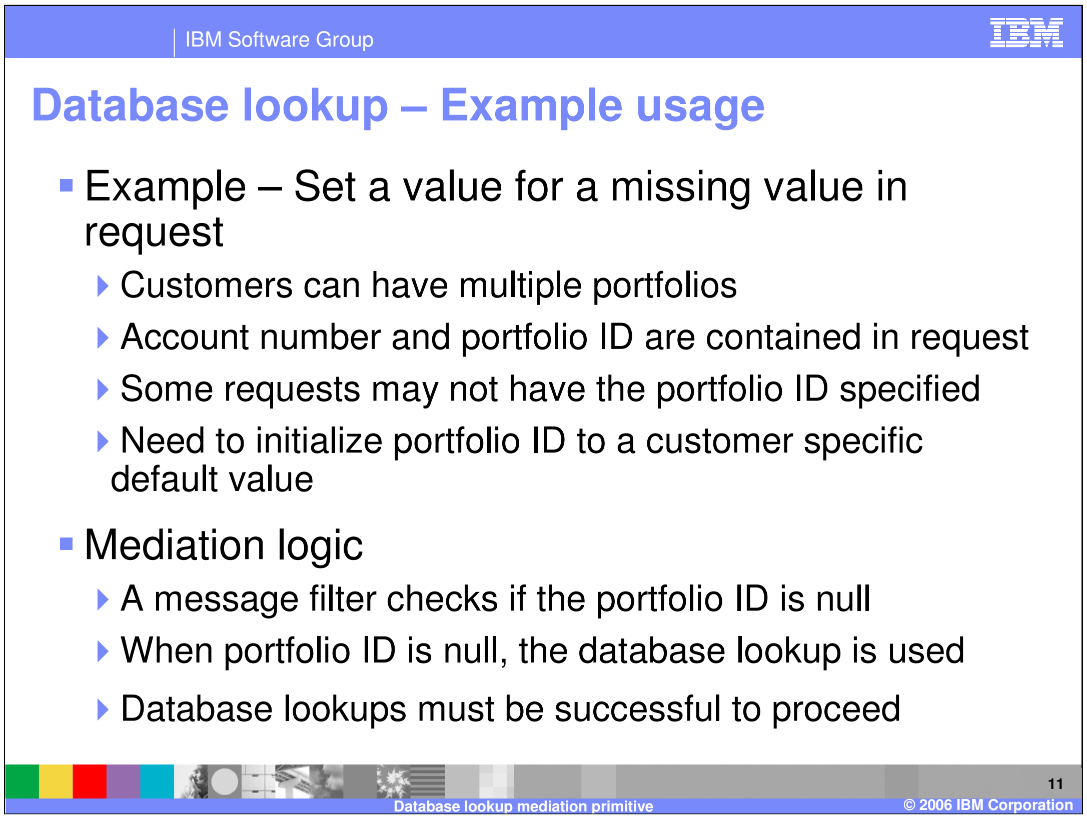
•Another cause is when Validate input has been specified and the message fails the validation processing.

•The MediationBusinessException will also occur when the value taken from the database cannot be converted to the specified message value type. For example, if the message type was “int” and the value obtained from the database was “ABC”.

•Another reason would be if the XPath expression for the Message element identified something in the SMO which was not a single element.

In all of these cases, if the Fail terminal is wired the flow from the Fail terminal will be followed rather than the exception being thrown.

A MediationConfigurationException occurs for any kind of issues with the configuration of the database such as an incorrect table name, key column name or value column name. It will also occur if there are any other problems accessing the database. In these cases, if the Fail terminal is wired the flow from the Fail terminal will be followed rather than the



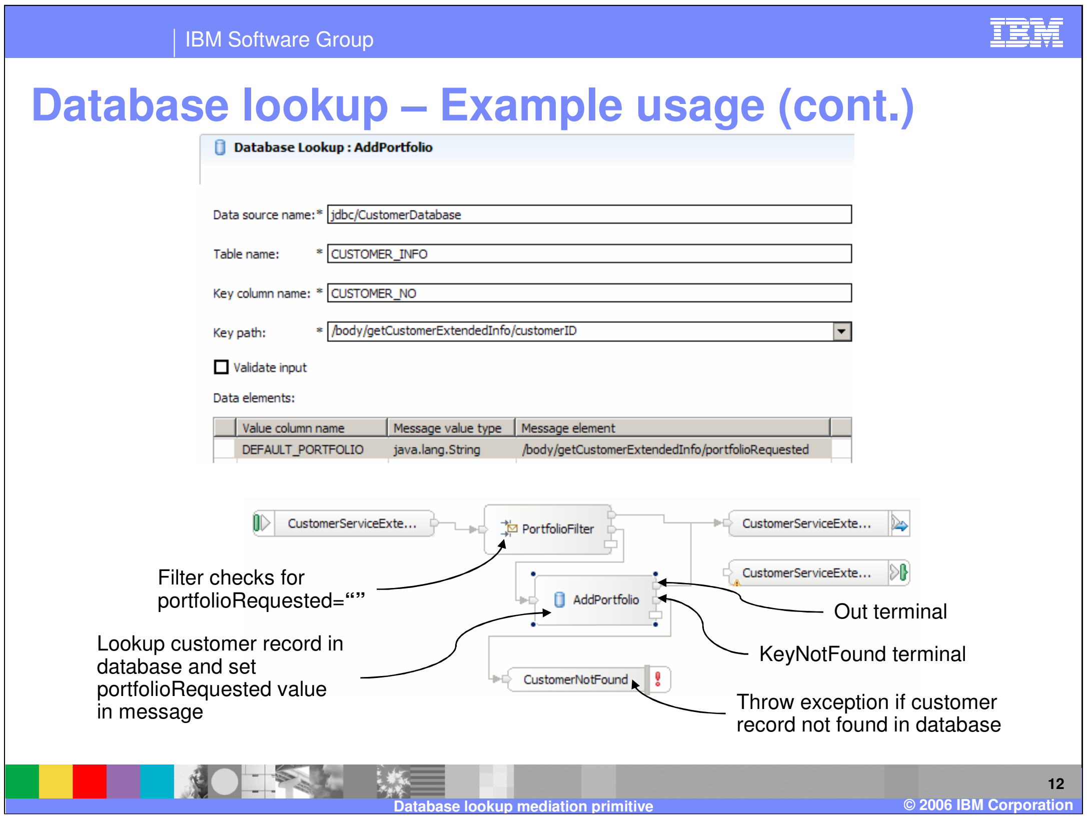
Database lookup – Example usage

- Example – Set a value for a missing value in request
  - Customers can have multiple portfolios
  - Account number and portfolio ID are contained in request
  - Some requests may not have the portfolio ID specified
  - Need to initialize portfolio ID to a customer specific default value
- Mediation logic
  - A message filter checks if the portfolio ID is null
  - When portfolio ID is null, the database lookup is used
  - Database lookups must be successful to proceed

In the next couple of slides you will be shown an example usage of the Database Lookup primitive. The example looks at a case where some service requests do not contain all of the data required by the service provider and the database lookup is used to obtain the data from a database and insert it into the message. In the scenario customers can have multiple portfolios and requests to the service provider should contain the customer account number and the portfolio ID. However, some service requests only contain the customer account number. The database lookup is used to determine the default portfolio ID for that customer's account and places it into the message.

The logic in the mediation flow is as follows. A message filter is used to check if the portfolio ID in the message is null. When the portfolio ID is null a database lookup is used to obtain the default portfolio ID. In the event that the database lookup fails it is considered an error.

The next slide shows the database lookup and mediation flow used in this scenario.



IBM Software Group

Database lookup – Example usage (cont.)

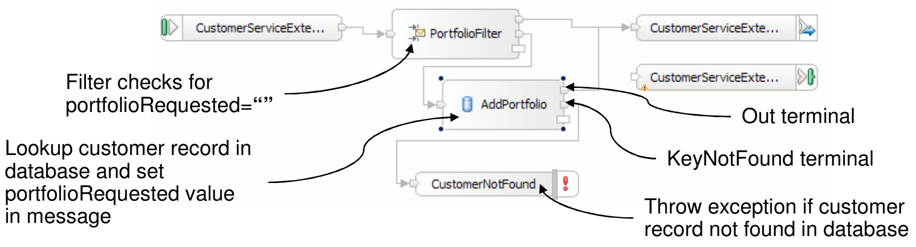
Database Lookup : AddPortfolio

Data source name:*	jdbc/CustomerDatabase
Table name: *	CUSTOMER_INFO
Key column name: *	CUSTOMER_NO
Key path: *	/body/getCustomerExtendedInfo/customerID

☐ Validate input

Data elements:

Value column name	Message value type	Message element
DEFAULT_PORTFOLIO	java.lang.String	/body/getCustomerExtendedInfo/portfolioRequested


CustomerServiceExte...
PortfolioFilter
CustomerServiceExte...
CustomerServiceExte...
AddPortfolio
CustomerNotFound
Filter checks for portfolioRequested=“”
Lookup customer record in database and set portfolioRequested value in message
Out terminal
KeyNotFound terminal
Throw exception if customer record not found in database

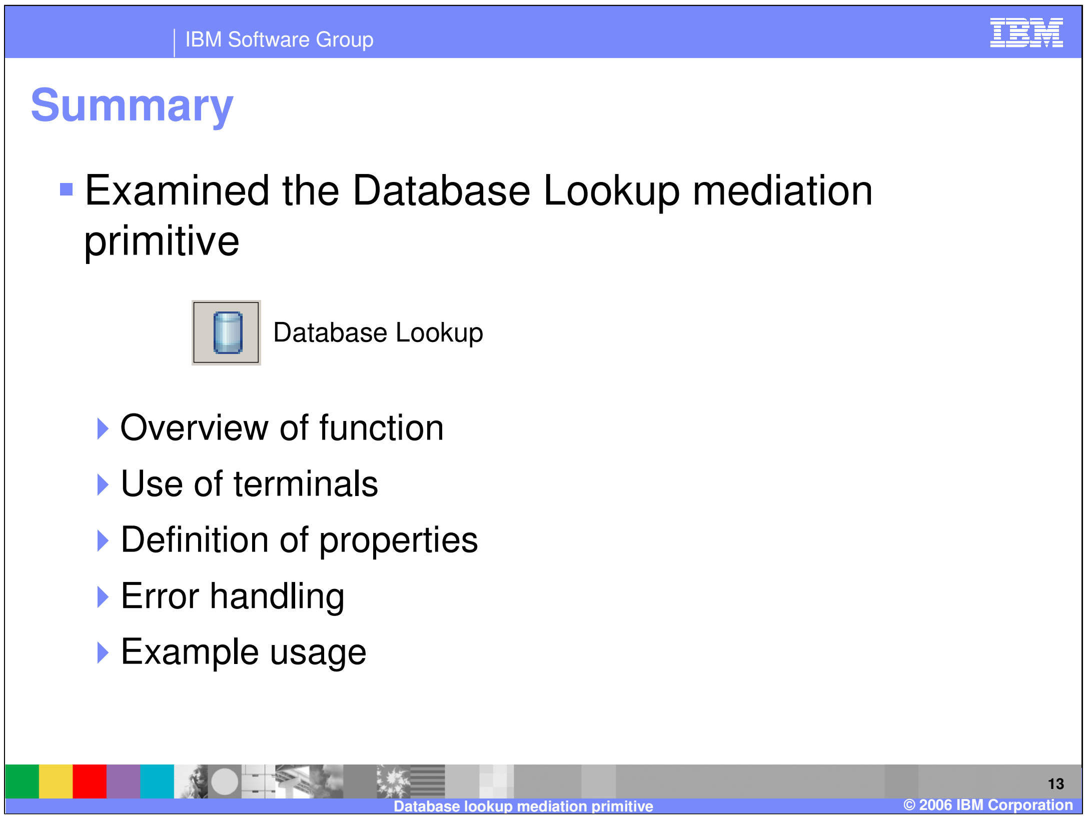
12

Database lookup mediation primitive

© 2006 IBM Corporation

The top part of the slide shows the properties for the Database Lookup. The lookup is done on the CUSTOMER_INFO table with a key column of CUSTOMER_NO. The key value is obtained from the SMO from the message body in the customerID element. In the Data elements table you will see that the value in the DEFAULT_PORTFOLIO column is obtained and used to update the portfolioRequested element in the SMO body.

The bottom of the slide shows the mediation flow. The message is first passed to the PortfolioFilter message filter primitive which determines if the portfolio ID is contained in the request. If it is, the flow goes directly to the callout for the service provider. If not, the AddPortfolio database lookup primitive is used to access the customer information in the database and update the SMO with the portfolio ID. If the lookup is successful the flow goes to the callout for the service provider. If the lookup is not successful, the keyNotFound terminal is wired to a Fail primitive which will cause an exception to be raised.



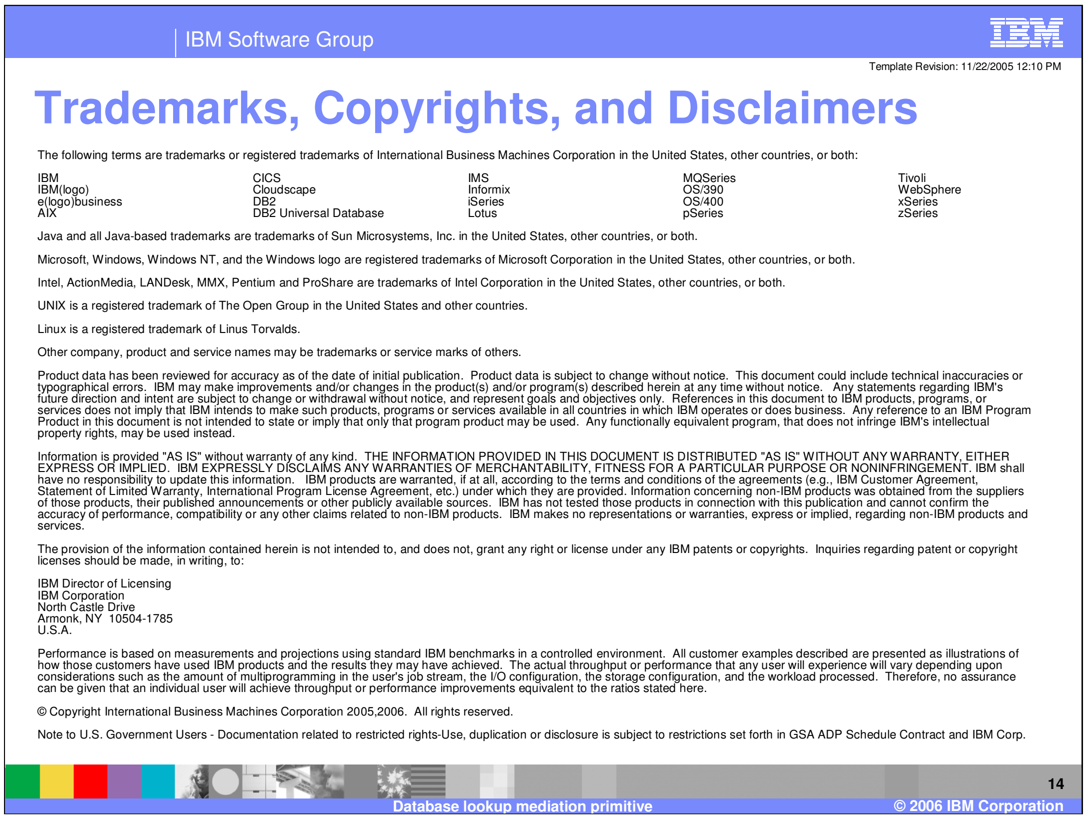
Summary

- Examined the Database Lookup mediation primitive

  Database Lookup

  - Overview of function
  - Use of terminals
  - Definition of properties
  - Error handling
  - Example usage

This presentation covered details about the Database Lookup mediation primitive. The presentation covered an overview of the Database Lookup along with information about the primitive's use of terminals, its properties and error handling characteristics. Finally an example usage of a Database Lookup was presented.



Template Revision: 11/22/2005 12:10 PM

Trademarks, Copyrights, and Disclaimers