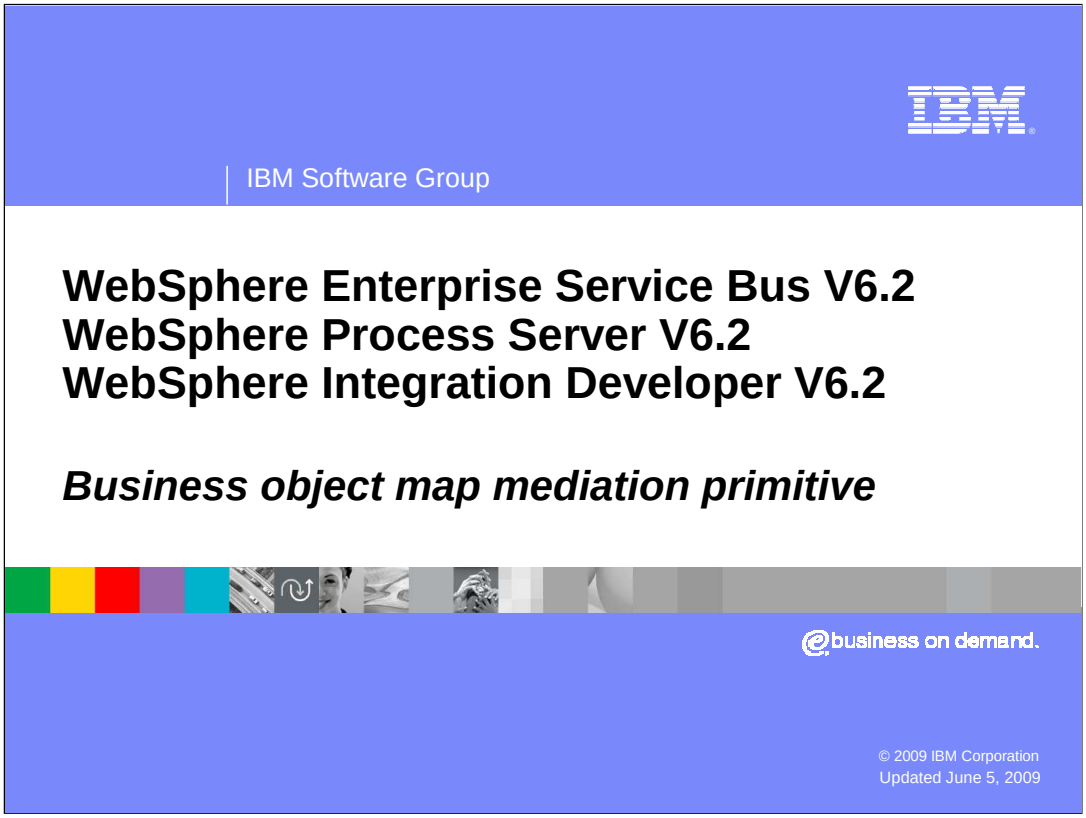


This presentation provides a detailed look at the business object map primitive.

Goals

- Understand the business object map primitive

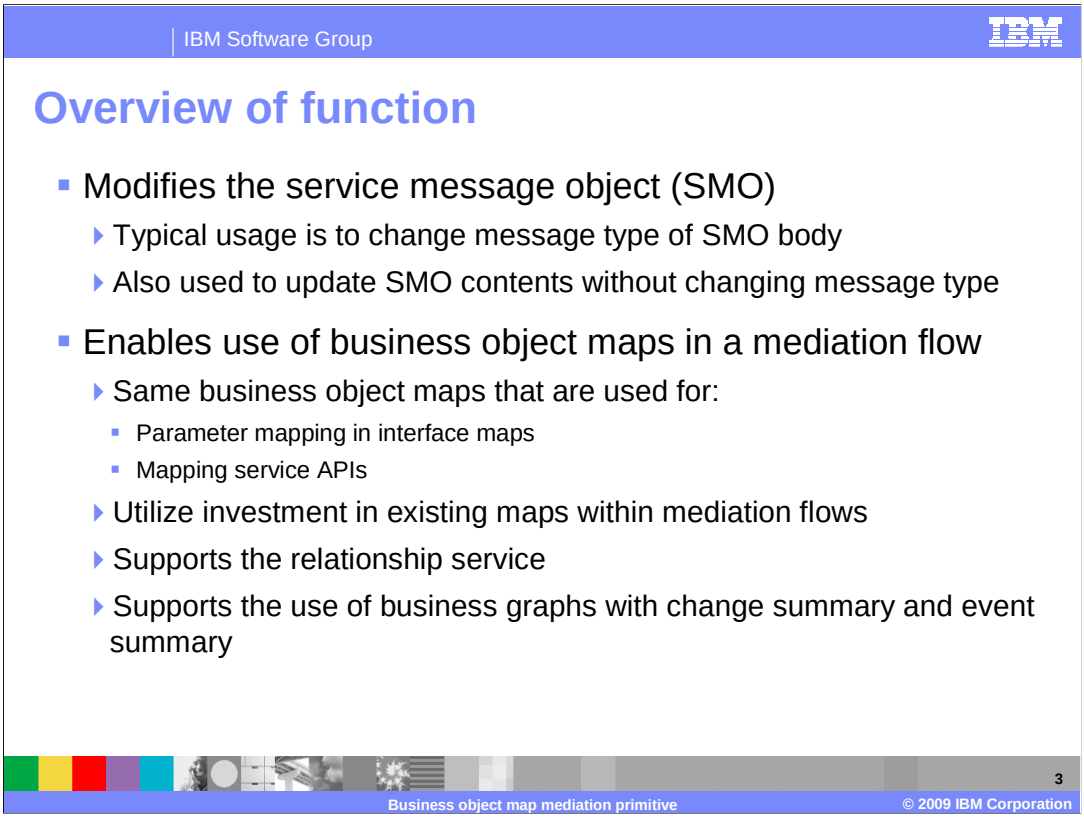
Business object map

- Overview of function
- Use of terminals
- Definition of properties
- Mapping editor capabilities
- Raising events
- Error handling
- Example usage

The goal of this presentation is to provide you with a full understanding of the business object map primitive.

The presentation assumes that you are already familiar with the material presented in the presentations that cover common elements of all mediation primitives, such as properties, terminals, wiring and the use of promoted properties. The general knowledge of mediation primitives they provide is needed to understand the business object map primitive specific material in this presentation.

An overview of the function provided by the business object map primitive is presented along with information about the primitive's use of terminals and its properties. The capabilities of the mapping editor used for creation of business object maps is described. The common event infrastructure events that business object maps can produce are examined. Some of the error handling characteristics are described and an example usage of a business object map is provided.



The purpose of the business object map primitive is to modify the service message object (SMO). In most cases, they are used to change the structure of the SMO body, thus changing the message type of the SMO. However, in some cases the business object map might be used to update the contents of the SMO without changing the message type.

The business object map primitive enables the use within a mediation flow of the same business object maps that are used in WebSphere® Process Server. In that context, they are used for parameter mapping within interface maps and are invoked through the mapping service APIs. The business object map primitive enables you to make use of an existing investment in business object maps within your mediation flows. You can also develop new maps that can be shared by mediation flows, interface maps and the mapping service APIs. The relationship service for both identity and look up relationships is supported as is the use of business graphs, allowing the change and event summaries to be updated to reflect the mapping operation.

Overview of function

- Root of mapping operation must be one of body, context, headers or entire SMO
- Can configure event settings to raise CEI events
- Business object mapping editor
  - Invoked from within mediation flow editor
  - Provides multiple transform types
  - Supports the use of variables
  - Toolbar with multiple functions, such as:
    - Map similar fields
    - Filters, sorting and hide/show options

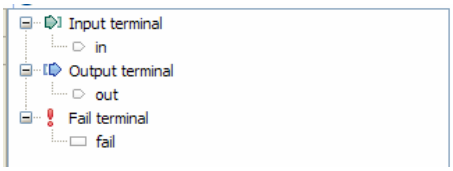
A business object map used with the business object map primitive must operate on the SMO. The root for the map can be the SMO body, the context, the headers or the entire SMO.

You can configure the business object map to raise CEI events. These can be related to the map as a whole or for individual transforms within the map.

The business object mapping editor is used to define your business object maps and can be invoked directly from within the mediation flow editor. The mapping editor can be opened from either the properties view of the business object map primitive, or directly by double clicking the primitive on the canvas of the mediation flow editor. The mapping editor provides a variety of different transform types and supports the use of local variables that can be used during the mapping operation. The editor also provides a toolbar with many useful functions, such as an automated capability to map similar fields between source and target. There are also filters, sorting and hide/show options that provide you variations on the visual display, making it easier for you to define your map. These are especially helpful when working with large business objects in your map.

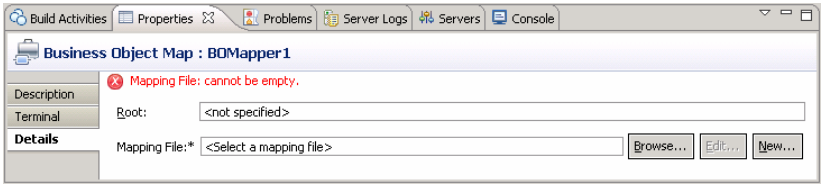
Terminals

- Terminals:
  - One input terminal
  - One output terminal
  - Fail terminal
- Terminal message types
  - Message type is not propagated between input and output terminals
  - Input and output determined:
    - Implicitly by the message type of the terminal they are wired to
    - Explicit by setting of message type
  - Fail terminal is the same message type as the input terminal



The business object map primitive has one input terminal, one output terminal and a fail terminal. The input and output terminals do not need to be of the same message type. Therefore, the message type is not propagated between these terminals. The message type for these terminals is determined implicitly by the message type of the terminal they are wired to, unless the message type has been explicitly set using the change terminal type dialog. The fail terminal always has the same message type as the input terminal.

Properties and promotable properties



- **Root**
  - Determines what portions of the SMO is mapped
  - Valid choices are: **/ /context /headers /body**
- **Mapping file**
  - File name of the business object map to use
  - **Browse…** provides a dialog to search for an existing map
  - **Edit…** opens the configured file in business object mapping editor
  - **New…** opens the new business object map creation dialog
    - If there is a currently a configured file, it is replaced with the new one
- Both of these properties are read only on this panel
- There are no promotable properties

There are only two properties for the business object map primitive. The Root property defines the portion of the SMO that is input to and output from the map. It can be specified as forward slash (“/”) meaning the entire SMO, “/context” or “/headers” for the context or headers portion of the SMO, or “/body” indicating the payload or body of the SMO.

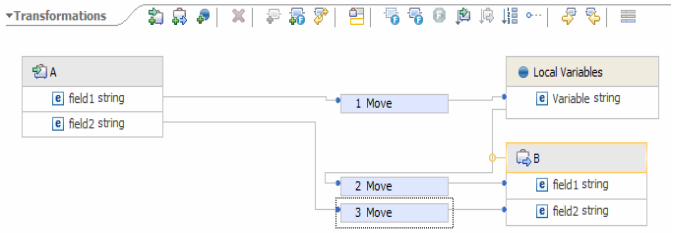
The second property is the name of the Mapping file which defines the business object map used by the primitive. There are three buttons associated with this property. The Browse… button opens a selection dialog to allow you to select an existing map from within the project or from a dependent library. The Edit… button opens the currently configured mapping file in the business object mapping editor. The New… button opens a creation dialog where you define the name and root of a new map to be created. If there already is a mapping file configured, it is replaced by the new map.

These properties are read only as displayed on this panel. It is during the creation of a new mapping file in the new business object map creation dialog that the values for these properties are specified. Changing them requires the creation of a new map.

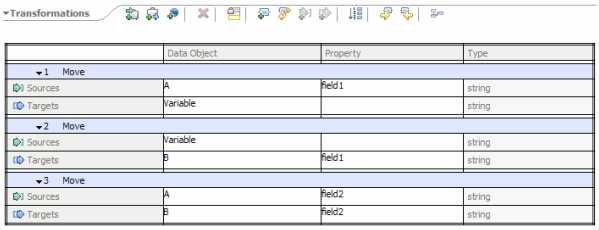
Neither of the business object map primitive properties is promotable.

Mode of operation

- Graphical View



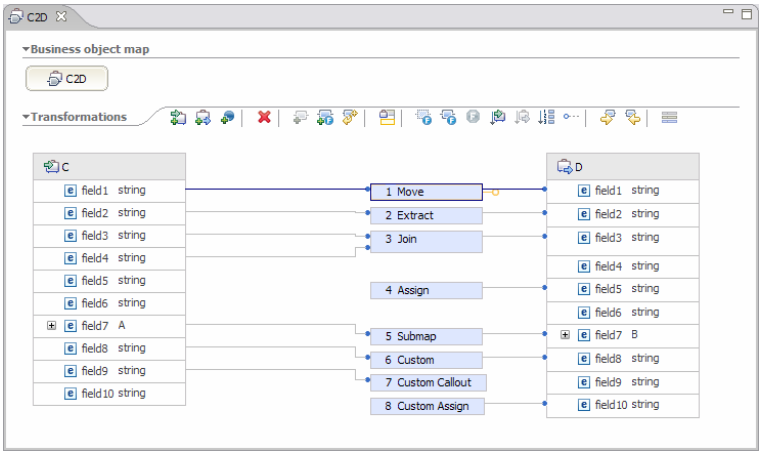
- Table View



The business object mapping editor has two modes of operation, the graphical view and the table view. Both of these modes of operation are used in conjunction with the properties view to fully define each of the transforms. The key differences between these modes is how the source and target of a transform are identified. In the graphical view, a wire is dragged from the source to the target. In the table view, the sources and targets panel of the properties view is used. Which mode of operation you use is based on your own individual preference.

Transforms

- Transforms processed in the designated order
- Move
- Extract
- Join
- Assign
- Submap
- Relationship
- Relationship Lookup
- Custom
- Custom Callout
- Custom Assign



This slide shows the graphical view of the mapping editor and lists the transforms that are available for use in a business object map. The source business object is on the left and the target business object is on the right. The transforms are shown in the middle, with wires connecting the source field, transform and target field. Notice that each transform has a number, designating the run order, which specifies the order in which the transforms are processed at runtime. The properties view is used to define the specifics for each type of transform.

The **Move** transform assigns the value in the source to the target. More details about the move transform are provided in the next slide.

The **Extract** transform takes a source string containing delimiters and extracts a portion of the string and assigns it to the target.

The **Join** transform combines the values of two or more sources into one value, separated by delimiters, and assigns it to a string target.

The **Assign** transform sets a constant value in the target.

The **Submap** transform is used between a source and target that are business objects. The input and output of the specified business object map must be of the same type as the source and target of the transform.

The **Relationship** transform performs relationship management for identity relationships defined for business objects.

The **Relationship Lookup** transform performs relationship management for lookup relationships defined for simple types.

The **Custom** transform uses Java™ code to perform custom mapping logic between one or more sources and one or more targets.

The **Custom Callout** transform uses Java code similar to Custom, except that it does not have any targets. This might be used to perform some initialization at the start of the map.

The **Custom Assign** transform is also similar to Custom, except that in this case it does not have any sources. It is useful for cases where the assign transform does not meet the requirements and Java logic is needed to construct the value to be assigned.

Move transform

- Simple types
  - Allowed when the simple types are the same type
  - Allowed when the simple types are of different types
    - Possibility of runtime errors if type conversions fail
- Complex types
  - Only allowed when Complex types are the same type
- XSD:anyType
  - Allowed when both fields are an XSD:anyType
  - Allowed between XSD:anyType and simple or complex types
    - Possibility of runtime errors if target type not compatible with actual source type

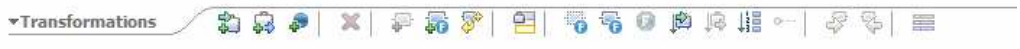
In a typical business object map, many of the transforms are move transforms. For simple types, a move can be made between a source and target that are of the same type or are different types. When the move is between different types, they must be of types for which a conversion is valid. At runtime, if a conversion fails, a MediationBusinessException occurs.

When the move transform is used for complex types, the source and target types must be the same.

Some business objects can contain fields that are defined as XSD:anyType. The move transform can be used when both source and target are XSD:anyType. Move is also allowed between XSD:anyType fields and simple or complex type fields. When the source is an XSD:anyType and the target is a simple or complex type, a runtime error can occur if the actual source value does not match the defined type of the target.

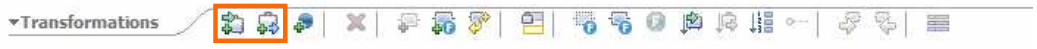
Local toolbar – Add input and output

- Local toolbar



  - Provides quick access to editor functions

- Add input and add output



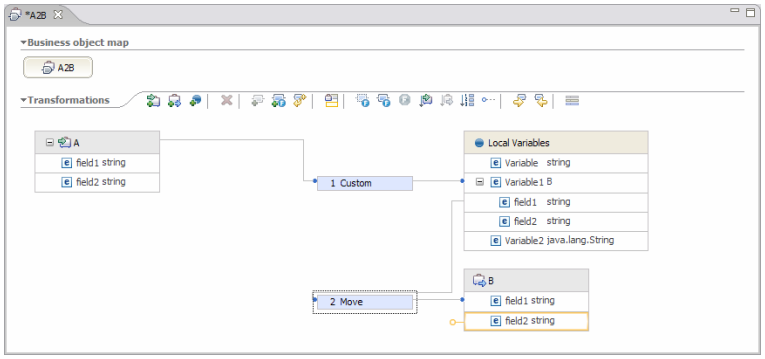
  - Used to add additional business objects to the map
  - Displays dialog to select business object to add
  - Not useful with business object map primitives
    - Inputs and outputs are within the SMO and are controlled by the mediation flow runtime

The local toolbar provides easy access to many of the functions provided by the business object mapping editor. This slide and the next several slides examine some of these functions. The add input and add output buttons display a dialog that allow you to add additional business objects as inputs or outputs for the mapping operation. However, in the context of a business object map associated with a business object map primitive running in a mediation flow, this functionality is not useful. The input and output to the map is controlled by the mediation flow runtime and must match the root property specification of the primitive. Adding an additional input or output causes initialization errors to occur in the mediation flow runtime.

Local toolbar – Add variable

- Add variable

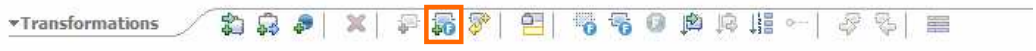
  - Used to store temporary values within a map
  - Can be a simple type, business object type or Java type



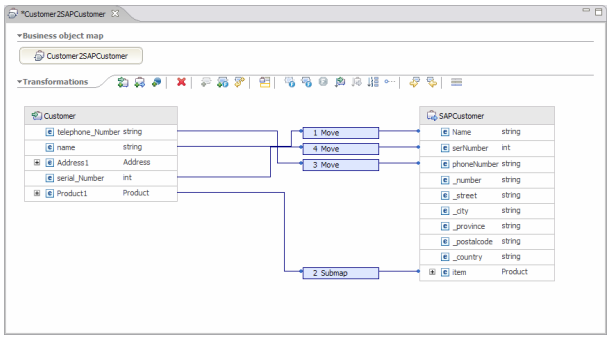
The add variable button allows you to define a local variable that can be used during the mapping operation. These local variables allow you to store temporary values within a map for use during the transforms. The variable can be a simple type, a complex business object type or a Java type.

Local toolbar – Map similar fields

- Map similar fields



  - Automatically creates simple transforms

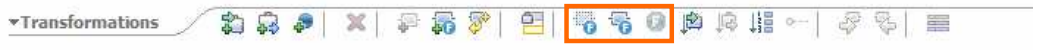


- Based on similarity
  - Names do not have to match exactly
  - Types do not have to be identical
- Optionally select input/output data objects for mapping analysis
- Must review results for correctness
  - Some transforms created might not be what you intended

The map similar fields button performs an automatic mapping between fields in the source and target business objects. The fields do not have to have the same name nor do they have to be of the same type. An analysis is done of both name and type to determine where transforms should be created. For example, in the screen capture you can see that telephone_Number in the source was matched with phoneNumber in the target based on name similarity. However, Product1 in the source was matched with item in the target based on the fact that they are both of the same complex type, Product. When performing mapping of similar fields, it can be done over the entire source and target business objects, or can be narrowed to be performed between selected portions of the source and target. The transforms created from this type of mapping might not always yield the results you require and therefore you need to review what was created. Some transforms might need to be deleted, some might need to be edited to a different transform type and some that were not created might need to be added.

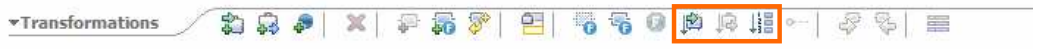
Local toolbar – Filter and sort fields

- Filter fields



  - Show only:
    - untransformed fields
    - transformed fields
    - all fields (default)
  - Applies only to outputs
  - Useful when working with large data objects

- Sort fields



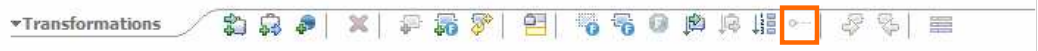
  - Sort transforms by source (input fields)
  - Sort transforms by target (output fields) (default)
  - Sort transforms by run order
    - Shown top-down based on run order

The filter fields buttons are used to control which fields in the output business object are visible. The three buttons let you switch between showing only untransformed fields, showing only transformed fields and showing all fields. The entire source business object is shown and is not affected by these filters. When working with large business objects, this can be very useful in helping you see if all the transforms you need have been created.

The sort fields buttons allow you to sort the transforms in the order of source fields, in the order of target fields or in the defined run order of the transforms. The first two result in unscrambling the wires between the transforms and the fields on which the sort is performed, making it easier for you to see which transforms are associated with which fields. The last one helps you analyze if you have the run order correct.

Local toolbar – Hide connections

- Hide connections

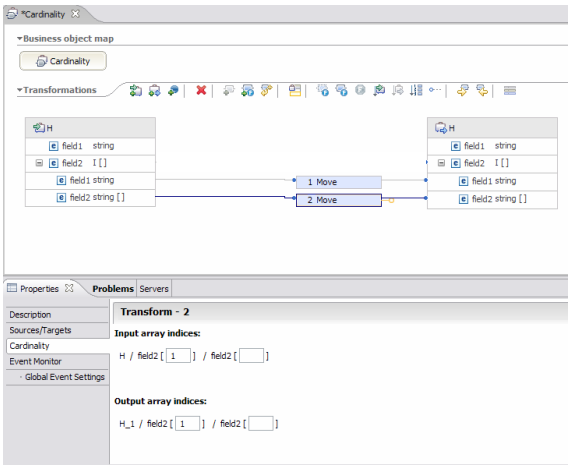


  - Toggles if connections are hidden or shown
  - Connections hidden based on sort order
    - Sort by source yields hide target connections
    - Sort by target yields hide source connections
    - Sort by run order hides all connections
  - Selecting a transform will cause its connections to appear
  - Useful to remove connection line clutter in large maps
  - Hide connections
    - Off by default if map has less than 50 connections
    - On by default if map had 50 or more transform

The hide connections button provides a function that is a companion to the sort order capabilities described on the previous slide. When the connections are sorted on the source or target side, the other side can often be a confusing tangle of wires. Using the hide connections, the tangled connections are hidden from view. When an individual transform is selected, the hidden connection for just that one transform is displayed, allowing you to easily see which field it is associated with. This is a very useful function for large maps and is turned on by default in maps with over 50 connections.

Cardinality

- Array iteration done with:
  - Submap transforms
  - Move transforms if the elements are the same type
- Arrays can be indexed to a specific element
  - Must be an integer constant
  - Base index is 1 (index=1 is the first element)



- Iteration can only occur at the end of transform path for input and output arrays
- Arrays within the transform path for input or output must be indexed

When mapping business objects that contain arrays, transforms can be defined that iterate over an entire array or an index can be used to indicate a specific element within an array. This slide looks at the rules for array handling and the cardinality panel in the properties view.

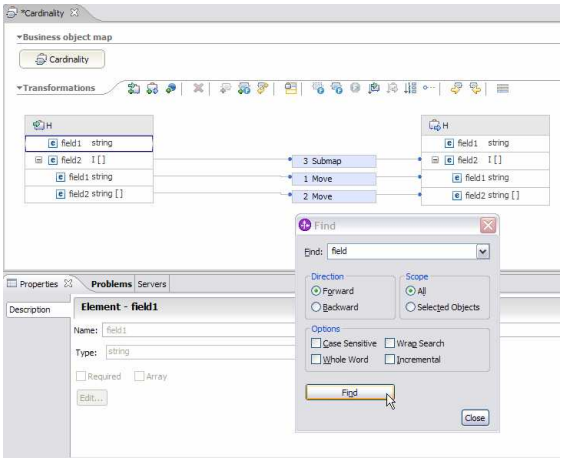
When iterating over an entire array, either a move or a submap transform can be used. The move transform can be used if the source and target array elements are of the same type. However, if the array elements are of different complex types, then a submap transform must be used.

In order to index into a specific element within an array, the index must be specified in the cardinality panel of the properties view. The index must be a constant and cannot be specified using a variable. The base index is one and not zero, so an index of one references the first element of the array.

In order to operate over an entire array, the array must be the last element in the path to the source or target. When an array occurs within a path to the source or target, that array must be indexed to a specific element. An example of this is shown in the screen captures on this slide.

Find

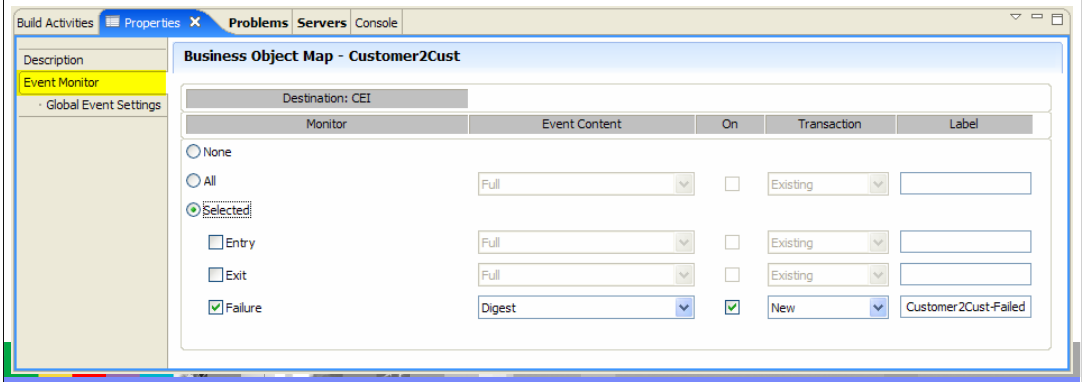
- Find dialog accessed using:
  - Edit -> Find in main menu
  - ctrl+f key combination
- Searches for matches in
  - Input data objects
  - Local variables
  - Output data objects
  - Transforms
- Supports
  - Forward/Backward search
  - Case Sensitive
  - Whole word
  - Wrap Search
  - Incremental Search
  - Scoped Search



Another useful function, especially when working with large business objects, is the find dialog. It is accessed through the Edit, Find menu option or by using the ctrl+f key combination. When performing a search, it includes the source and target business objects, local variables and transforms. The search can be scoped to only selected objects if needed. Typical find capabilities such as forward and backward searching, case sensitivity, whole words only, wrapping and incremental searching are also supported.

Raising events

- Maps can be configured to raise events
  - For the map
  - For individual transforms in the map
- Event points are entry, exit and failure
- Event content choices are empty, digest and full



Business object maps can raise CEI events. This capability is configured using the event monitor panel in the properties view as is shown in the screen capture. The configuration of events can be associated with a map and can also be associated with individual transforms within a map. Events can be raised upon entry or exit of the map or transform and can also be raised when failures occur. The contents of the event are controlled using the settings of empty, digest and full. Also, a label can be defined that is included in the event.

This slide shows two examples of events that were raised from a business object map. On the top you can see an event that was raised because of a failure in a map. The event point data is highlighted in yellow showing the type of event, the configured payload type, the label assigned to the event and the name of the map in which the failure occurred. The application data is only partially shown, as it contains an entire stack trace from the time of the failure. The key problem is highlighted in green, indicating that it was a failure in the move transform that is number six in the run order.

In the lower part of the screen is an exit event for a transform. The event point data, highlighted in yellow, is similar to the kind of information provided in the other event. It contains additional information that the event is related to the transform which is number three in the run order. In the application data highlighted in green you can see path of the target field and the value that it contains after the transform.

Error processing

- MediationRuntimeException raised for:
  - Configured map file does not exist
  - No map configured for primitive
- MediationBusinessException (fail terminal flow)
  - Failure of a transform such as data conversion error
  - Input or output business object not part of the SMO

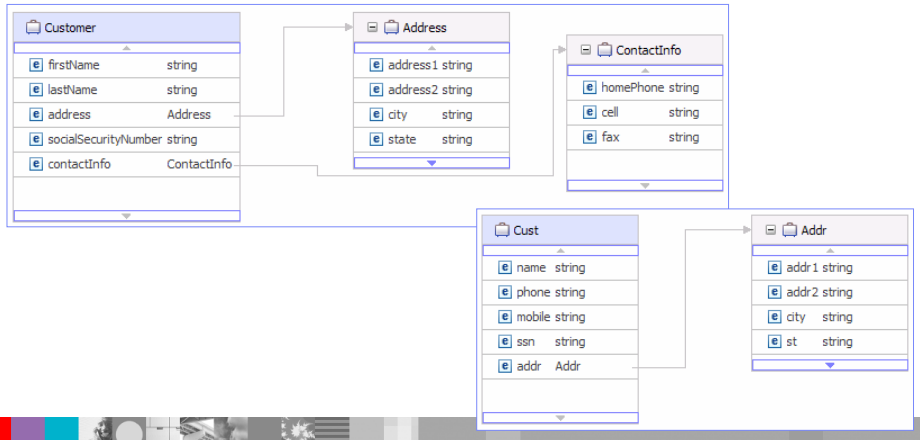
The error processing details and considerations are examined in this slide.

A MediationRuntimeException is raised in the case where the map file configured for the business object map primitive cannot be found. This can occur when the map is deleted or removed from the project without the configuration of the primitive being updated. WebSphere Integration Developer does not flag this situation as an error, so it is possible to not detect this problem at development time and therefore encounter this runtime error. Another situation that can produce a MediationRuntimeException is when a business object map primitive does not have a configured mapping file. This situation is flagged by WebSphere Integration Developer and therefore is not likely to be seen, but it is conceivable to occur during an iterative development and testing cycle.

There are a couple of situations that can cause a MediationBusinessException. For these cases, if the fail terminal is wired, the fail terminal flow is taken rather than the flow failing with an exception. One cause of a MediationBusinessException is when there is any failure of a transform, such as when a conversion error occurs. Another situation that can lead to a MediationBusinessException is when there is an input or output business object in the map that is not part of the SMO. This can occur if the wrong map is configured for the business object map primitive or if an additional input or output was mistakenly added to the map.

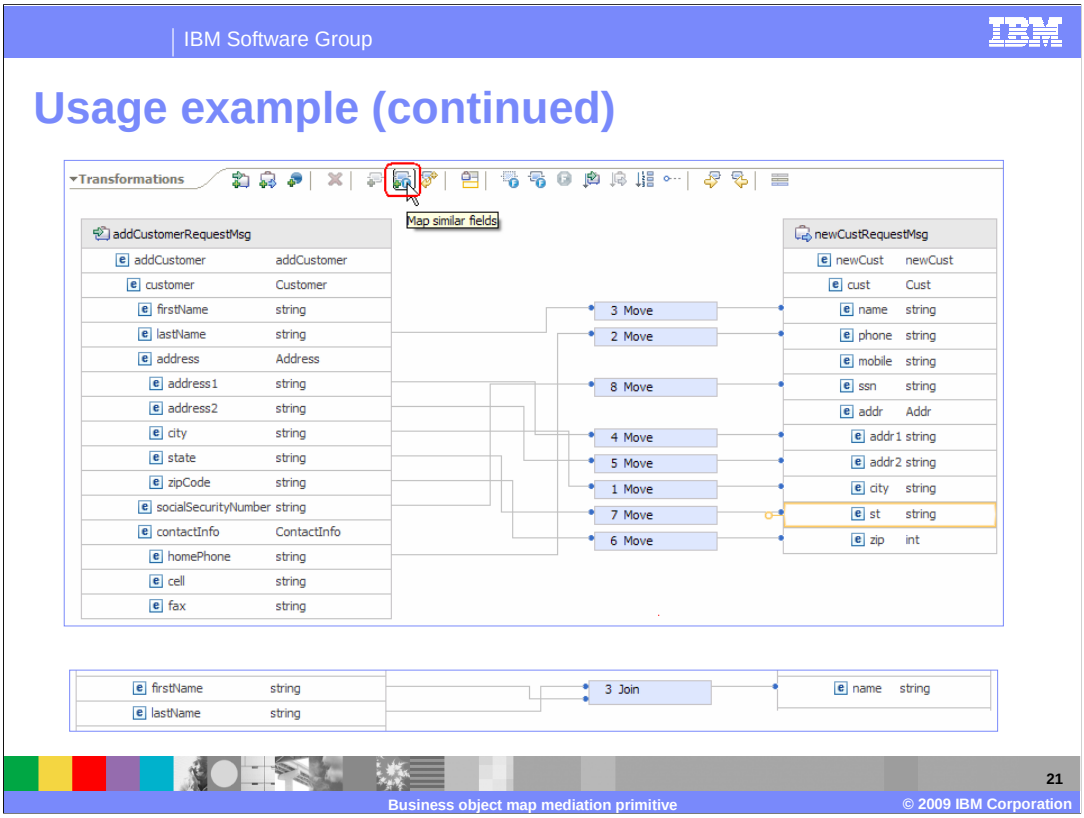
Usage example

- Merger of two companies
- Transformation of customer records
- Use map similar fields to develop transforms



The next two slides illustrate an example usage of the business object map primitive. The scenario being shown is the merger of two companies and the transformation that needs to occur within a mediation flow to transform a customer record in one company's format to the other company's format. The purpose of the example is to illustrate the map similar fields capability to provide the initial mapping between the two formats.

On this slide you can see the two formats. One company has a Customer business object with simple types of firstName, lastName, socialSecurityNumber and the nested business objects of type Address and ContactInfo. The other company has a Cust business object, with the simple types of name, phone, mobile, ssn and a nested business object of type Addr.



The top screen capture shows the graphical view of the mapping editor and highlights the map similar fields button that is used to produce the mapping that is shown.

Notice that the mapping produced transform number eight that matched the field socialSecurityNumber with the field ssn. Both of these are simple types contained in the top level business objects.

Also, notice that transform number six matches the fields zipCode, which is a string, to the field zip, which is an int. In this case, the simple types that were matched were both contained in nested business objects and their types are different.

Transform number two matches homePhone which is contained in a nested business object with phone that is in the top level business object.

All of these transforms, along with most of the other generated transforms, are the required result and do not need to be changed when reviewed.

However, notice transform number three which matched lastName with name as a move transform. In this case, what is really needed is a join transform that combines lastName, a comma and then firstName into the name field. This is a case where you have to edit the generated transform, as is shown in the screen capture at the bottom of the slide.

Finally, the field named cell was not matched with the field mobile. In this case, you need to add a move transform to complete the map.

Summary

- Examined the business object map primitive

Business object map

  - Overview of function
  - Use of terminals
  - Definition of properties
  - Mapping editor capabilities
  - Raising events
  - Error handling
  - Example usage

In summary, this presentation provided details regarding the business object map primitive. It presented an overview of its function along with information about the primitive's use of terminals and its properties. The capabilities of the mapping editor, used for creation of business object maps, was described. The common event infrastructure events that business object maps can produce were examined. Some of the error handling characteristics were described and an example usage of a business object map was provided.

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_WBPMv62_BusinessObjectMapPrimitive.ppt

This module is also available in PDF format at: ../WBPMv62_BusinessObjectMapPrimitive.pdf

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

Trademarks, copyrights, and disclaimers

IBM, the IBM logo, ibm.com, and the following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

WebSphere

If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of other IBM trademarks is available on the Web at "Copyright and trademark information" at http://www.ibm.com/legal/copytrade.shtml

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

Other company, product, or 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 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.

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.