Graphical Data Mapping in IBM Integration Bus v9

Marisa Lopez de Silanes
IBM Integration Bus Development
IBM Hursley Park, UK
lopezdsr@uk.ibm.com

Agenda

- Graphical Data Mapping overview
- Designing a message map
- Graphical Data Mapping editor
- Editing message maps
- Transforming a SOAP message
- Executing a message map
- Troubleshooting message maps

Graphical Data Mapping

- ***Graphical data maps*** offer the ability to achieve the transformation of a message without the need to write code, providing a visual image of the transformation, and simplifying its implementation and ongoing maintenance.

- A ***message map*** is the IBM Integration Bus implementation of a graphical data map. It is based on XML schema and XPath 2.0 standards.

- You can use a message map to perform any of the following actions:
  - Transform a message
  - Enrich a message with data available in an external database
  - Modify data located in an external database
    - Note: You can call DB2 stored procedures from a graphical data map in IBM Integration Bus version 9
  - Route a message based on content.

Designing a message map

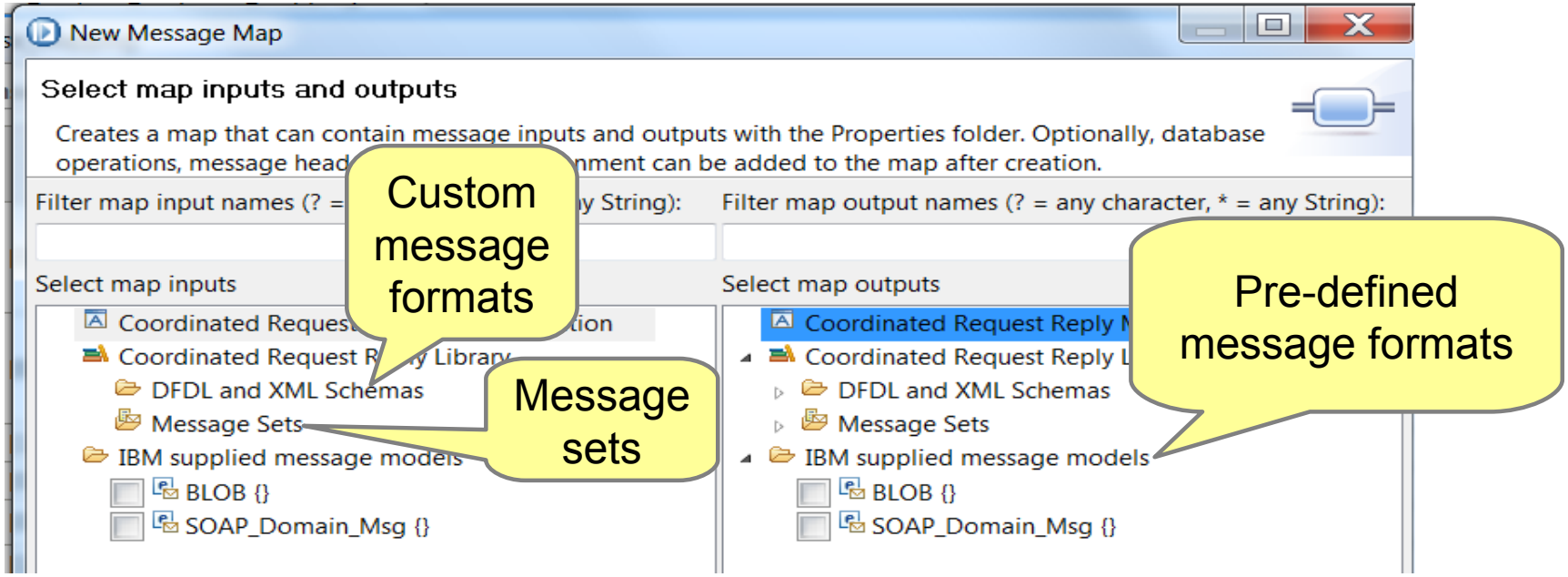
- The data structure that you define in a message map for an input or an output message is the IBM Integration Bus internal representation of the message.
- Each data transformation is driven by the type of the output element and the mapping operation required to calculate its value.
- A conditional expression can be defined per transform to define the condition under which a transform should be applied.
- Maps can be broken down into a hierarchy of ‘nested’ maps using one of the structural transforms to avoid clutter in large maps.
- Submaps allow you to reuse common transformations multiple times.
- If the only transform that you define between an input map component and an output map component is the ***Move*** transform so that the component is copied over without any modification, you are recommended to remove the component from the message map. The map transformation will be more efficient since the mapping engine will only focus on the structures that do require change.
- A null value mapped from an input element to an output element produces an output element with an empty value.
- Simple or complex type output elements can be created as nil elements.

Choosing a graphical data map

	Recommended use	Type of resource
Message map	Graphical data mapping	.map file
Submap	Reuse of common data transformations	.map file
Local map	Reduce complexity reading and managing a Message map	No file. It is embedded within a Message map
Legacy message map (*)	Solutions migrated from earlier versions of IBM Integration Bus	.map file

- (*) Note: You can use a legacy message map, but you cannot modify it in IBM Integration Bus. These type of maps are maintained for compatibility with earlier versions of IBM Integration Bus.

Input and output messages to a message map



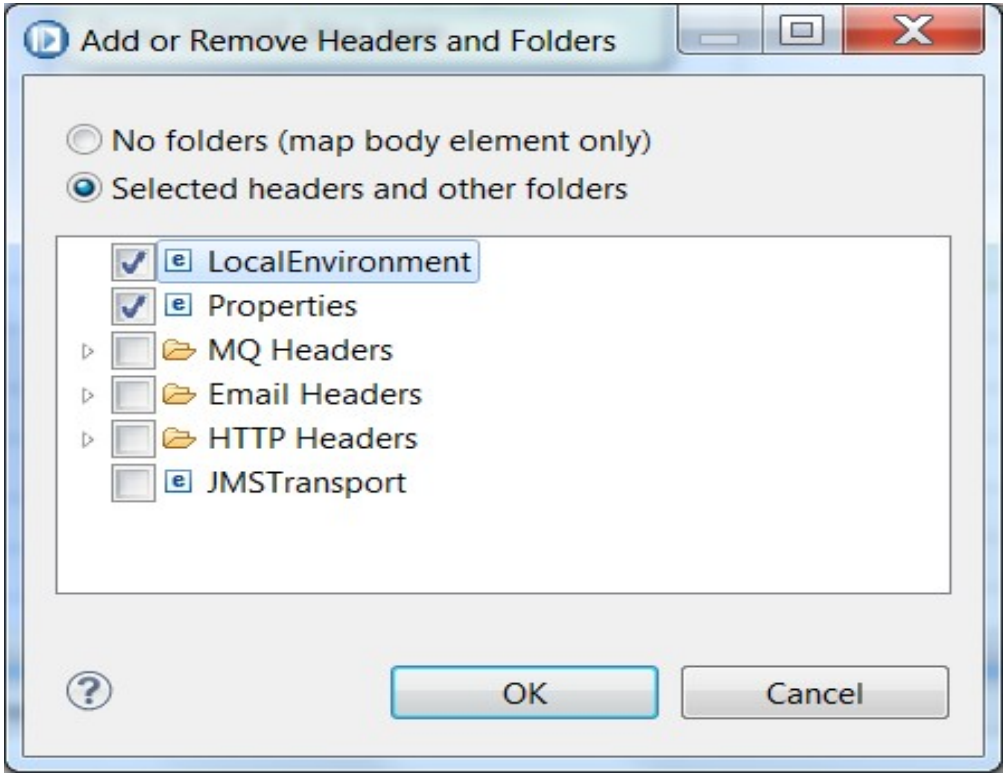
The following message domains are supported in a message map:

- DFDL
- XMLNSC
- DataObject
- SOAP
- BLOB
- MRM (The MRM domain is supported for compatibility with legacy message maps.)

Additional inputs and outputs: Message assembly components

The data structure that you define in a message map for an input or an output message is the IBM Integration Bus internal representation of the message.

In IBM® Integration Bus, the message assembly is the internal representation of a message. You can configure a message map to include the following message assembly components:

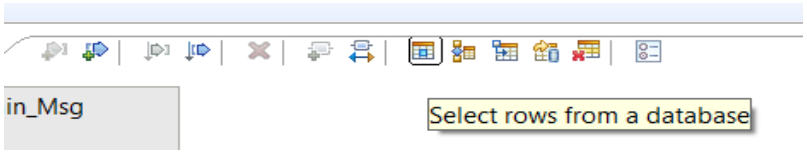


Additional inputs and outputs: Database tables

Database tables can be set as additional inputs and outputs of a message map.

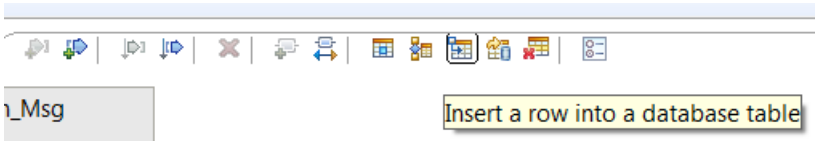
Inputs:

- **Select** transform

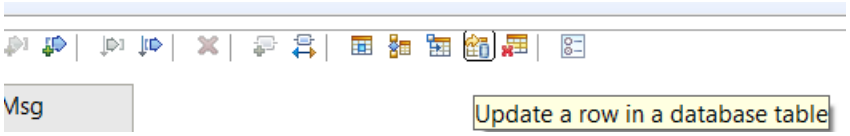


Outputs:

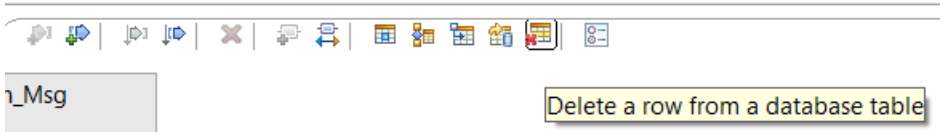
- **Insert** transform



- **Update** transform



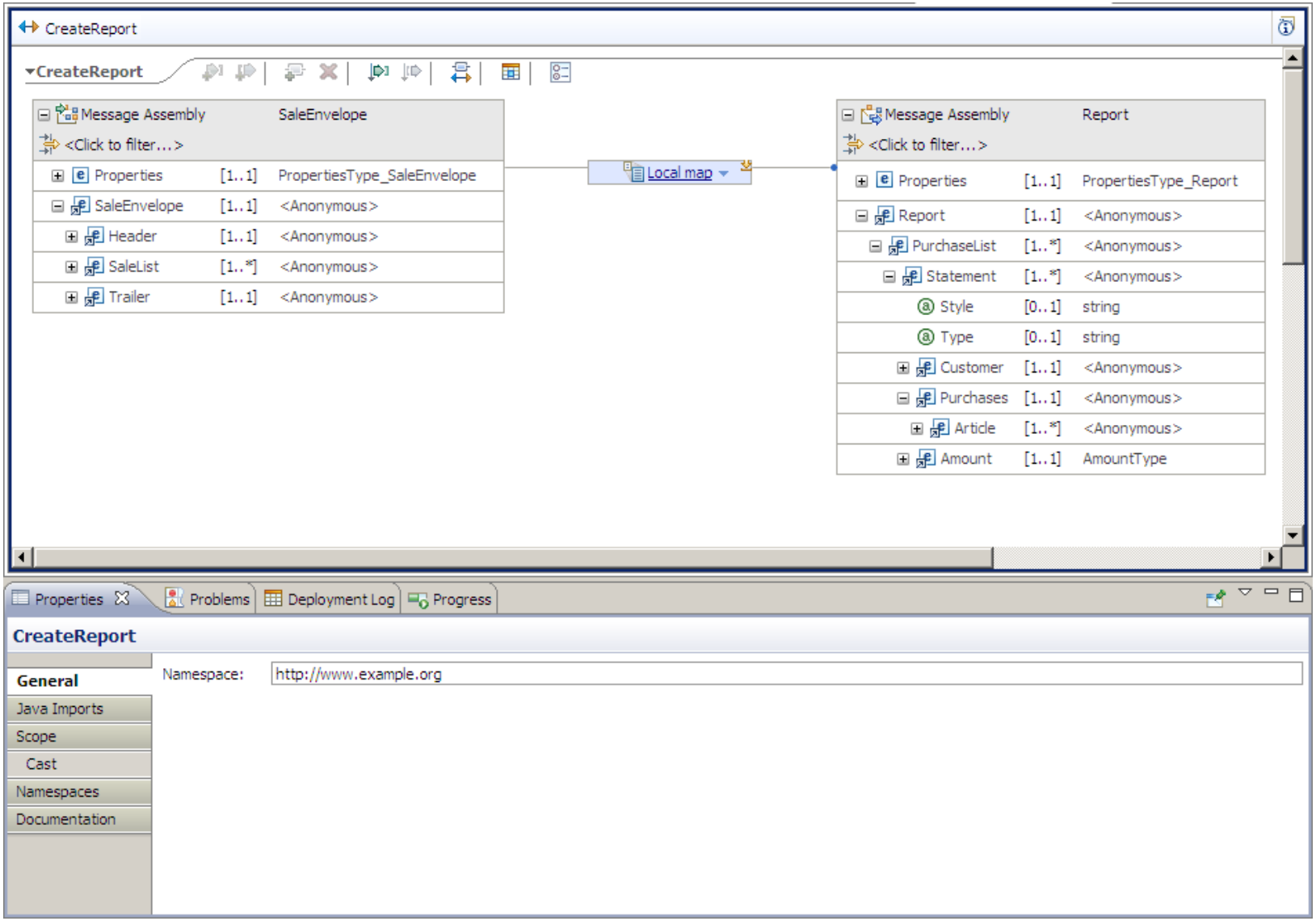
- **Delete** transform



Adding a message assembly component

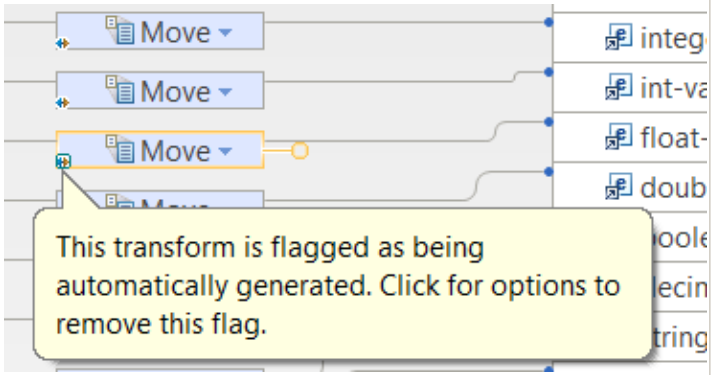
- ***Copy a message assembly component unchanged***: Do not include the component in the message map.
- ***Read elements of a message assembly component***: Add the component to the input message assembly only. The Mapping node passes it through unchanged
- ***Modify all the elements of a message assembly component***: Add the component such as the local environment tree to the input message assembly and to the output message assembly. Then, define transforms between each of its elements.
- ***Modify some elements of a message assembly component***: Add the component such as the local environment tree to the input message assembly and to the output message assembly. Define a Move transform for the entire component, that is, at folder level, and then specific transforms for each of the elements that you want to transform within an Override function.
- ***Initialize a message assembly component*** (create a new message assembly component in the output message): Add the message assembly component only to the output message assembly.
- ***Add a message assembly component***: Add the message assembly component to the output message assembly and populate at least one field. The Mapping node creates a new output structure containing the results of your transformations.
- ***Delete a message assembly component from the input message***: Add the message assembly component to the output message assembly and do not set any field.

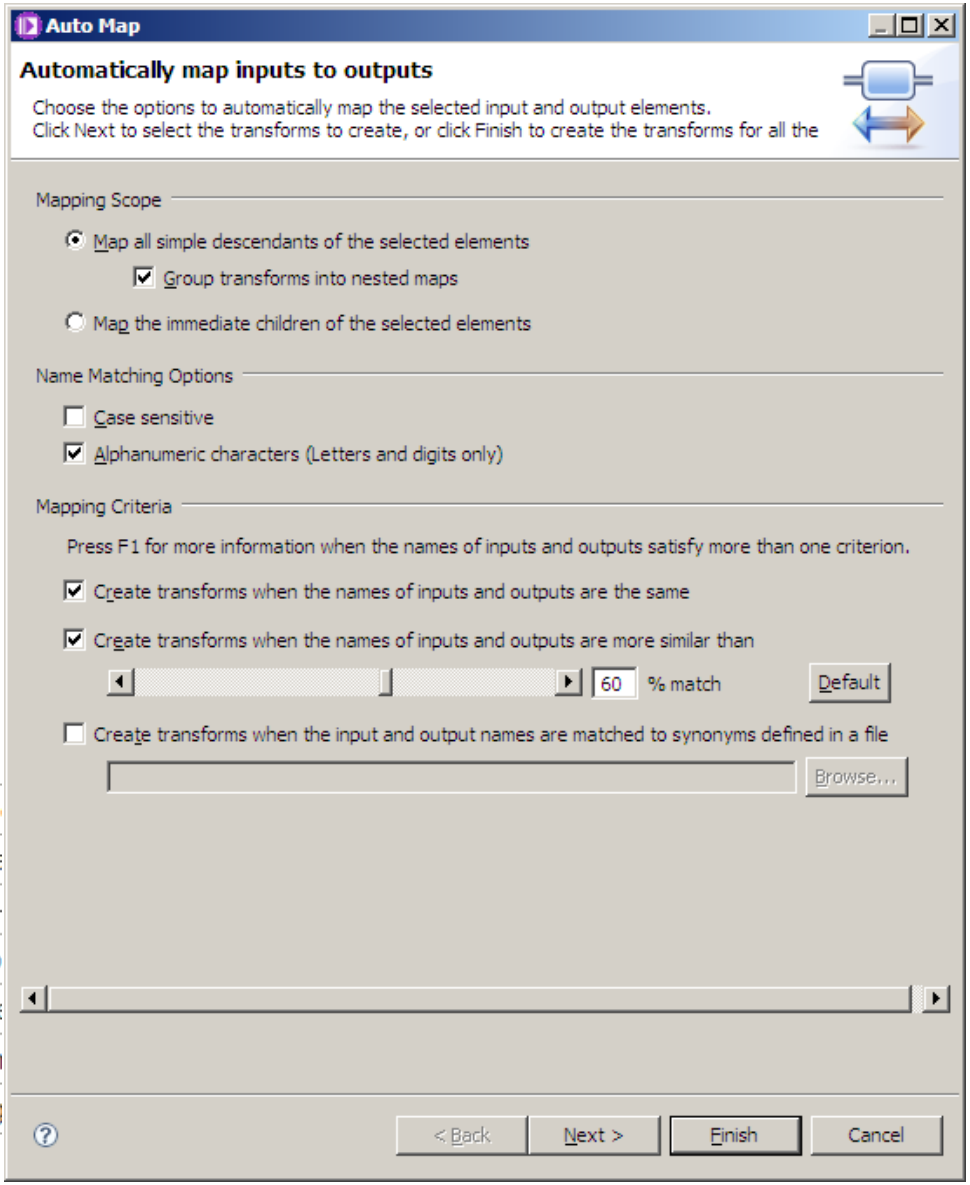
Graphical Data Mapping editor



Automap

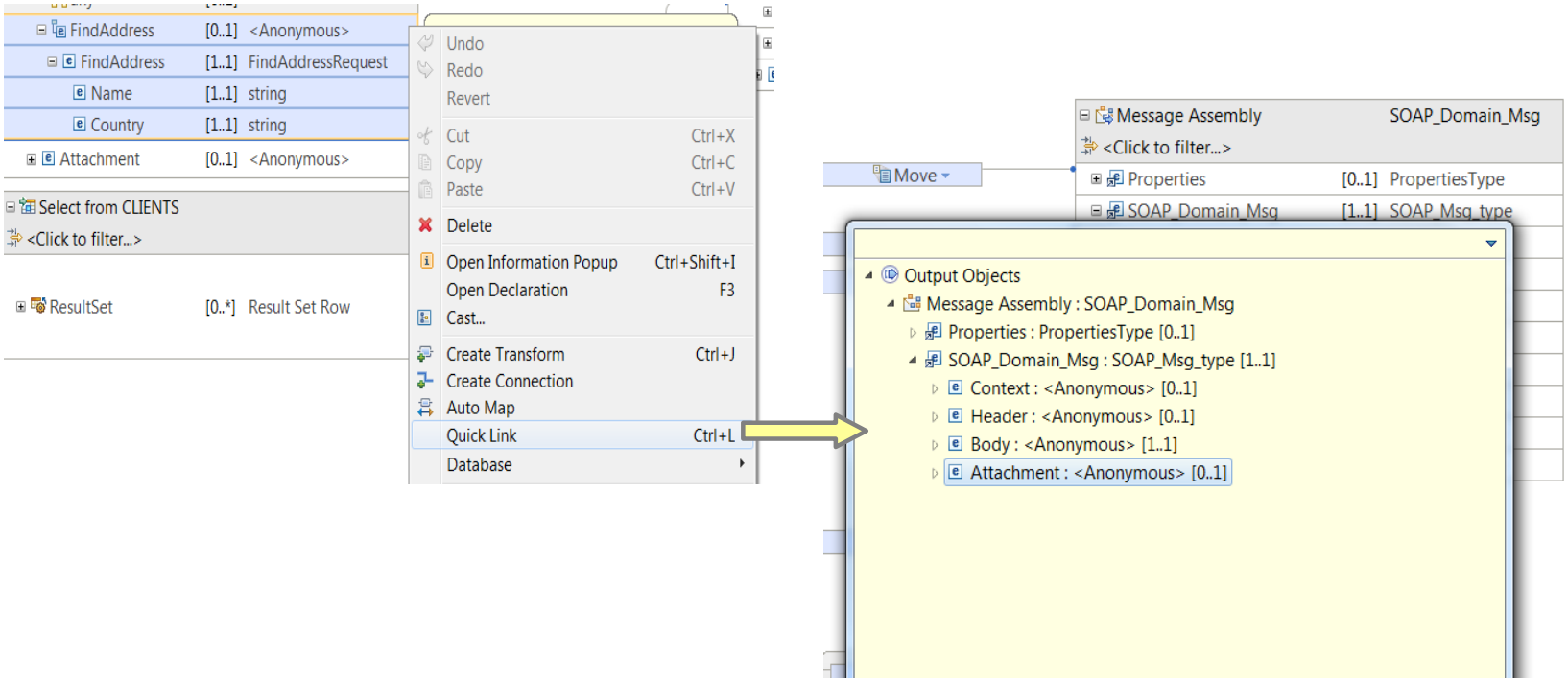
- Automates the task of producing transformations (***Move*** transforms) between source and target elements of the same or similar names
- Useful for working with large schemas
- You can use Auto map as a quick fix when you create a Local map or another nested mapping, and a warning or error marker is displayed to prompt you to complete the nested mapping.





Quick Link

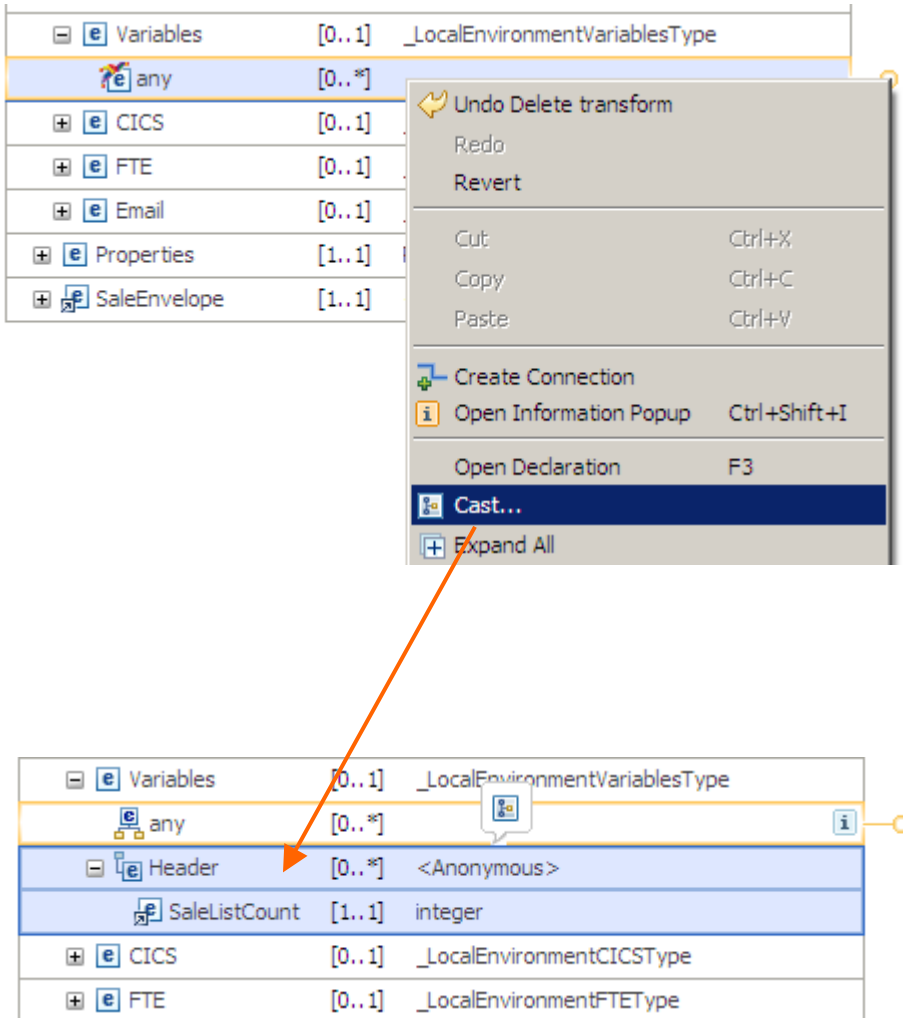
- You can then use the quick outline view and its built-in filter to find and select the required element.
- When you have selected the required element, a transform is created in the Graphical Data Mapping editor.



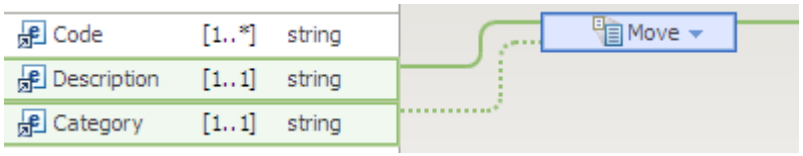
Mapping xs:any on an input or output message

- Option 1: You can use the ***Cast function*** to redefine parts of the input or output model in a graphical data map by specifying the specific simple or complex global type defined in a particular schema file.

- Option 2: You can define a transform, such as the ***Submap transform***, and define the input and output xs:any elements within the nested map of the transform.
  - Move transform
  - If transform
  - Submap transform
  - Custom XPATH transform
  - Custom Java transform

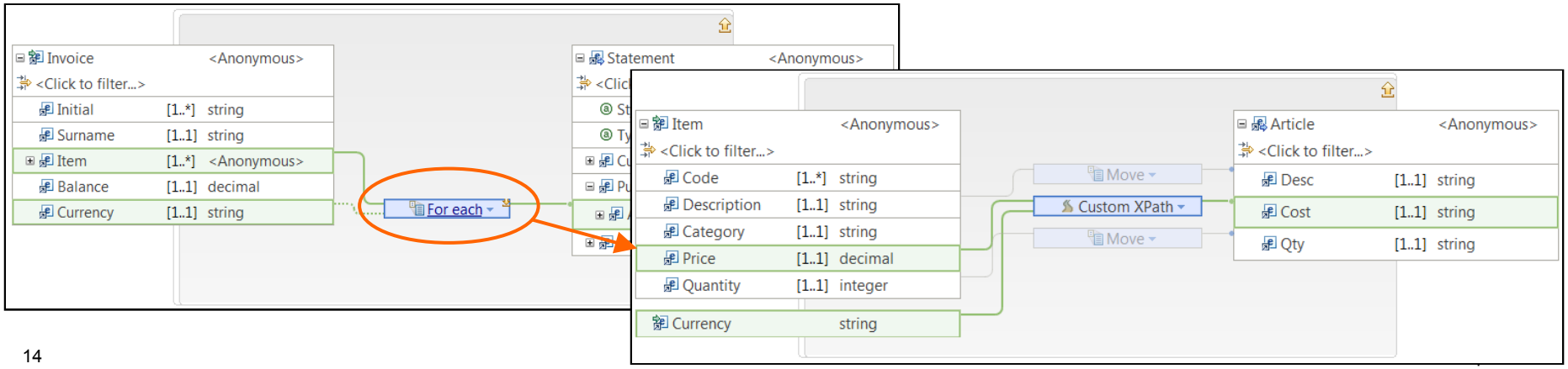


Primary and Supplementary inputs and connections



- Primary inputs are directly involved in the transformation:
  - They contain the data used to produce the output.
  - They can also be used to define properties of the transform (E.g. *Condition* property).

- Supplementary inputs are used to pass elements into a transform without affecting the primary purpose of that mapping:
  - They are NOT used directly in the transformation.
  - They can be use to define transform properties.
  - You can have as many supplementary inputs as required.
  - They can be used to pass extra data into a nested map. E.g. Supplementary inputs (both repeating and non-repeating) can be passed into the ‘For each’.

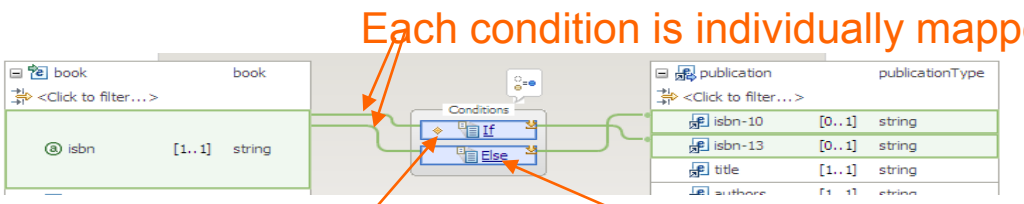


Transforms and functions

- **Core mapping transforms**:
  - You can use built-in structural and functional mapping operations, which enable you to graphically construct the required message transformations to build the output message.
- **Custom transforms:**
  - You can use custom transformations to build your own XPath 2.0, Java™, or ESQL functions, which can be invoked to perform specialized transformations within the message map.
- **XPath functions (fn:<functionName>):**
  - You can use XPath 1.0 and XPath 2.0 functions to transform data in a message map.
- **Database transforms:**
  - You can use the Select transform to query one or more database tables, and retrieve data that you can use in the message map to set output element values, define conditions, or use as input to build other transforms conditions.
  - New in IIB v9: You can use a database routine transform to call a stored procedure from a database, and retrieve data that you can use in the message map to set output element values, define conditions, or use as input to build other transforms conditions.
    - Note: *Only IBM® DB2® stored procedures are supported in IBM Integration Bus*

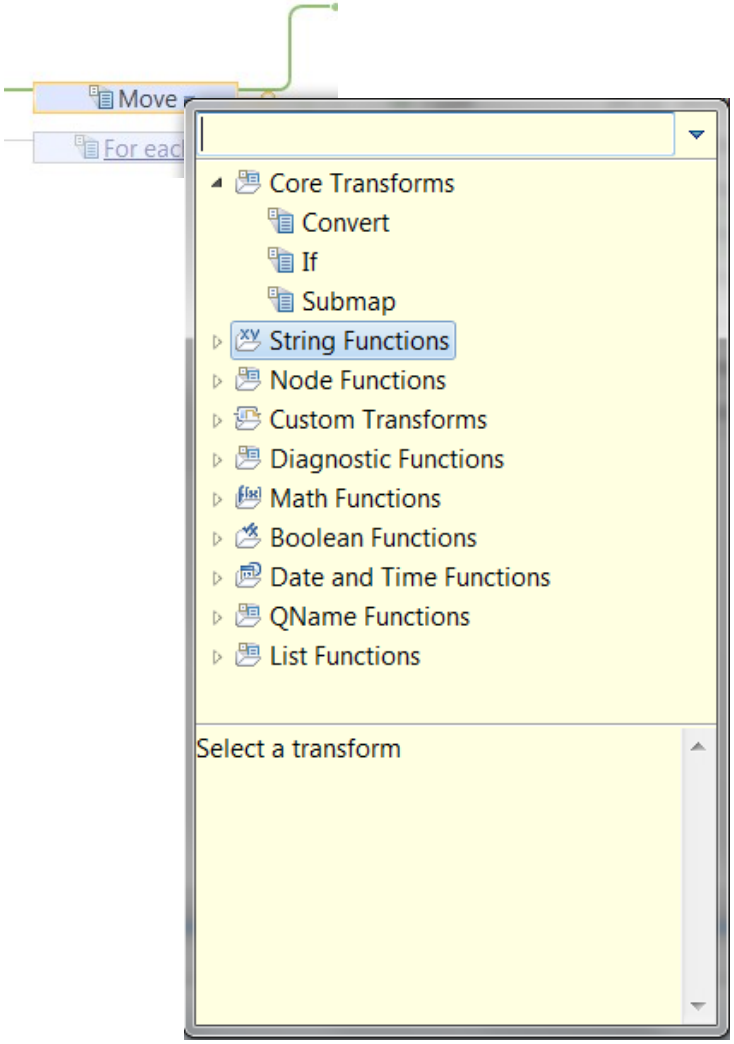
Structural transforms

- Maps can be broken down into a hierarchy of 'nested' maps using one of the structural transforms:
  - ***Local***
    - Map a complex source to a complex target in a separate 'page'
    - Reduces clutter in the mapping editor
  - ***For each***
    - Iterate over a repeating source element (array) and produces the same number of target elements
    - How the structure gets mapped is determined by the nested map
  - ***Append***
    - For appending one array to the end of another
    - Has two (or more inputs), one output.
    - The order in which the inputs are processed (appended) is specified in the 'Order' tab in the properties page
  - ***Join***
    - For merging the data from two or more arrays
    - Arrays are correlated by specifying a join condition (XPath 2.0 predicate)
  - ***Submap***
    - Similar to local, but the mapping is defined in a separate map file and is reusable
  - ***If / Else*** transform
    - Allows 'else' condition to be mapped as well
    - 'Grouped transform' containing as many else-if conditions as necessary
    - Each clause contains a nested map

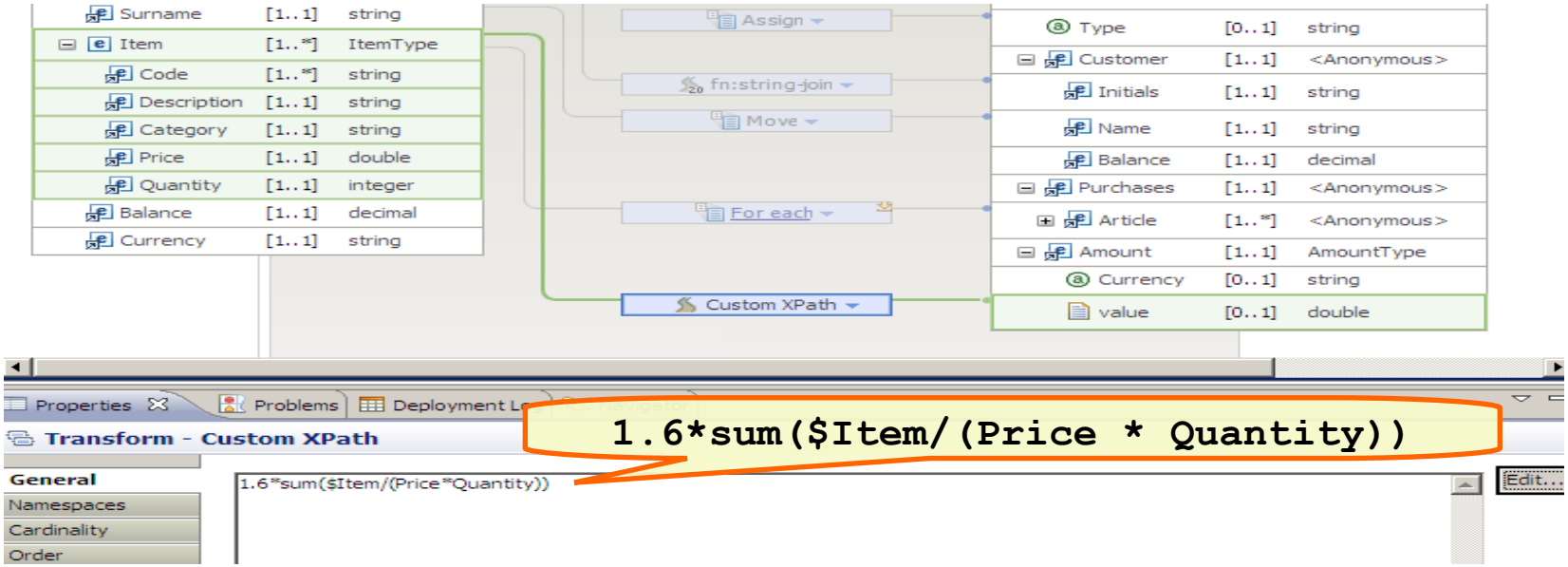


Functional Transforms

- The target value can be computed by applying a function to one or more inputs
- Large function library from XPath 2.0
- String manipulation
  - E.g. concatenation, sub-string, matching, find/replace, regex
- Numeric calculation
  - E.g. counting, summing, rounding, min/max etc
- Date/time processing
  - E.g. creating timestamps, extracting components of date/time, duration processing



Custom Transforms



- ***Custom XPath*** transform
  - Use this transform to write custom transformation logic by using XPath 2.0 expressions

- ***Custom Java*** transform
  - Use this transform to call Java methods from within a message map

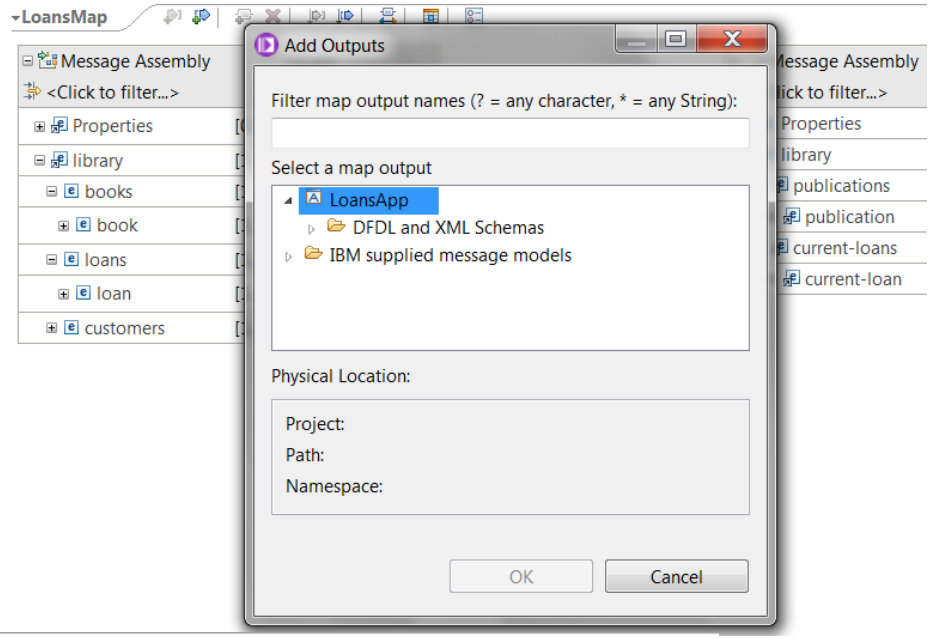
- ***Custom ESQL*** transform
  - Use this transform to call ESQL code from within a message map

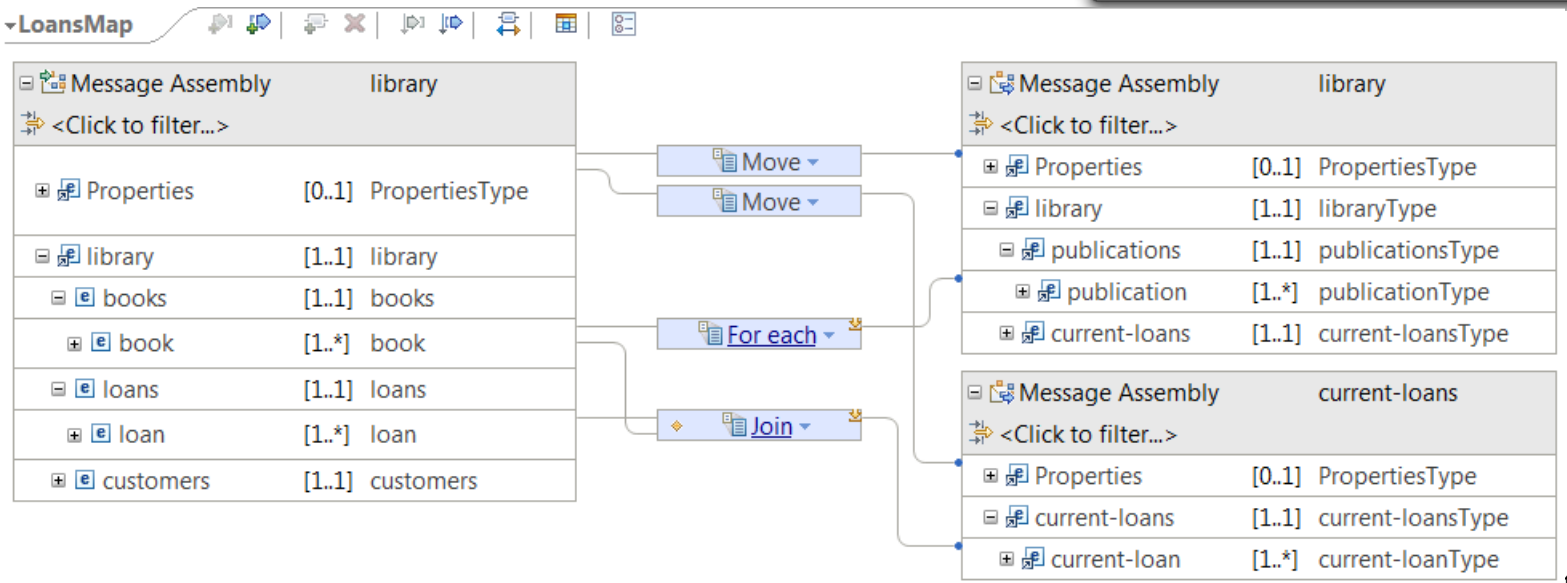
Mapping operations to modify data in a database

- ***Insert transform***:
  - You use the Insert transform to add one new row of data, or multiple rows of data, into a database table.
- ***Update transform***:
  - You use the Update transform to modify a row of data, or multiple rows of data, in a database table.
- ***Delete transform***:
  - You use the Delete transform to delete a row of data, or multiple rows of data, in a database table.
- ***Database routine transform (New in IIB v9)***:
  - You use a database routine transform to call a stored procedure from a database to insert, delete, or update data in one database table.

Multiple Outputs – Splitting

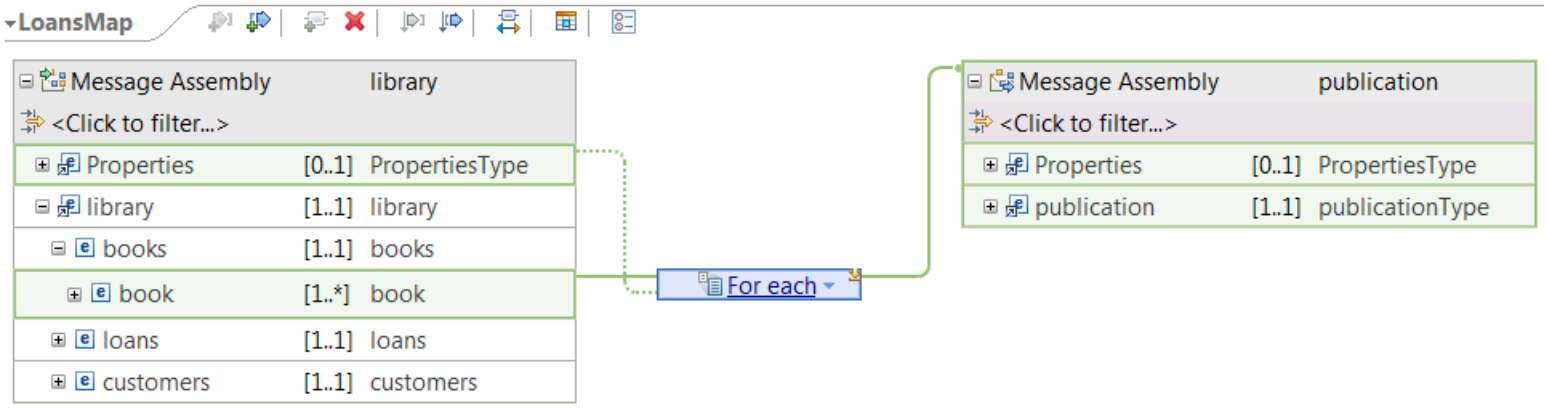
- Multiple target trees can be added to the map
- Each target propagates one message per input message





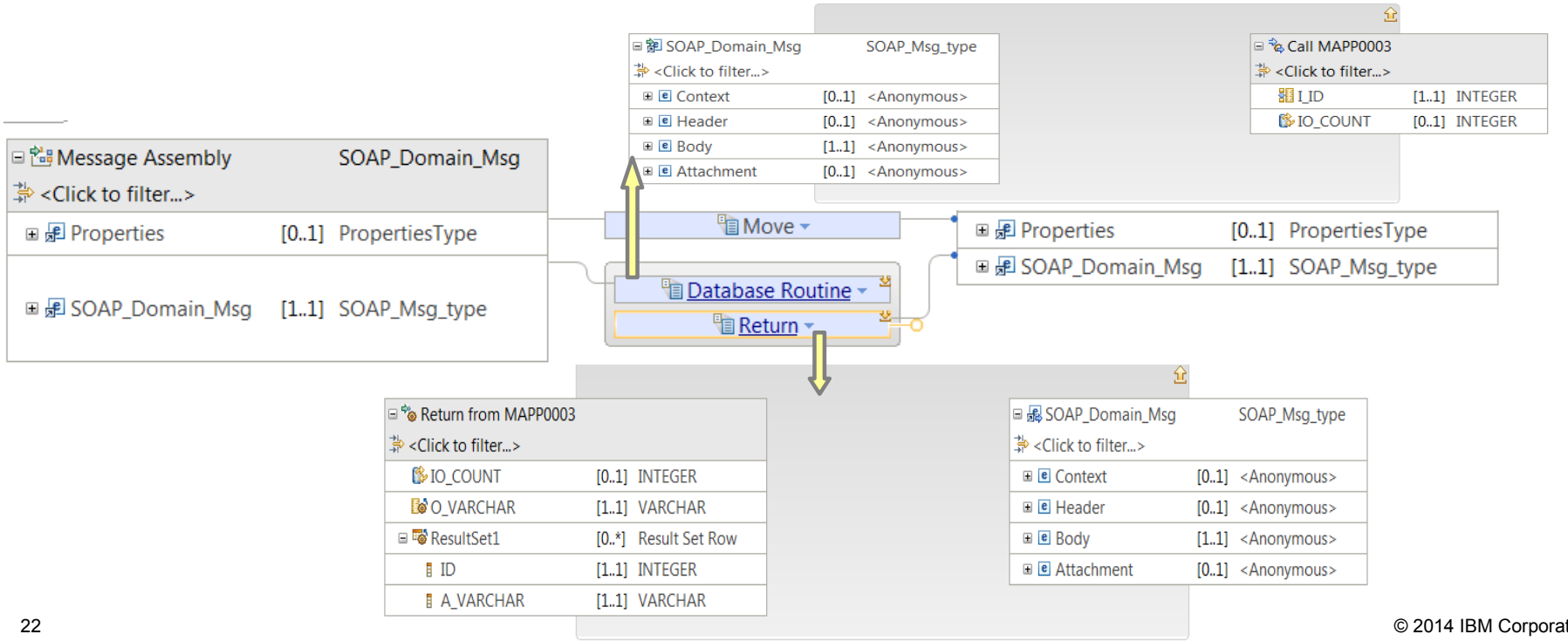
Multiple Outputs – Shredding

- A repeating input element can be mapped to the target message assembly (top level element)
- 'For each' transform will propagate one output message per input element
- Other parts of the input tree can be referenced by adding supplementary inputs to the 'For each'



Store procedures New in IIB v9

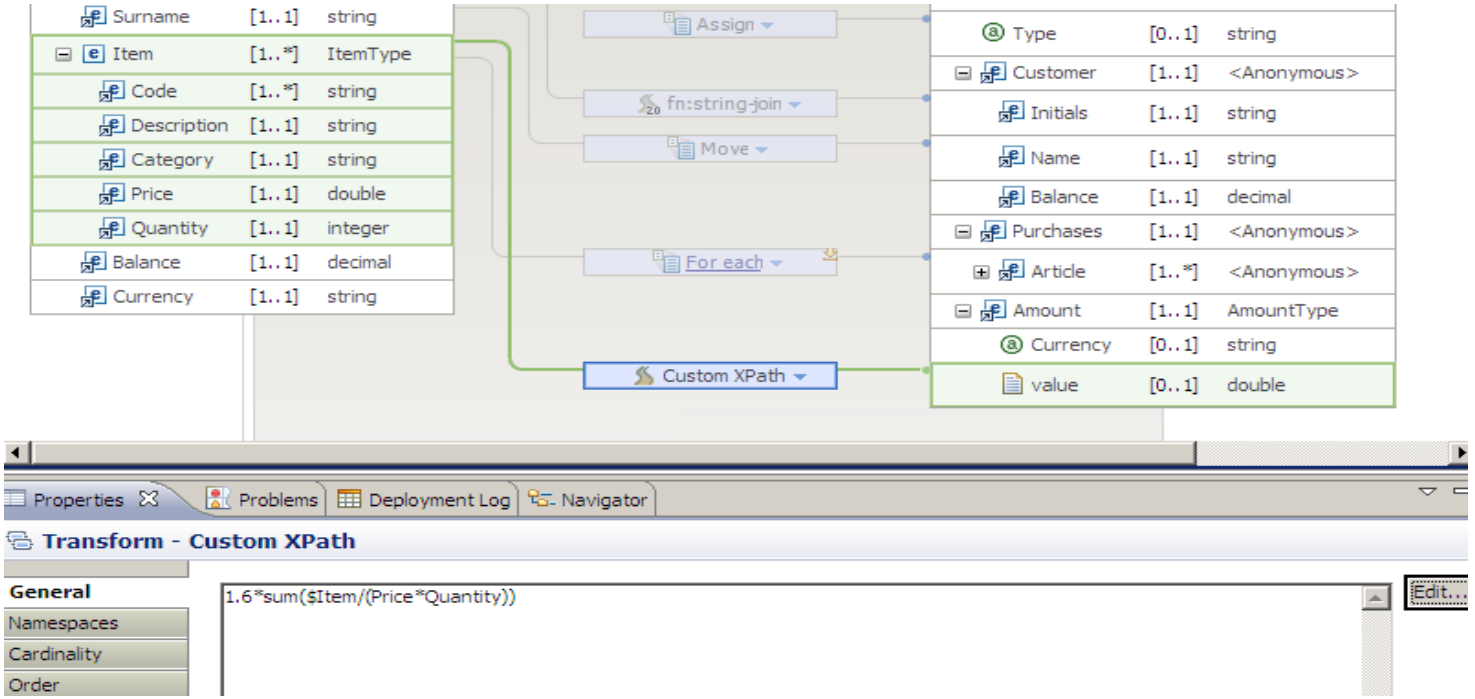
- Use the ***Database Routine*** transform to call a stored procedure from a database.
  - Only IBM DB2® stored procedures are supported.
  - Use the nested mapping of the Database Routine to provide values for the routine input parameters
- Use the ***Return*** transform to map any output parameters, return values, or ResultSets produced by calling the Database Routine.
  - You can also connect inputs from the message to the Return transform so that they can be mapped to the output message on succesful call of the routine
- Optionally, if you want the map the handle any possible DB exception, use the ***Failure*** transform.



Accessing user-defined properties from a Mapping node

New in IIB v9 FP1: A Mapping node can access properties that you have associated with the message flow that contains the node.

- To access these properties from a Mapping node, call the function mb:getUserDefinedProperty("propertyname") from a custom XPath mapping. The function returns a string that contains the property value, regardless of the original type of the property.



Transforming a SOAP message

AddressBook

Interface

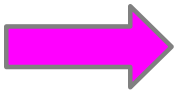
Configuration

Name	AddressBook
Namespace	http://AddressBook

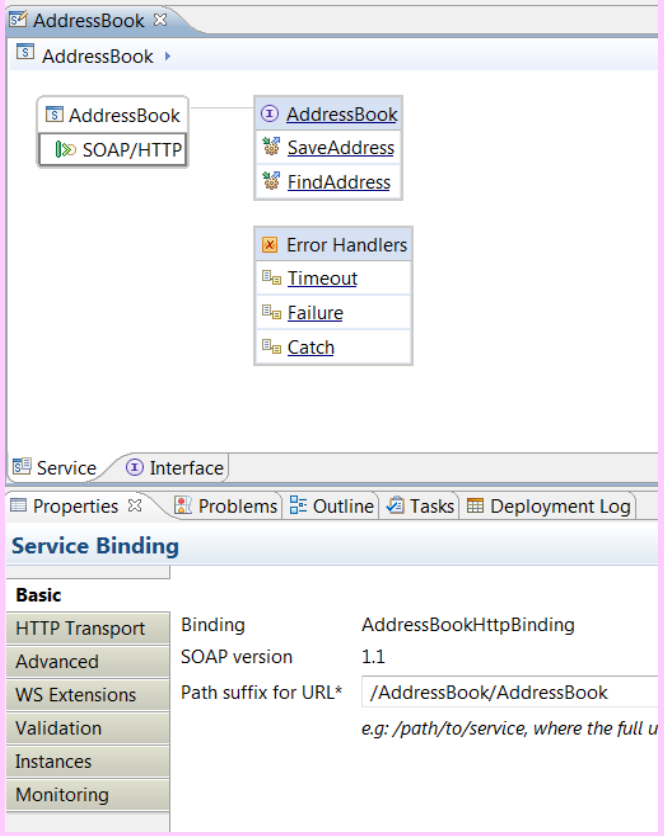
Operations

Operations and their parameters

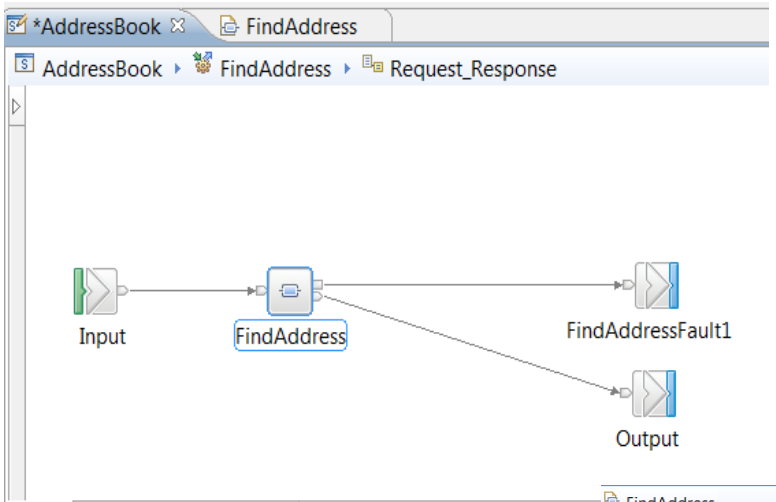
Message Type	Name	Type
SaveAddress		
SaveAddress	Person	PersonType
SaveAddressRe...	SaveAddressResponse	boolean
FindAddress		
FindAddress	FindAddress	FindAddressRequest
FindAddressRes...	FindAddressResponse	FindAddressResponse
FindAddressFau...	FindAddressFault1	FindAddressFault

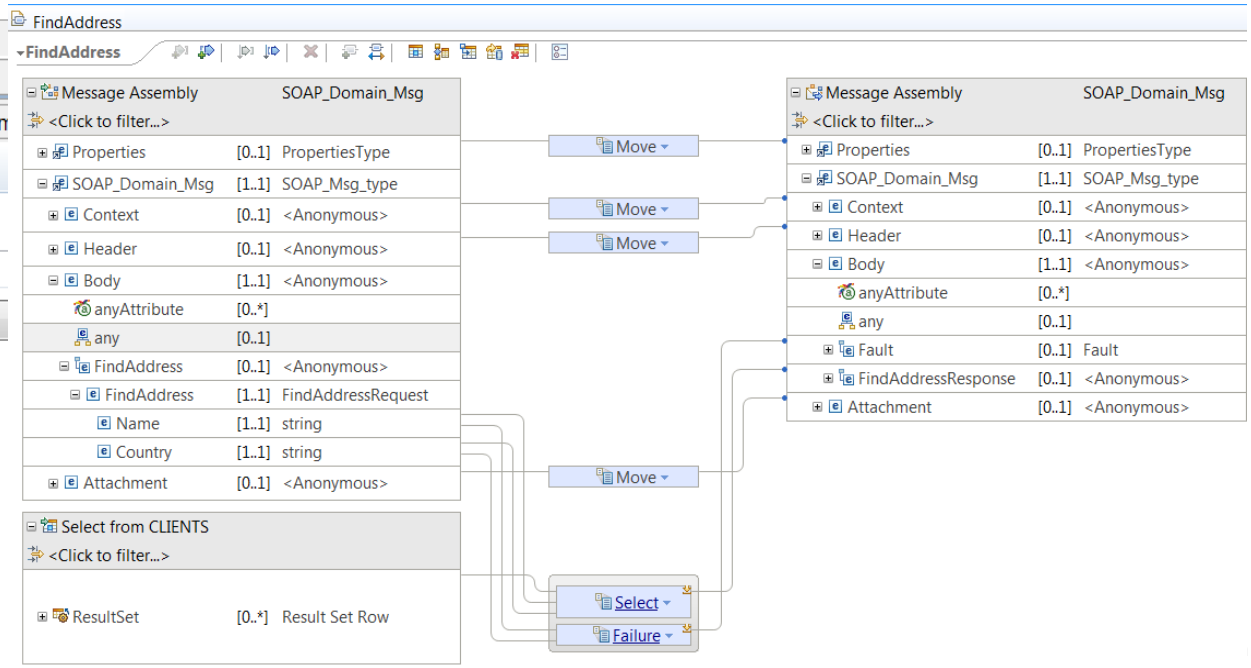


Integration service

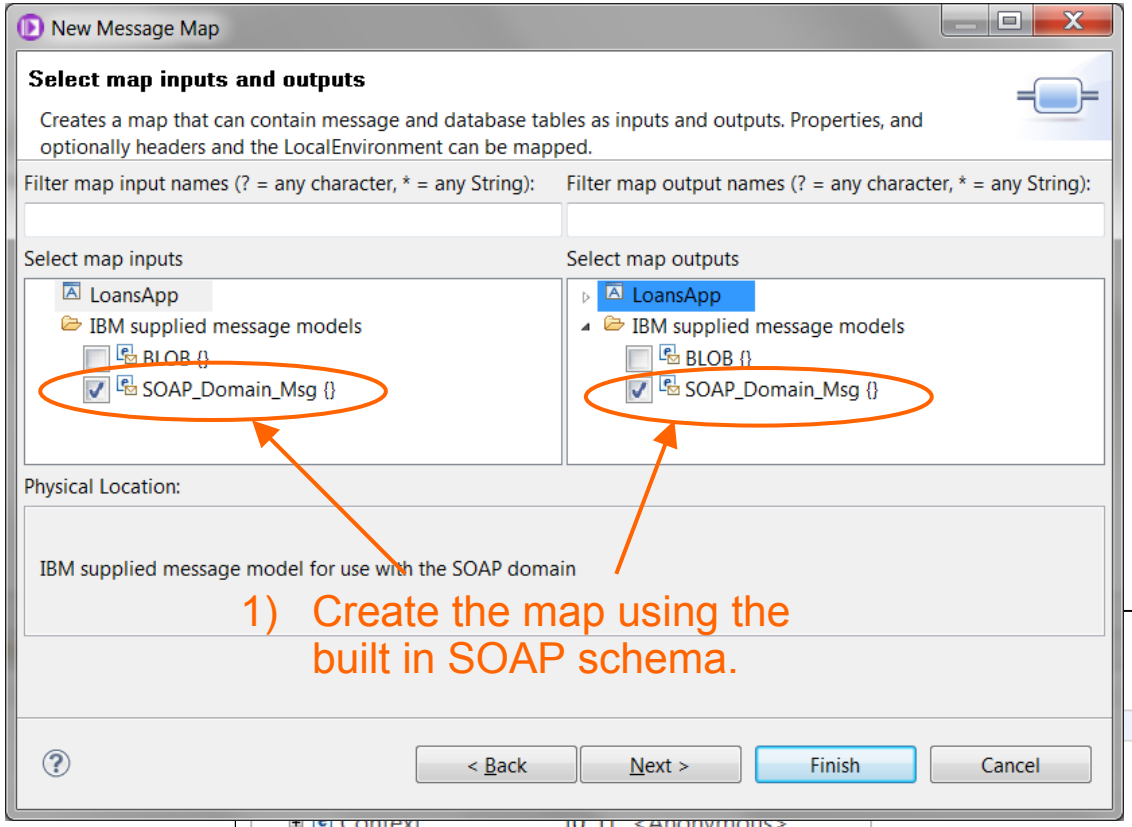


Transforming a SOAP message

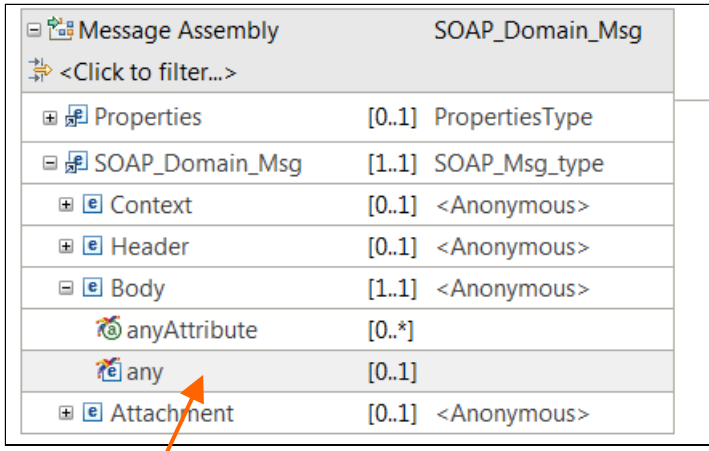




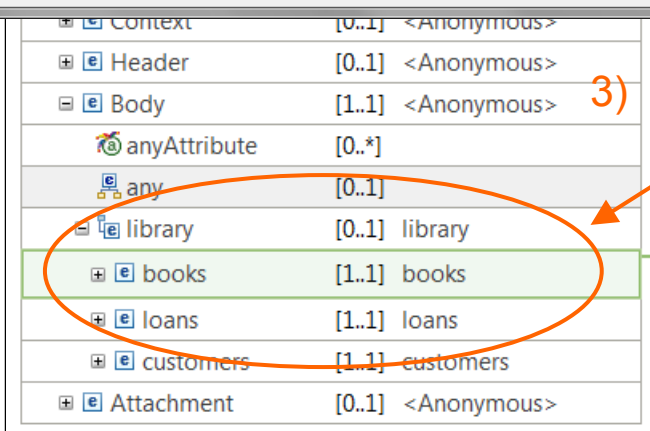
Mapping a SOAP message



1) Create the map using the built in SOAP schema.

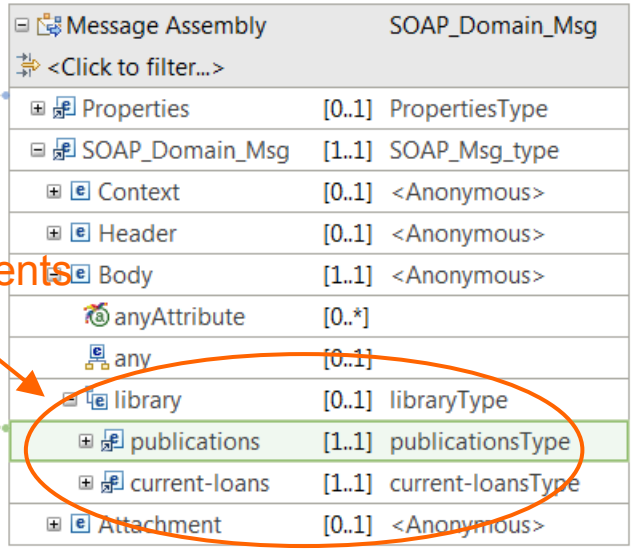


2) Right click -> Cast… select required element



3) Map the cast elements

Local map



Executing a message map

- Working with databases:
  - At design time, you must have a database definition file (.dbm file) in an available Data Design project for each database that you want to access. A data definition file contains one connection per database system.
  - At runtime:
    - You must have a JDBC connection of Type 4 defined for each database that your message map uses.
    - You must configure a JDBCProvider configurable service per database.
    - The JDBCProvider service name for a runtime database must be the same name as the development database name that you use in your message map.
- New in IIB v9 FP1: When the function level is changed to 9.0.0.1 or later using the ***mqsichangebroker*** command, the message maps are prepared for execution on deployment instead of when the first message is flowed through the node.
  - There is no drop in performance from initialization when the first message is flowed through the node.
  - The map and its dependencies, such as any referenced message models, are validated during deployment to ensure that the map runs successfully on first message.
  - All map dependencies, such as a valid message model, must be resolved at deployment for the deployment process to complete.
  - When IBM Integration Bus is restarted, dependencies are validated before the message flow can be restored.

Troubleshooting a graphical data

- A user trace will track the progress of the message though the map
- Logs messages when
  - individual transforms are entered and exited
  - The input tree is navigated
  - Output tree elements are created
  - Values are assigned to the output
  - Variables are assigned/updated or go out of scope
  - Functions are invoked, including user defined functions

```
…
2012-03-12 12:07:53.044555    96400   UserTrace   BIP3962I: The Mapping node is assigning the value "Twister" to the current output element.
2012-03-12 12:07:53.044647    96400   UserTrace   BIP3956I: The Mapping node is exiting the transform "Move".
2012-03-12 12:07:53.044689    96400   UserTrace   BIP3955I: The Mapping node is entering the transform "Move".
2012-03-12 12:07:53.044815    96400   UserTrace   BIP3959I: The Mapping node is traversing the input tree by using the nodetest "element(Price)" and the relationship "child". The number of matching elements is "1".
2012-03-12 12:07:53.044876    96400   UserTrace   BIP3960I: The Mapping node is adding a new element with name "out:Cost" into the output tree.
…
```

Additional information

- IBM Integration Bus version 9 infocenter
  http://pic.dhe.ibm.com/infocenter/wmbhelp/v9r0m0/index.jsp?topic=%2Fcom.ibm.etools.msgbroker.helphome.doc%2Fhelp_home_msgbroker.htm
- IBM Integration Community blog
  https://www.ibm.com/developerworks/community/blogs/c7e1448b-9651-456c-9924-f78bec90d2c2/?lang=en
- IBM Integration Community wiki
  https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/W37b629a0f7aa_4709_9506_bba2a096693d
- Open technologies for integration
  https://github.com/ot4i
- DFDL schemas on github
  https://github.com/DFDLSchemas
- IBM Integration media -YouTube
  https://www.youtube.com/user/IBMintegrationMedia
- DeveloperWorks technical library
  http://www.ibm.com/developerworks/websphere/zones/businessintegration/wmb.html

Copyright Information

-
- Coremetrics is a trademark or registered trademark of Coremetrics, Inc., an IBM Company.
- SPSS is a trademark or registered trademark of SPSS, Inc. (or its affiliates), an IBM Company.
- Unica is a trademark or registered trademark of Unica Corporation, an IBM Company.
- Java and all Java-based trademarks and logos are trademarks of Oracle and/or its affiliates. Other company, product and service names may be trademarks or service marks of others. References in this publication to IBM products and services do not imply that IBM intends to make them available in all countries in which IBM operates.