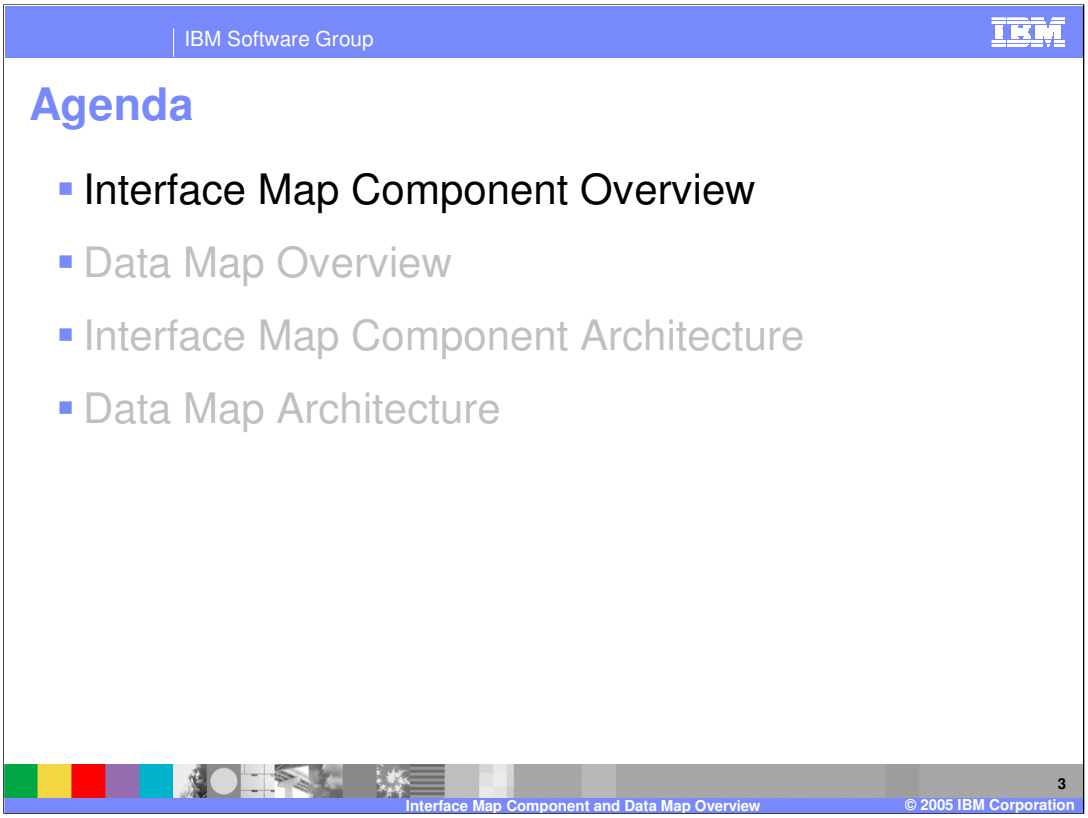
This presentation will provide an overview of the interface map component and data map features of WebSphere Process Server and WebSphere Integration Developer V6.0.

Goals

- Introduce the interface map component concept
- Introduce the data map concept
- Understand interface map component
- Understand data map architecture

© 2005 IBM Corporation

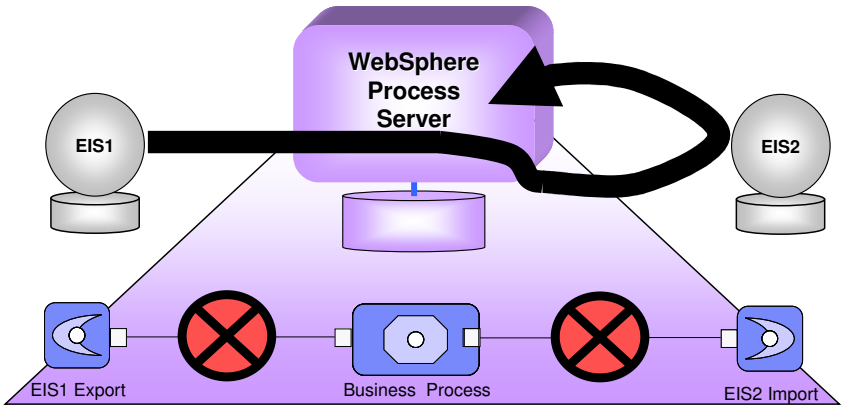
The goals for this presentation are to introduce the concepts of the interface map component and data maps and to help you understand the architecture.



The agenda is to first provide an overview of the interface map component.

Interface Map Component Overview

- Various components interact in an integration solution
- Interface mismatches exist between:
  - Components in the same module
  - Components in different modules
  - Components and external services

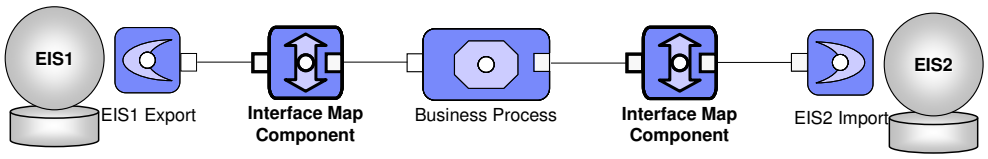


© 2005 IBM Corporation

In the process of integrating various systems, the components that make up these systems are likely to have different interfaces that do not match. Mismatches could be encountered between these interfaces in various places; for example, between Service Component Architecture (SCA) components within the same module, SCA components within different modules, or SCA components and external services, such as an enterprise information system. In the diagram, the red circles with the X represent the disconnect between components due to incompatible interfaces.

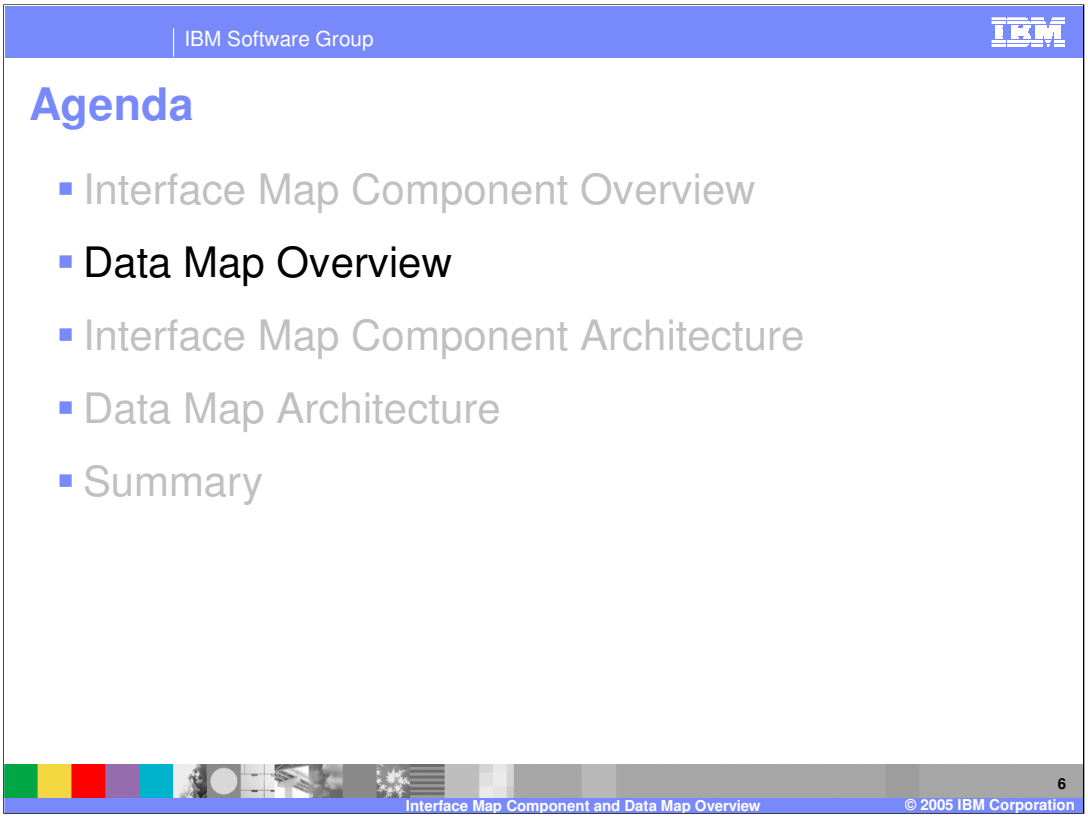
Interface Map Component

- Requirement to integrate disparate interfaces through:
  - Exports and Imports
  - Interface Map Component
- The interface map component performs the following services:
  - Mediates all source operations to target interface operations
  - Utilizes data mapping for parameter-to-parameter transformations
  - Provides the context information for relationship maintenance



© 2005 IBM Corporation

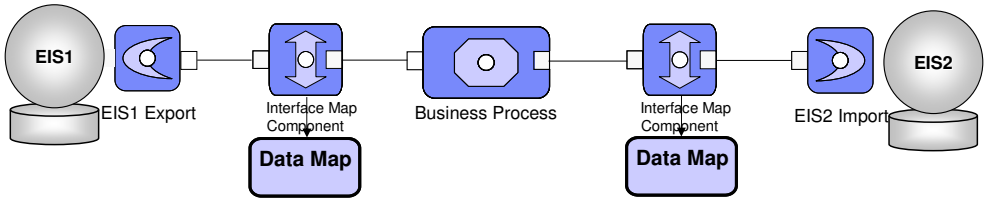
Obviously, there is a need to integrate these interfaces and there are multiple options for doing so. Options include using SCA imports and exports as well as interface map components and data maps. In the case of integrating external services or other SCA components, you can use exports and imports. In the case of integrating between two components, you can use interface mapping, also known as the interface map component. Interface maps mediate all source operations from the source component interface to the target operations of the target component interface. Once a source operation is bound to a target operation, you can utilize data mapping for parameter-to-parameter transformations. Within an operation, there are parameters that must be transformed to the compatible parameter type on the target operation. Interface maps also provide context information to the relationship service so that it can maintain and manage relationship instances. The diagram shown here illustrates that the map component fits in to the integration scenario between the two components having disparate interfaces and resolves the mismatch.



This section will provide an overview of the data map.

Data Map Overview

- Responsible for transforming business objects from one type to another as they flow through the WebSphere Process Server system
- Defines the transformation rules between the source and destination data
- Enables the Canonical Object Model pattern, i.e. enables processes to drive logic in an application-independent way
- Provides an integration environment that allows for the use of the “best of breed” applications available

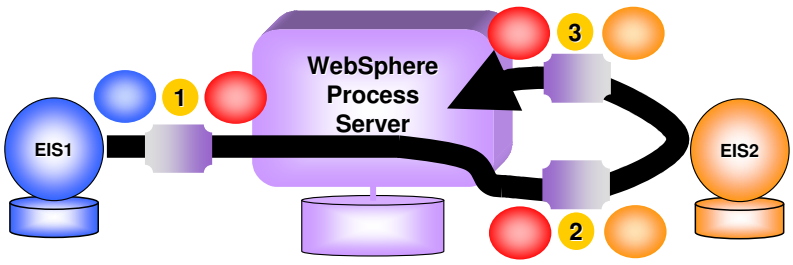


© 2005 IBM Corporation

One of the means of transforming data passed between business entities across your enterprise solution is data mapping. Mapping is responsible for transforming business objects from one type to another so that the business object is compatible with the expected business object type defined by the target operation. Data values from one or more attributes in a source business object are mapped to one or more attributes in a destination business object. Data mapping also defines transformation rules for how to transform data from the source to the destination format. Mapping is the enabler of the canonical object model pattern and provides the ability to design business logic in an application-independent manner, allowing you to preserve the core of your business integration logic independent of the actual enterprise integration systems or external services being integrated. It also provides the integration environment that allows for the ability to utilize the “best of breed” applications approach by enabling you to plug in various applications, while maintaining application independence. The diagram shown here illustrates that the data map fits in to the integration scenario by being invoked through the interface map component. Maps are one of the possible transformation mechanisms used by the interface map component. All four mechanisms will be discussed in this presentation.

Typical Mapping Flow

- A typical flow involves three mapping transformations:
  - Source application to the WebSphere Process Server
  - WebSphere Process Server to the destination application
  - Destination application back to the WebSphere Process Server

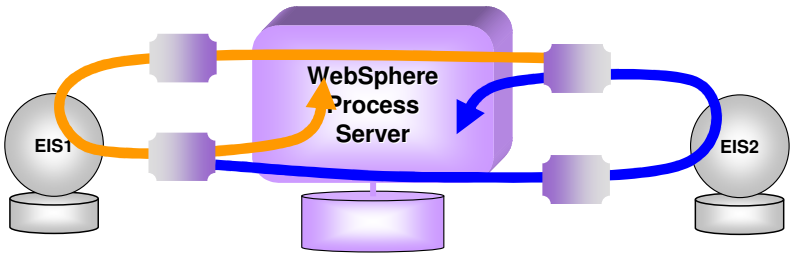


© 2005 IBM Corporation

This graphic illustrates a typical mapping flow for integrating two disparate enterprise information systems. A typical flow involves at least three mapping transformations. The first mapping transformation takes place between the source application and the WebSphere Process Server where the application specific data entity is transformed to a generic format with which the WebSphere Process Server integration application components can work. The second transformation takes place between the integration application components and the external application on the target side where the generic representation of the business entity is transformed to the application specific format compatible with the target application. The third transformation takes place on the response part of the scenario where the application specific entity is sent back to the integration application and transformed to generic format for subsequent processing or for establishing relationship cross references. A fourth transformation, or map, would be needed if the result of the business integration application processing was expected to be returned to the original source application.

Synchronizing Two Systems

- Two systems synchronizing each other use (at least) four map transformations
- Each business integration flow uses three maps
- Map used for Event Publishing can be reused in Request / Response interaction in another business integration flow



© 2005 IBM Corporation

Another typical scenario you might encounter involves a scenario where two enterprise information systems need to keep their data synchronized. This could involve two applications with two flows implemented. Each application initiates a flow in order to synchronize business entities across the WebSphere Process Server in order to reflect those business entities in the target system. This scenario would require at least four mapped transformations in order to transform the application specific business objects to their generic representation and back. If a specific transformation is needed for a specific flow, for example if EIS1 in the diagram requires one map for publishing an event and another for sending the response back to the process server, you could have two different maps and utilize a specific map for a specific event flow. Maps can be reused, but do not have to be. Maps are unidirectional, that is they cannot be used in a “reverse” or bidirectional manner.

Mapping Requirements Gathering

- To design a map, you need to know:
  - What data is required by the destination application
  - Where the source application stores this data
  - What transformations are necessary to migrate the data from source to destination
- To build a map, you need to:
  - Identify the necessary business objects
  - Determine if submaps are needed
  - Outline complex data transformations
  - Track the flow of data from the source application to the destination application

© 2005 IBM Corporation

Map design is typically complex as it is rare for one person to know two or more enterprise information systems well. The same could apply to mapping between various components within the WebSphere Process Server system. For example, what is described in one application as “Customer Status” may mean something else in another application or component. The following are some thoughts to keep in mind when preparing to work with maps. The first step in building a map is to design it. You will need to know what data is required by the target component or application. Next, you will need to know where the source application or component stores the data that will be transformed and what transformations must be performed in order to migrate the data from the source format to the target format. Once all these requirements are gathered, the necessary business objects and business graphs must be identified, the use of submaps determining granularity must be decided, and the flow of data from the source to the destination must be tracked across the WebSphere Process Server.

Agenda

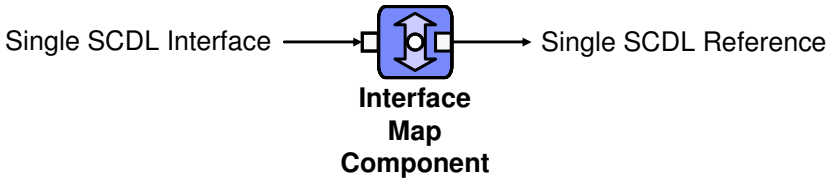
- Interface Map Component Overview
- Data Map Overview
- Interface Map Component Architecture
- Data Map Architecture
- Summary

© 2005 IBM Corporation

This section will discuss the interface map component architecture.

Interface Map Component Architecture

- Interface map component:
  - Resolves and reconciles differences between two interfaces
  - Has one source interface and one target interface (reference)
  - Performs the transformation of the operations and parameters
  - Invokes the destination/target component



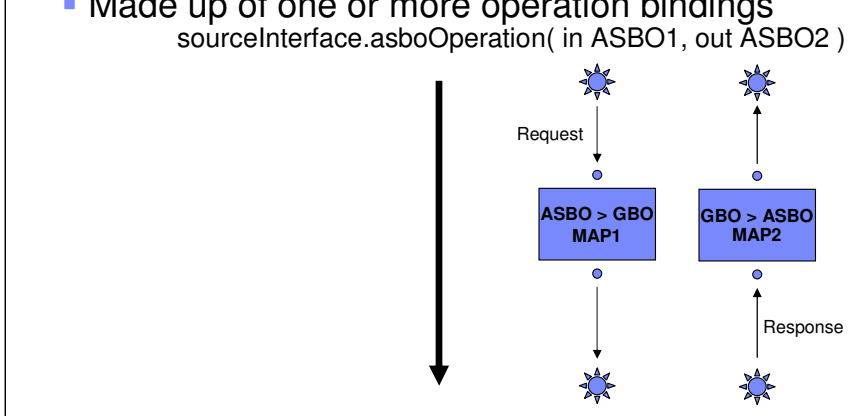
- Map component resolves interface disparities between:
  - Components in the same module
  - Components in different modules
  - Components and external services

© 2005 IBM Corporation

The interface map component resolves differences between two interfaces. It has one source interface and one target interface, which is also called a reference. Source operations and parameters are transformed to target operations and parameters, and ultimately, the destination component will be invoked. As previously stated, the map component resolves interface disparities between components in the same module, components in different modules, and components and external services.

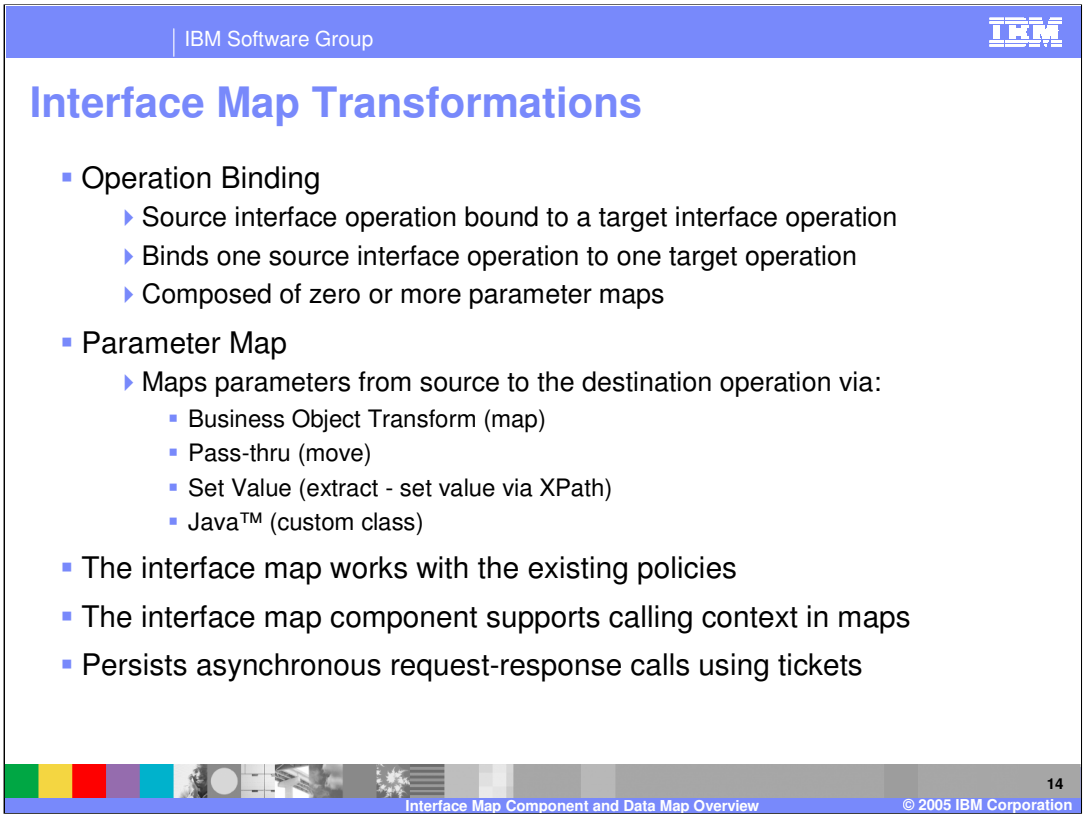
Interface Map Transformations

- Reconcile the differences across component interfaces by:
  - Operation Binding
  - Parameter Map
- Made up of one or more operation bindings



© 2005 IBM Corporation

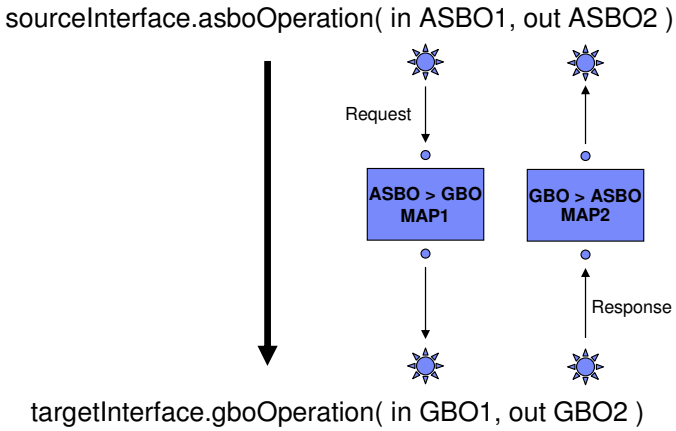
This graphic illustrates different interface map transformations. The way in which the interface map component resolves interface disparities is through operation binding and parameter mapping. Each interface map consists of one or more source operations, which are bound to their counterpart operations on the target side. In addition, an operation might contain input and output parameters. These source operation input and output parameters might need to be mediated to the parameters on the target operation. The graphic illustrates a simple scenario that might be encountered.



Operation binding performs the transformation of one source interface operation to one target interface operation and can be composed of zero or more parameter maps. Once two operations are bound, and if the source operation has parameters that must be transformed to target operation parameters, there are four map transformations available: business object transform, also known as a data map; pass-thru transformation, in which the source parameter is copied to the target parameter, also known as a move; set value transformation, in which a specific value is extracted via XPath expression from the source parameter and passed to the target parameter, also known as an extract; and Java map, in which any possible Java coding can be used, also known as a custom map. Interface maps work with existing policies and do not define any special policies of their own. Interface maps support passing the calling context required for the relationship service. Interface maps also persist asynchronous request-response calls using the SCA asynchronous programming model ticket mechanism.

Parameter Maps – Business Object Transform

- Incompatible input and output argument types in the source and target interface operations
- Transformed via maps

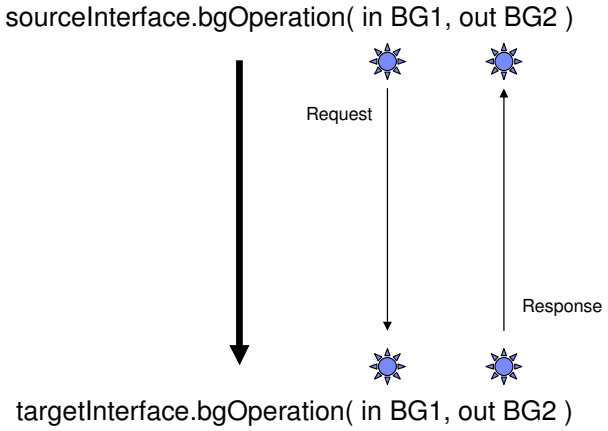


© 2005 IBM Corporation

The four types of parameter map transformations will now be examined more closely. The business object transformation is used when there are incompatible input and output argument types. A business object data map is used to transform the source parameter to the target parameter and can be used in either the request or the response direction or both.

Parameter Maps – Pass-thru

- Compatible input and output argument types in the source and target interface operations
- Passed directly from one operation to another

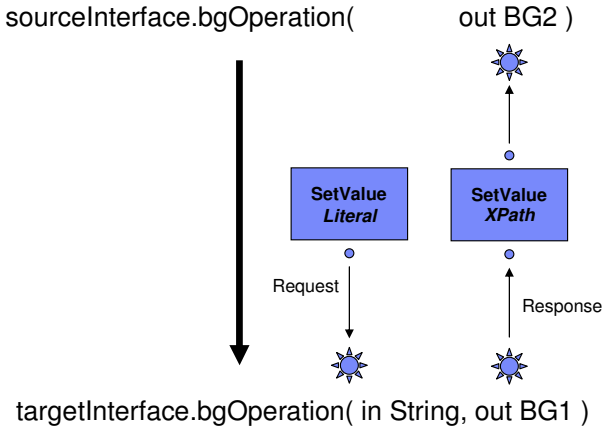


© 2005 IBM Corporation

The pass-thru transformation is used when the input and output argument types on the source and target are compatible and there is no need to perform any transformation of the parameter. In this case, the parameters can be passed directly to the target operation.

Parameter Maps – Extract/Set Value

- Incompatible input and output argument types in the source and target interface operations
- Source input argument may not exist

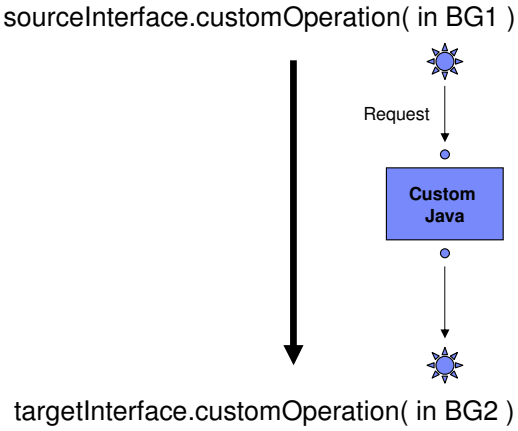


© 2005 IBM Corporation

The extract or set value transformation is used when there are incompatible input and output argument types, and when you want to set a literal value to pass to the target or extract a value from a complex data type and pass it to the target parameter. In this example, the source interface operation defines no input parameters while the target operation defines and expects a String input parameter. In this case, you could use a set value transformation. The example also shows the target operation responds with the BG1 output parameter which is routed to the XPath process to extract the wanted BG2 output value. The output parameter is populated and passed back to the source interface operation as the BG2 output parameter.

Parameter Maps - Java

- Incompatible input and output argument types in the source and target interface operations
- Custom transformation necessary for the argument types



© 2005 IBM Corporation

The Java transformation is used when there are incompatible input and output argument types of the source and target operations. In this case, none of the previous transformation options discussed meets your requirements, so you write your own Java transformation utility class to perform the wanted transformation and pass the output to the target parameter.

Agenda

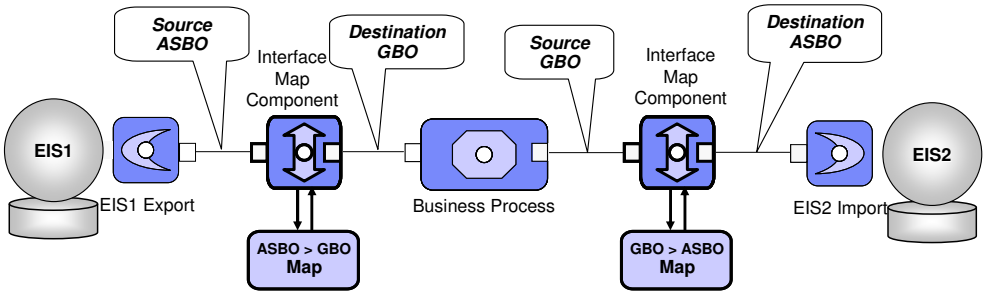
- Interface Map Overview
- Data Map Overview
- Interface Map Architecture
- Data Map Architecture
- Summary

© 2005 IBM Corporation

This section will cover the data map architecture.

Data Mapping Architecture

- Mapping provides support for Business Objects and Graphs
- Invoked by any component that requires BO transformation
- Mapping provides support for the following capabilities:
  - Transforming the Change Summary / Event Summary
  - Utilizing the Relationship Service



© 2005 IBM Corporation

Mapping provides support for business objects and business graphs and can be invoked by any component that requires business object transformation. Besides transforming the attributes of the business object, mapping is also capable of transforming the change summary and event summary and utilizing the relationship service. As shown in the graphic, maps are one of the transformations an interface map component can utilize in order to mediate the source operation parameters to the target operation parameters. Maps are defined with transformation rules, one of which is a relationship transformation rule used to invoke the relationship service. These transformation rules will be discussed further in following slides.

Mapping Terminology

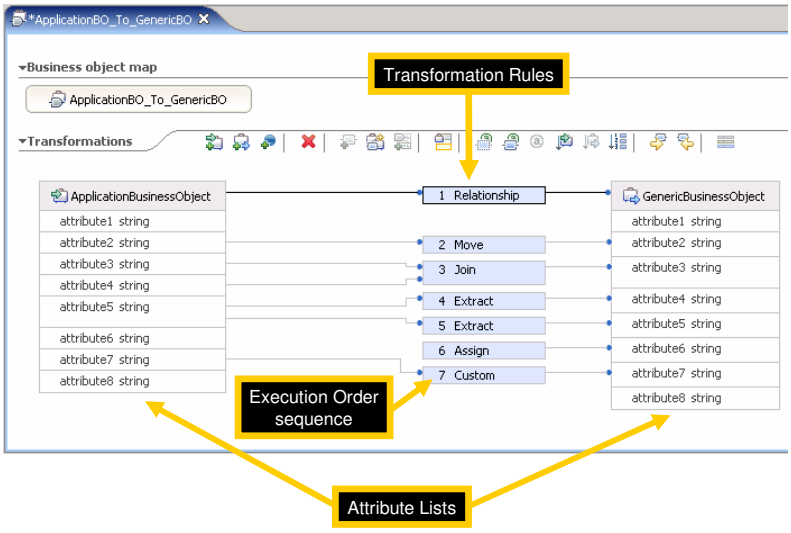
- *Map definition* is a template that specifies the transformation rules from one Business Object to another Business Object
- *Transformation rule* defines a method of transferring information from the source to the destination objects/graphs
- Map definition supports the use of other maps as transformation rules (*submaps*)
- Map definition provides the following information:
  - Name
  - Target namespace
  - Source and destination Business Objects (or Business Graphs)
  - Temporary variables
  - Transformation rules
- *Map instances* are created by Mapping Service at runtime
- *Calling context* is received by maps from the interface map components and passed to the relationships

© 2005 IBM Corporation

Shown here is a list of typical mapping terminology. The map definition is a template that specifies the transformation rules from one business object to another business object. The transformation rule defines a method of transferring information from one source attribute to a destination attribute or attributes. The map definition supports the use of other maps, known as submaps, which perform transformation from one business object to another. If you have a map that has a business object with a complex type attribute, you could use another map, or submap, to perform the transformation of that complex type attribute. The map definition provides basic information such as the name, target namespace, source and destination business objects or graphs, temporary variables, and transformation rules that define how individual attributes of those business objects are being transformed. Map instances are the actual map instantiated at runtime by the mapping service. The calling context is received by the map from the interface map component and passed to the relationship service.

Transformation Rules

- Transformation rule defines a method of transferring data:
  - **Move**
  - **Join**
  - **Extract**
  - **Assign**
  - **Custom**
  - **Custom Callout**
  - **Custom Assign**
  - **Relationship**
  - Submap

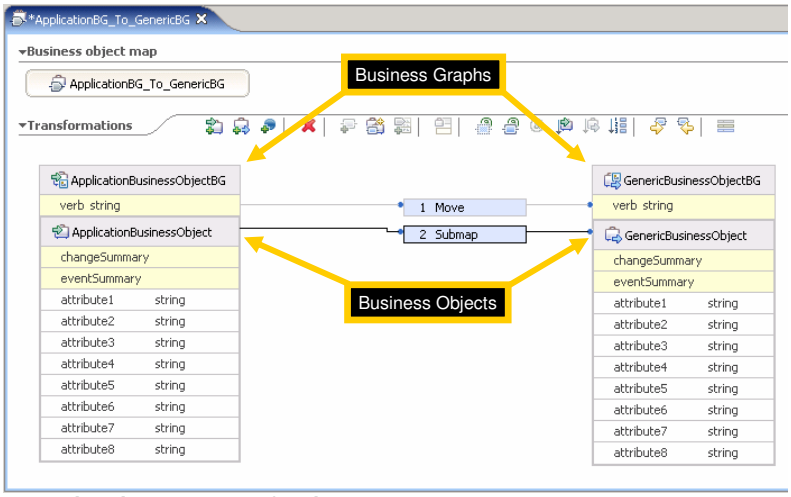


© 2005 IBM Corporation

A list of currently supported and predefined transformation rules is shown here. This includes move, join, extract, assign, custom, custom callout, custom assign, relationship, and submap. When you create a new map, you define the source and target business objects or business graphs and then the transformation rules for the business objects or the attributes within them. The execution order sequence is specified as part of defining the transformation rules.

Transformation Rules - Submap

- Transformation rule defines a method of transferring data:
  - Move
  - Join
  - Extract
  - Assign
  - Custom
  - Custom Callout
  - Custom Assign
  - Relationship
  - **Submap**



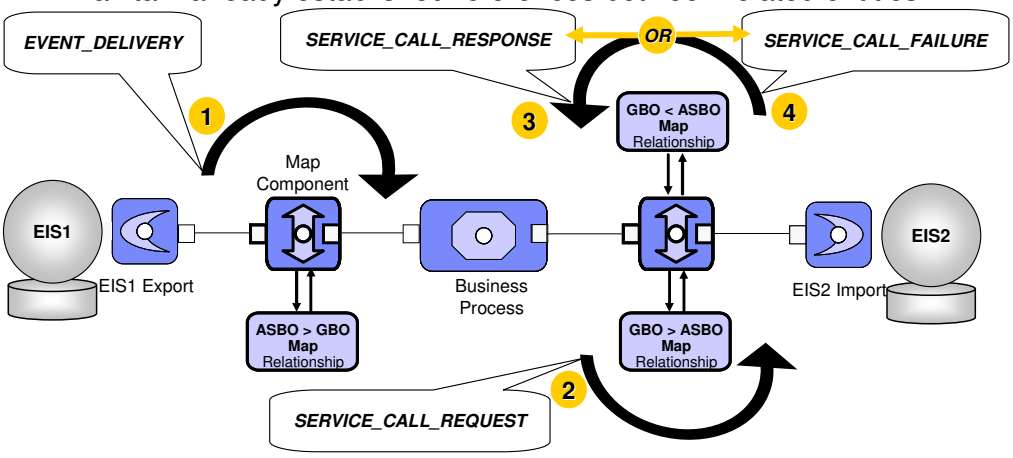
- Several ways to design maps/submaps

© 2005 IBM Corporation

A submap is a special transformation rule that gives you the ability to call another map. In this graphic, you can see a map defined to transform an application specific business graph to a generic business graph. Inside the business graph is a business object for which you already have an existing map so you can use a transformation rule of submap that will call the existing map for the business objects rather than needing to define the transformation rules again.

Maps – Calling Context

- Calling context is passed to the Mapping Service
- Mapping passes the context to the Relationship Service to:
  - Establish references across the integration environment
  - Maintain already established references between related entities



© 2005 IBM Corporation

This graphic shows the four different calling contexts that the interface map component passes through the mapping service to the relationship service. The relationship service provides the ability to establish cross references across the integration environment. The calling context gives the relationship service the information it needs to determine what phase the flow is in in order to know what to do with the underlying relationship cross reference tables. In some stages, it may need to write, lookup, or delete data. There are four possible calling contexts: event delivery, indicates that the event is being sent into the business integration environment; Service Call Request, indicates a request/response type of call; Service Call Response or Service Call Failure, indicates either successful completion of the service call request or a failure at the target system.

Agenda

- Interface Map Overview
- Data Map Overview
- Interface Map Architecture
- Data Map Architecture
- Summary

© 2005 IBM Corporation

This section will provide a summary of this presentation.

Summary

- The Interface Map Component definition:
  - Has one source interface and one target interface (reference)
  - Binds source interface operations to target interface operations
  - Associates maps for parameter-to-parameter transformations
- The Interface Map Components:
  - Perform the transformation of the operations and parameters
  - Provide the context for relationship maintenance (via maps)
- The Map definition:
  - Defines the transformation rules between parameter attributes
  - Supports the use of other maps as transformation rules (submaps)
  - Supports multiple input output business objects/graphs
- The Maps:
  - Transform business object parameters from one type to another
  - Pass the calling context from the map to the relationships

© 2005 IBM Corporation

An interface map component definition always has one source and one target interface or reference, and it binds source interface operations to target interface operations. The interface map component definition associates maps for parameter-to-parameter mapping transformations. The interface map component performs the transformation and provides the calling context, via maps, for the relationship service maintenance. Map definitions define the transformation rules between parameter attributes, including the predefined ones such as move, join, extract, custom, and assign. The map transforms the business object parameters from one type to another and passes the calling context from the map to the relationship service.

Trademarks, Copyrights, and Disclaimers

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

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

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

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

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

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

Linux is a registered trademark of Linus Torvalds.

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

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

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

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

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

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

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

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