



# Using WebSphere Enterprise Service Bus for z/OS



# Trademarks

The following are trademarks of the International Business Machines Corporation in the United States, other countries, or both. For a complete list of IBM trademarks please visit [www.ibm.com/legal/copytrade.shtml](http://www.ibm.com/legal/copytrade.shtml)

CICS	IBM Logo	S/390
DB2	IMS	Tivoli
E-business logo	iSeries	VM/ESA
ESCON	MVS	VSE/ESA
eServer	OS/390	WebSphere
FICON	pSeries	z/OS
IBM	Rational	zSeries
	RS/6000	System z

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 trademarks of Microsoft Corporation in the United States, other countries, or both.

Microsoft trademark guidelines

Intel is a registered trademark of Intel Corporation in the United States and other countries.

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

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

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



## ***Agenda***

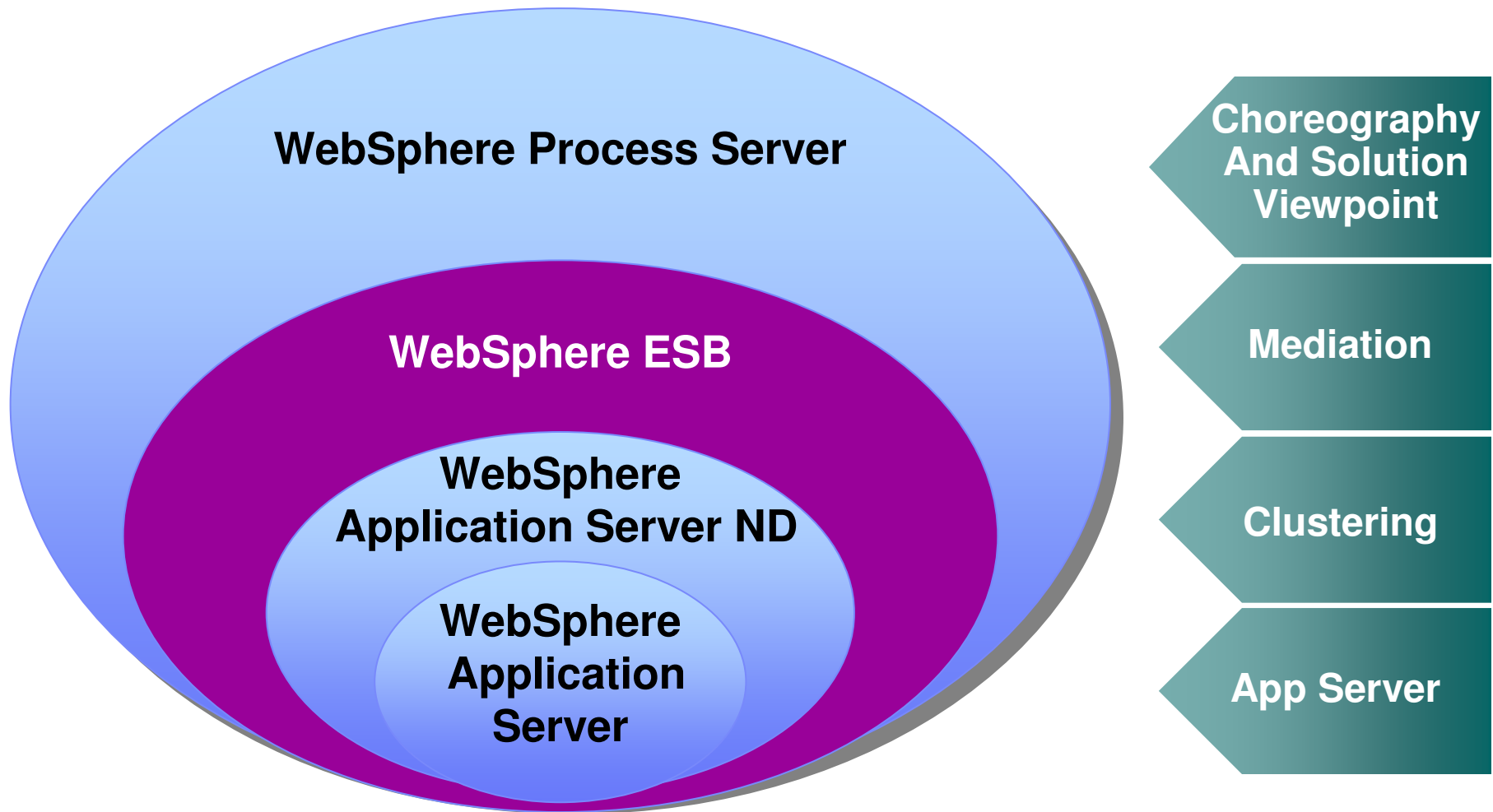
- What is WebSphere Enterprise Service Bus
- What's new with V6.1
- A System z customer scenario



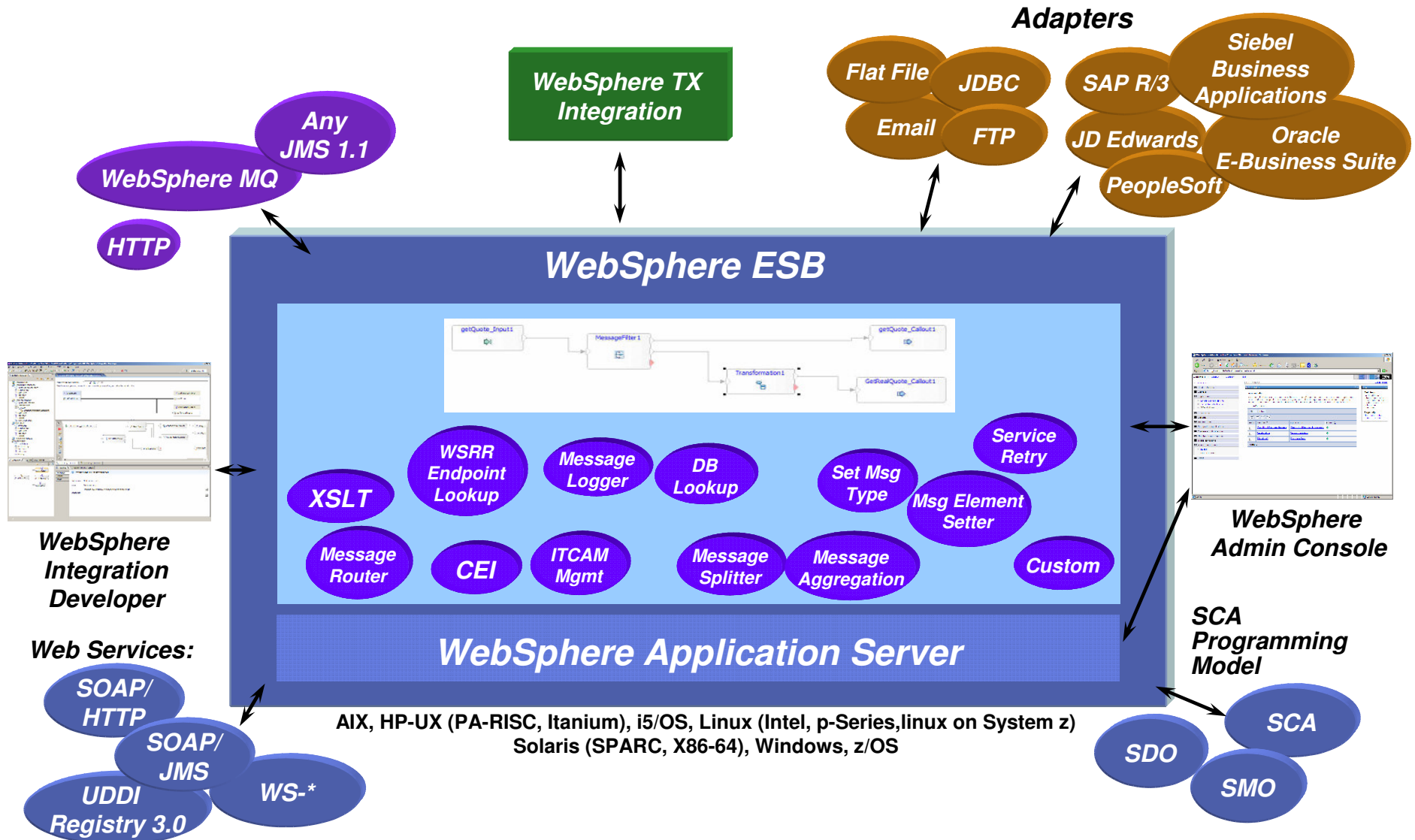
SOA: Unlock business value.  
→ New software and services.



## *WebSphere Application Server, ESB, and Process Server*



# WebSphere ESB



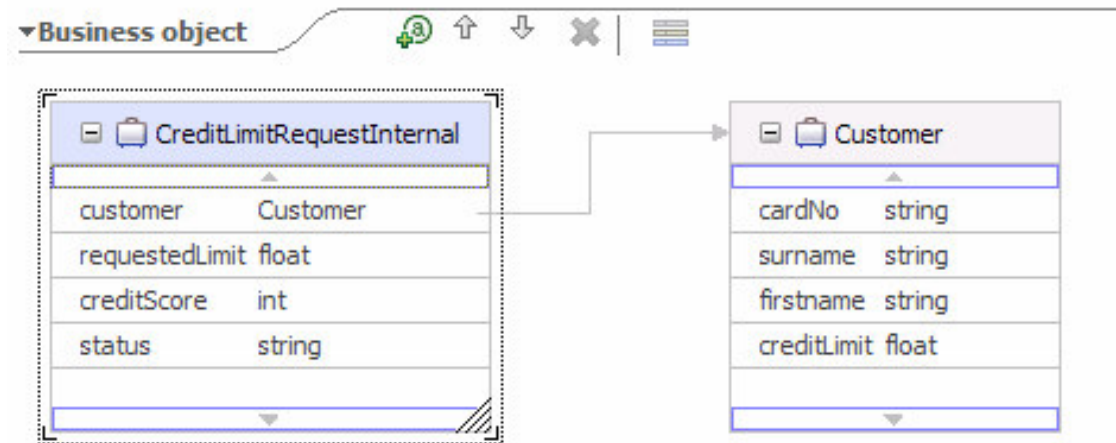
## Service Component Architecture (SCA)

- Component-based programming model for constructing distributed applications
  - ▶ Components run in SCA container
  - ▶ Component interfaces described by WSDL
- Component invocations via API or by wiring in SCA assembly
- Application data represented by business objects (SDO)
- Many possible component implementation types
  - ▶ ESB focus is on Mediation Flow Component



# Common Data Model: Business Objects

**Business object  
definition  
(SDO/XML schema)**

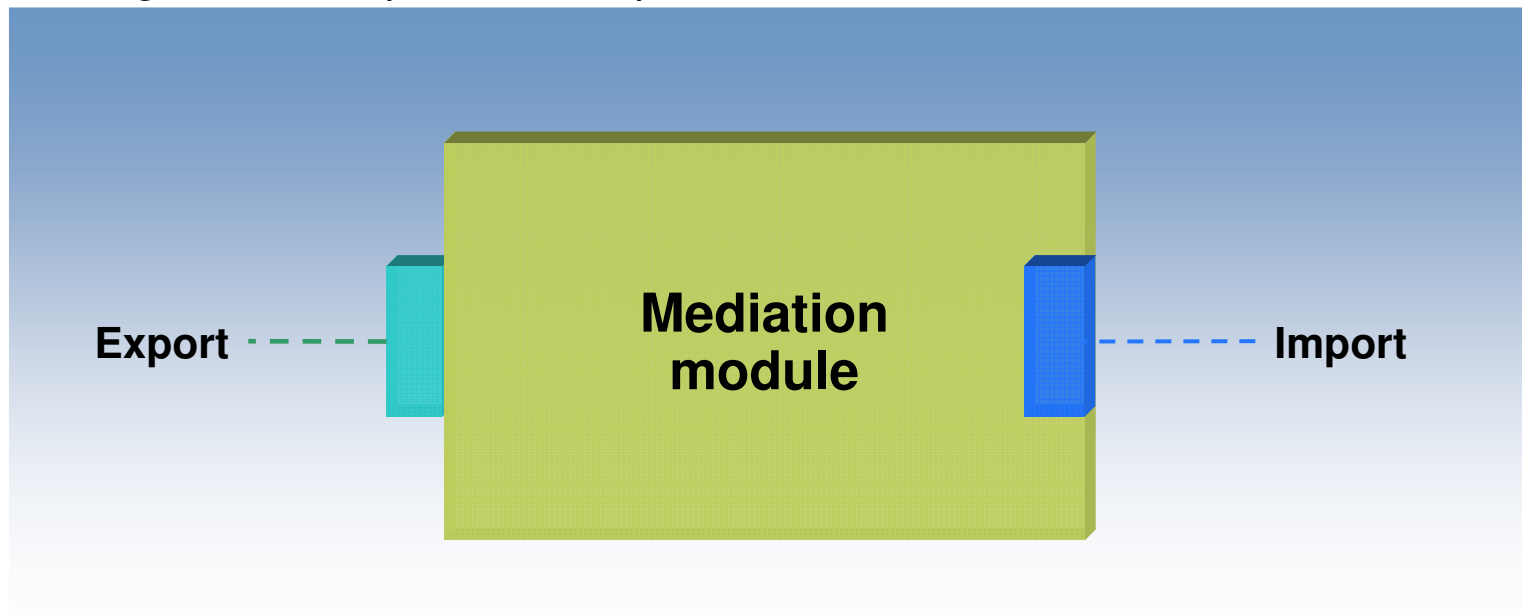


- Business Objects represent structured application data
- Consistent logical representation, independent of data source or wire format
- Based upon SDO standard
- Multiple data bindings can be supported



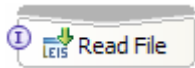
## Concepts: Mediation Module

- **Interactions with external service requesters and providers defined by imports and exports**
  - ▶ Interfaces are defined using the Web Services Description Language (WSDL)
    - Which may contain several service *operations*
  - ▶ Different kinds of requester and provider are made available via different *bindings* for the imports and exports

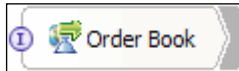
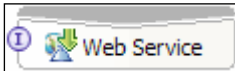




# Export/Import Bindings



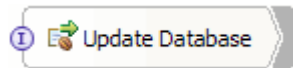
**WebSphere Adapters (e.g. CICS, IMS, SAP...)**



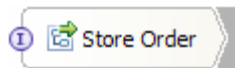
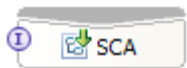
**Web services (SOAP/HTTP, SOAP/JMS)**



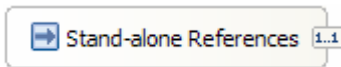
**JMS / WebSphere MQ JMS / WebSphere MQ**



**EJB**



**Native (SCA)**



**Java invocation**



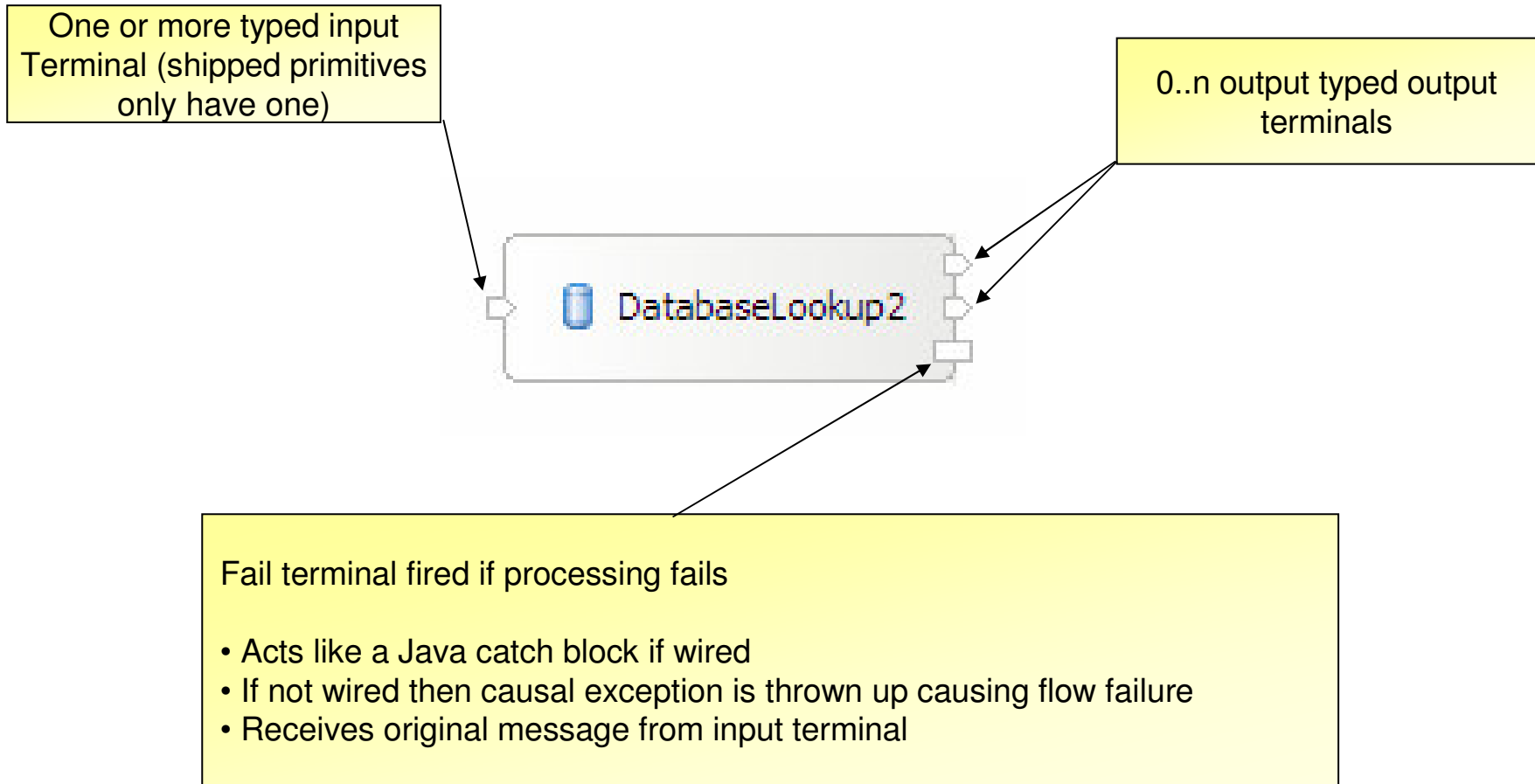
# Concepts: Integrating request-response interactions using WebSphere ESB Mediation Flow Components

- **Processing is performed by one or more mediation *flows*, comprising instances of mediation *primitives***
- **Processing of requests is separated from processing of responses**
- **WebSphere ESB Mediation Flow Components can act as ‘service intermediaries’**
  - ▶ Allowing the mediation flow component to
    - pass a (potentially modified) request from a service requester to a service provider
    - pass a (potentially modified) response from a service provider to a service requester
- **Request processing within a mediation flow component can send a response back to the requester without necessarily needing to contact a service provider**

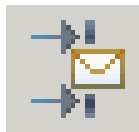




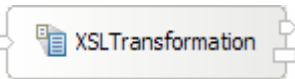
# Anatomy of a Mediation Primitive



# WebSphere ESB v6.0.1 Mediation Primitives



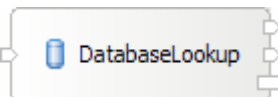
MessageFilter – routes within a flow



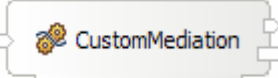
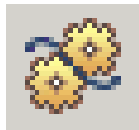
XSLTransformation – transforms content



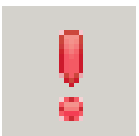
MessageLogger – logs to database



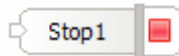
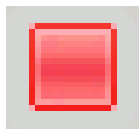
DatabaseLookup – adds content from database



CustomMediation – executes custom logic (Java)



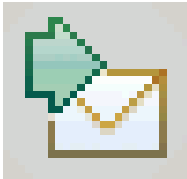
Fail – terminates flow with an exception



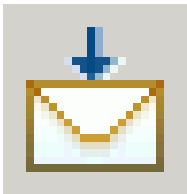
Stop – terminates flow normally



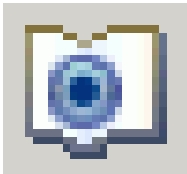
## WebSphere ESB v6.0.2 Additional Primitives



**EventEmitter** - emits a common base event at a point significant in the mediation flow.



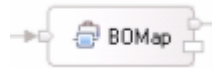
**Message Element Setter** - sets, copies, or deletes the content of message headers or bodies



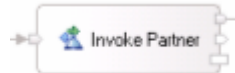
**Endpoint Lookup** - queries the WebSphere® Service Registry and Repository and retrieves service endpoints, which it places in the message context. The retrieved endpoints can then be used to dynamically invoke a service



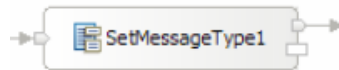
# WebSphere ESB 6.1 Additional Primitives



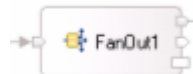
**Business Object Mapper Primitive:** performs structure-structure mapping, provides access to relationships function.



**Service Invoke:** allows a service to be invoked from within a request or response flow (not just at the end, as in 6.0.2)



**Set Message Type:** allows a weakly typed element (xs:any, xs:anyType, ...) to have a type asserted for it at runtime



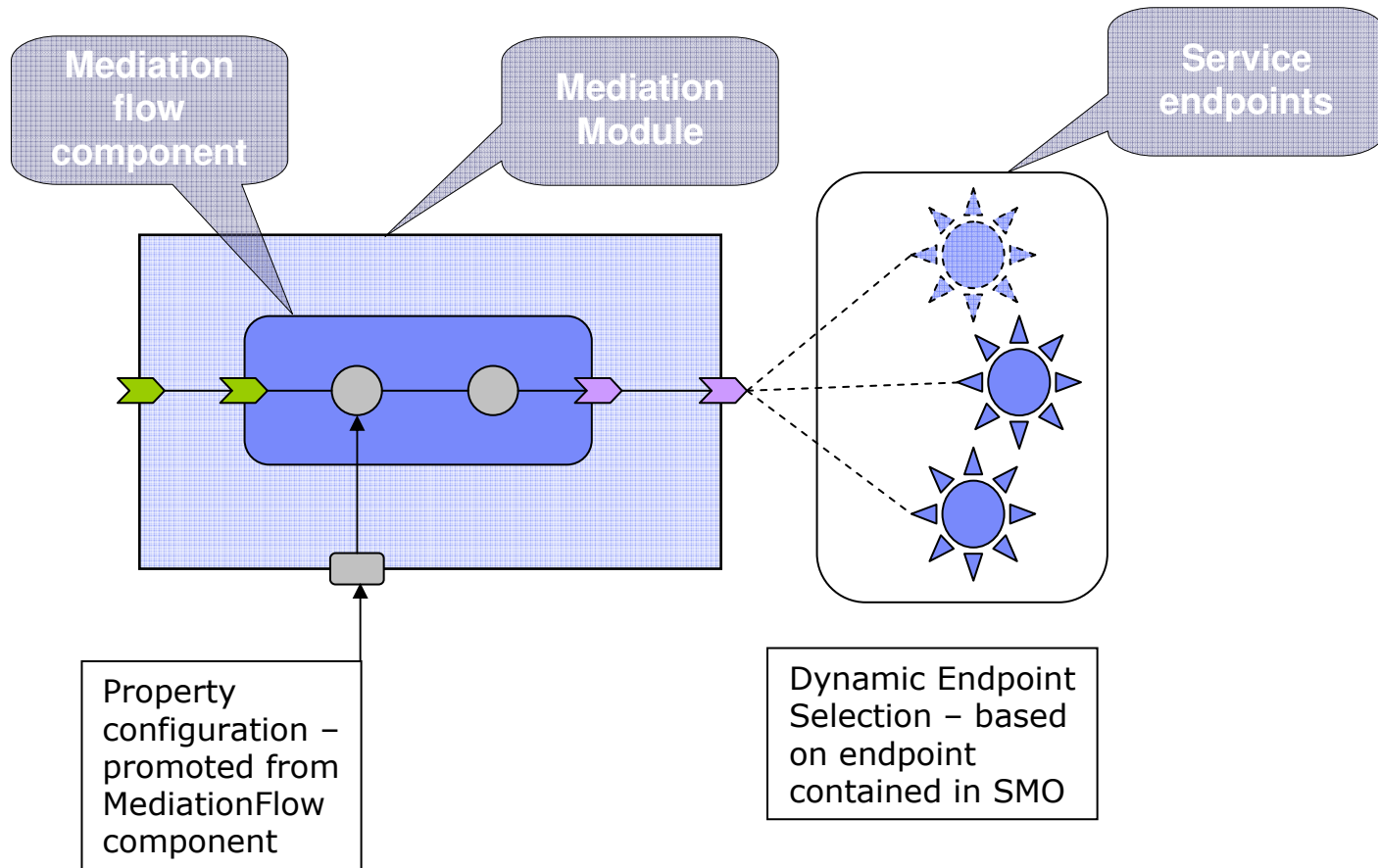
**Fanout:** allows a flow to be split - and to operate on repeating elements of the message, and may be paired with **Fanin**



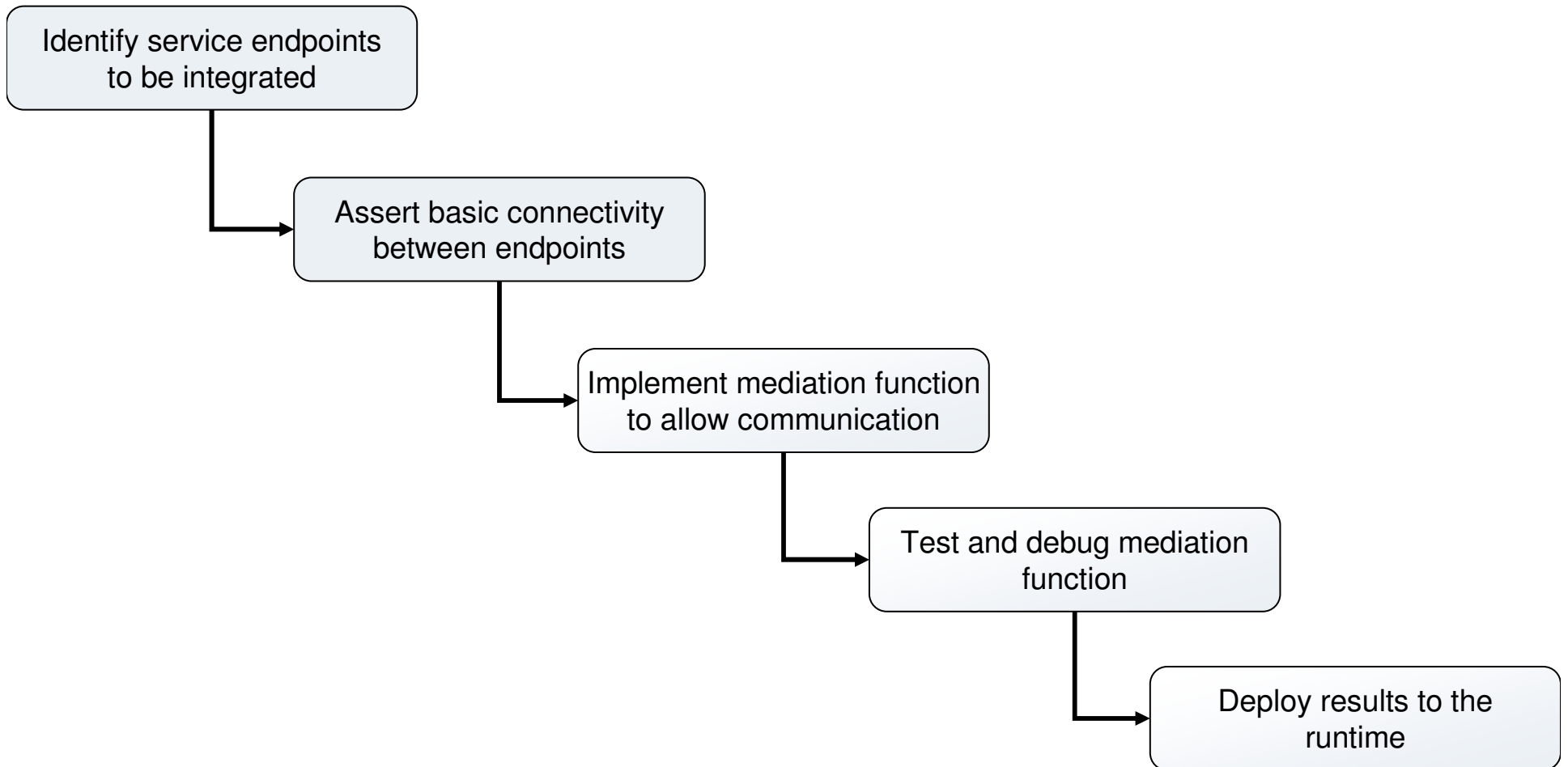
**Fanin:** allows a flow that has been split using Fanout to be joined - can be used to perform *aggregation*



# Dynamic Service Invocation



# Typical Integration Developer Task Flow

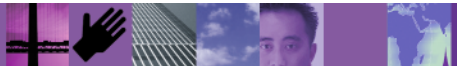
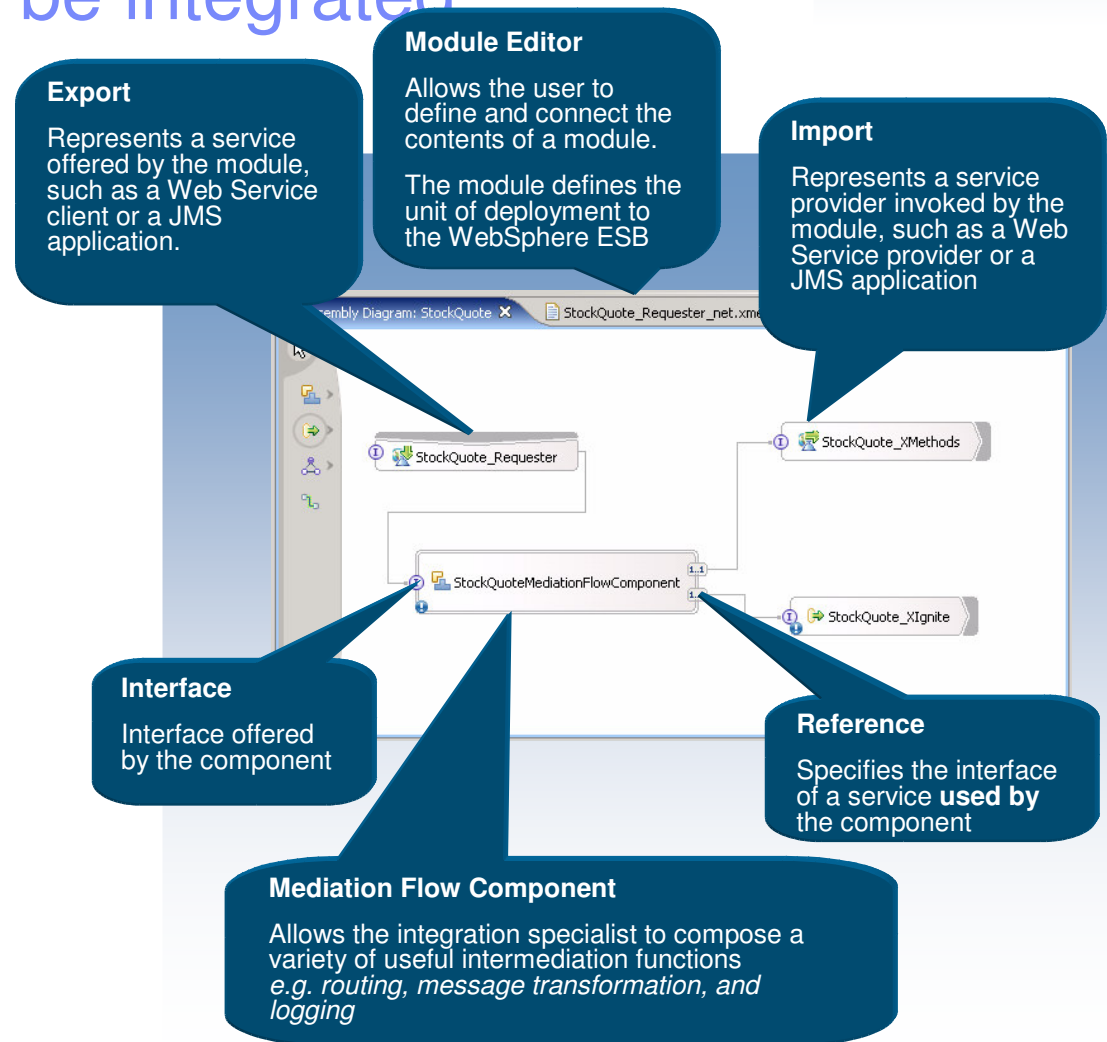






# 1. Integration developer specifies the service endpoints that need to be integrated

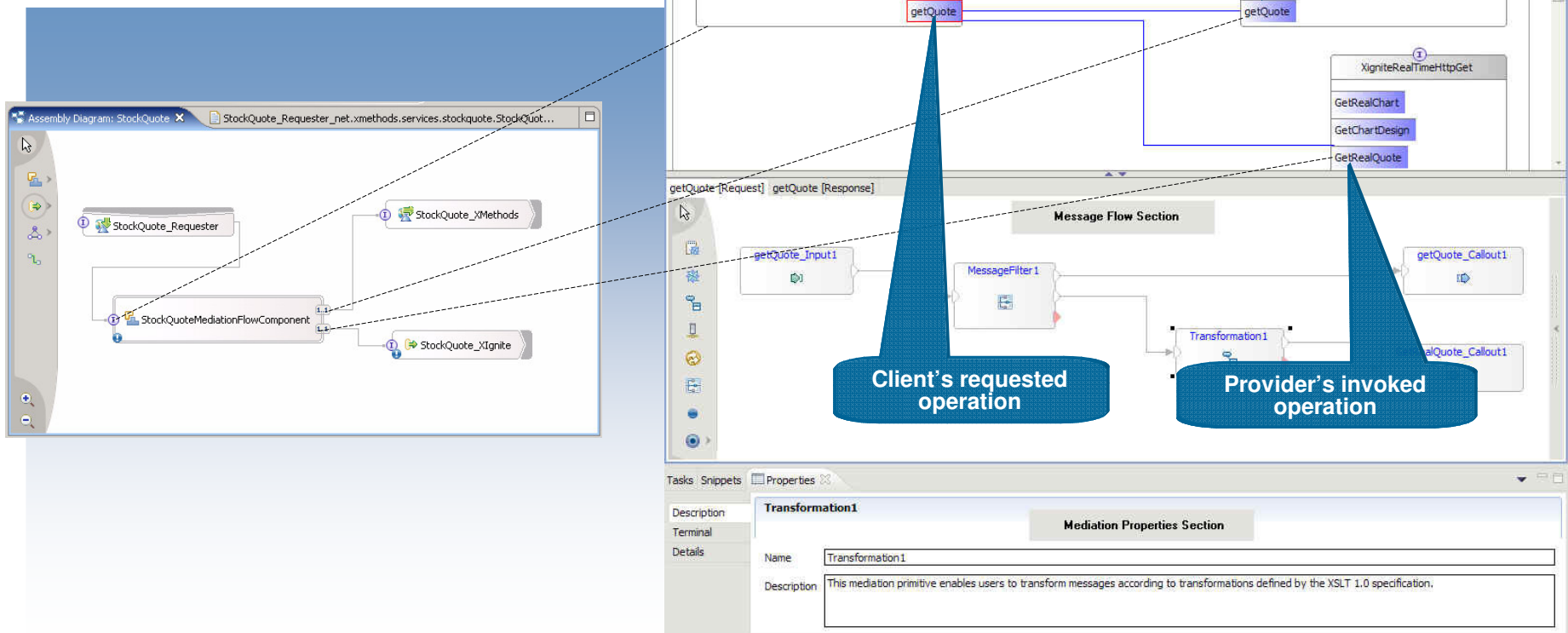
- **Uses the 'module editor' to construct a mediation module**
  - ▶ specifies how a subset of WebSphere ESB's service requesters and service providers interact
- **Within the module**
  - ▶ Service requesters are represented as 'exports'
  - ▶ Service providers are represented as 'imports'
  - ▶ The integration (mediation) function is represented as a 'mediation flow component'
  - ▶ Imports and exports are connected to the mediation flow component





## 2. Integration Developer asserts the basic connectivity between these endpoints

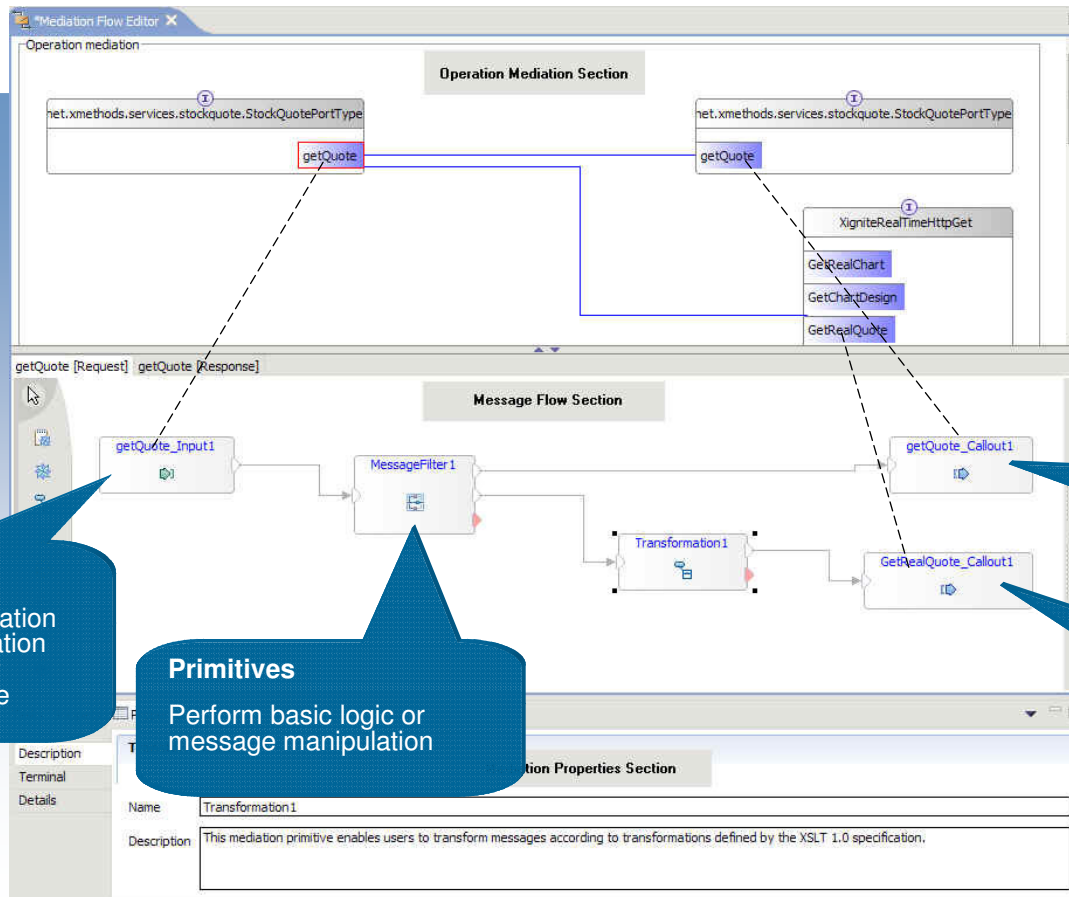
- The integration developer uses the *mediation tooling* to specify the essential connectivity between a requester and one or more service providers





### 3. Decides on the mediation function required to allow endpoints to communicate effectively

- The integration developer constructs a *mediation flow* for the service request by selecting and connecting *mediation primitives* from supplied function



**'Input'**  
Represents the invocation of the specified operation on the mediation flow component's interface

**Primitives**  
Perform basic logic or message manipulation

**'Callout'**  
Causes the specified operation to be invoked

**'Callout'**  
Causes the specified operation to be invoked



## 4. Integration developer customizes the elements of the mediation flow

- e.g. Customizes the XSLT transform mediation primitive by using mapping tool to construct an XSLT transform

The structure of the message is represented graphically

A properties view is provided where the details of the mapping can be specified

Define functions that apply to the mapping

Target	Source	Applied Function/Grouping
StockQuoteXIgnite_GetRealQuote...	StockQuoteXMethod_getQuoteRequest1	
tns:GetRealQuote		
Exchange		string
Symbol [0..1]	symbol	
IncludeBidAsk		boolean

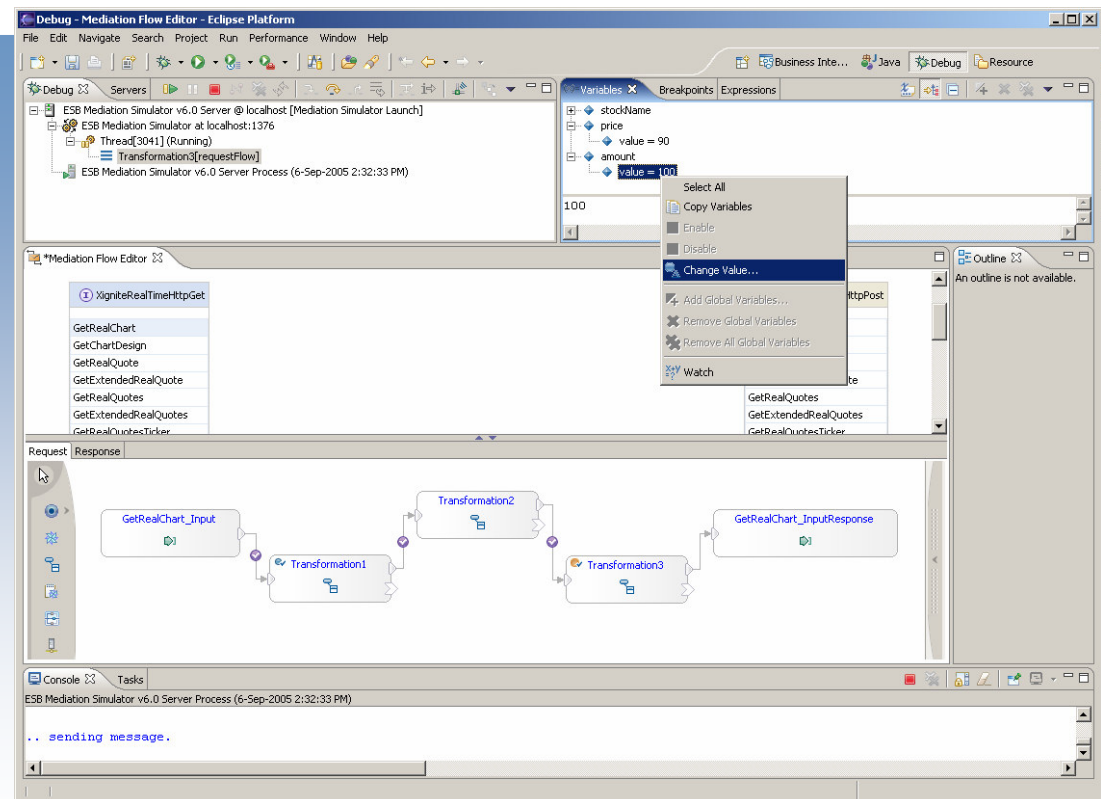




## 5. Debugs the composed/configured mediation function: WID Mediation Visual Debug

*Use the visual debugging tools to debug a solution*

- Debug mediation flows using an in-place visual debugger
- Breakpoints can be added, step into, through, or over areas of interest while inspecting the values of the messages



# Test and Debug – Integration Test Client

**Launch Integration Test Client**

**Select Module, Operation**

Configuration: Default Module Test

Module: Main\_Module

Component: TranslateProcess

Interface: TranslateProcess

Operation: startProcess

Initial request parameters

Name	Type	Value
input1	Translate_IN	
name	string	Paul Pacholski
message	string	hello
language	string	german

**Examine, Event Trace & Output**

Module: Main\_Module

Component: TranslateProcess

Interface: TranslateProcess

Operation: startProcess

Return parameters:

Name	Type	Value
result	TranslateOUT	
message	String	Hallo Paul Pacholski!

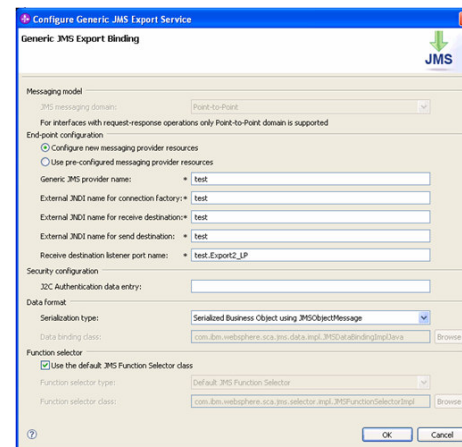
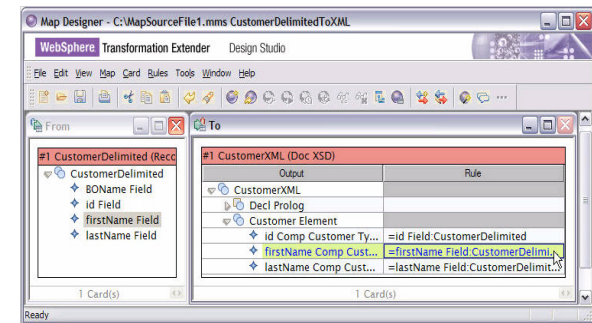
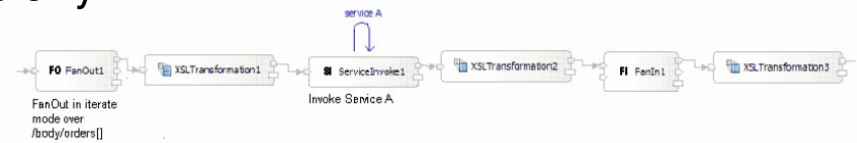
**Enter Input Data & Launch Operation**

Continue



## What's New in WebSphere Enterprise Service Bus v6.1

- Supports a broader set of mediation patterns quickly and with reduced development effort
  - New message splitting and aggregation patterns
  - New service retry capability
  - Updated Business Object Mapper and relationship support
  
- Integrates a broader range of services with expanded connectivity
  - New WebSphere TX integration
  - New WS-Notification support
  - New generic HTTP support
  - Enhanced 3rd party JMS support
  
- Enables streamlined operational infrastructure with platform currency
  - Updated WAS 6.1 based runtime
  - New and expanded OS platforms, including 64-bit exploitation and i5/OS support
  
- Improves consumability and usability across the solution lifecycle
  - Easier server installation and configuration
  - Expanded XML and WSDL support



## ***WebSphere ESB 6.1.2 Available 1 Aug 2008***

- **Enhanced support for Web Services Description Language (WSDL) XML Schema Definition (XSD), enabling the use of many industry-standard XML schemas.**
- **Improved error messages, logging, and First Failure Data Capture (FFDC) usage. FFDC can reduce defect resolution times and the number of times IBM Support asks a client to recreate a problem with different trace settings turned on.**
- **Adds support for manipulating MQCIH message headers for WebSphere MQ, which are exploited by the CICS bridge to enable interaction with CICS applications.**
- **Adds new format support for the following additional message formats:**
  - Delimited and full support for Comma Separated Values (CSV) Fixed-width format, and JavaScript Object Notation (JSON)
  - Enhances support for COBOL Copybook, C Struct, and PLI





## ***WebSphere ESB for z/OS – Extends the value of WebSphere Application Server for z/OS***

- Clustering support for fault tolerance, scalability
- Java platform with zAAP offload
- Native application integration
  - WebSphere MQ for z/OS (including CICS)
  - WebSphere TX integration
  - CICS Transaction Gateway (JCA)
  - IMS Connect (JCA)
  - DB2 – DB2 Universal JDBC Provider, IBM Application Connectivity to DB2 for z/OS Feature
  - PDS files
  - WebSphere Classic Federation Server for z/OS integration
- IBM z/OS Security Server support
- ...



**Industry:** Education  
**URL:** <http://cms.bsu.edu/>

***“SOA has been such a gift to us. It enables us to embrace a new technology that provides services at a level that we couldn’t even imagine before.”***

*–Dr. O’Neal Smitherman*



BALL STATE  
UNIVERSITY



## *Ball State University*

*Ball State University bridges disparate systems and solves key administrative issue with IBM SOA solution.*

### CHALLENGE

- Coordinate 40 name and address systems to streamline administrative processes and ensure information integrity for users

### SOLUTION

- SOA with Enterprise Service Bus to connect siloed applications without hand-coding individual API calls (WESB, CICS TS, System z)

### BENEFITS

- Ability to develop and implement services in an SOA environment for resolving name and address discrepancies in 10 months, as opposed to several years for hand-coding individual application connections
- Confidence that IBM solution can lead to wider use of SOA to further streamline administrative business processes
- Services created here can be reused in later SOA efforts



Thank  
You

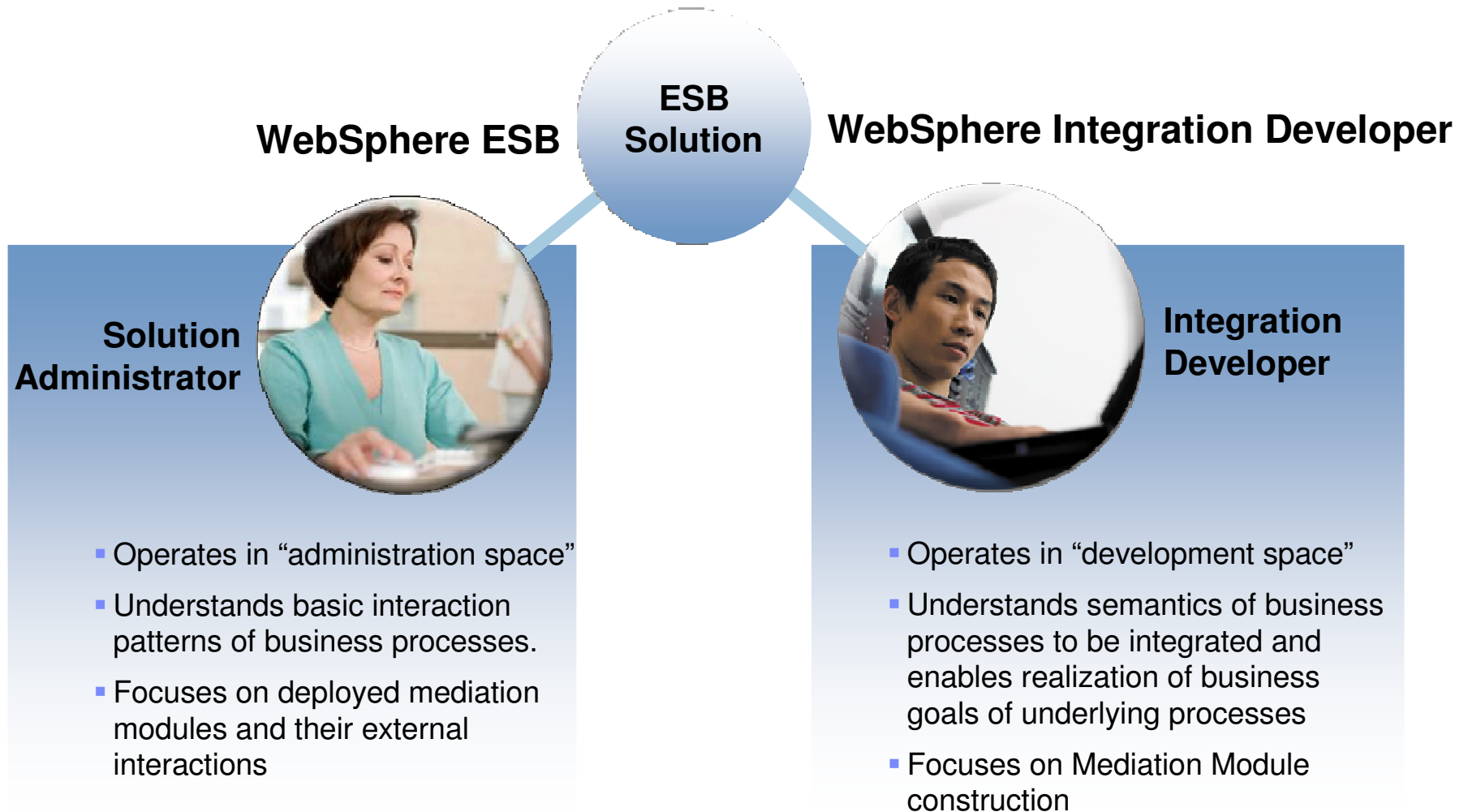




# Additional Information on WESB V6.1



# User Roles and Tasks





# *WebSphere ESB Core*

## Fundamental Services and Connectivity

**WebSphere** software



## HTTP Export / Import [New]

- HTTP 1.0 and 1.1
- SSL over HTTP
- Request/Response invocation
- Static header setting in WID and Admin
- Dynamic header setting via SMO in mediation modules
- Binary, XML and SOAP payloads
  - Plus custom data bindings
- Custom HTTP methods in Import
- Specifiable Content and Transfer encodings
- Endpoint based routing in Export

The screenshot displays the IBM WebSphere IDE interface. At the top, a context menu is open for the component 'CustomerOrderImport'. The 'Generate Binding...' option is selected, which has opened a sub-menu with 'HTTP Binding' as the first option. Below this, the 'Import: CustomerOrderImport (HTTP Binding)' configuration dialog is visible. The dialog has several fields:

- Endpoint URL: `http://temp.url`
- Data Binding: `com.ibm.ws.sca.databinding.impl.DataBindingImplXML`
- HTTP Version: `1.1`
- HTTP Method: `GET`
- Binding description: (empty text area)

The left sidebar of the IDE shows the 'Properties' view with a tree structure under 'Binding':

- Method bindings
- Advanced configuration
- Summary

## Generic JMS Export / Import Bindings [New]

- Simplify definition of JMS Export/Import binding
  - For 3<sup>rd</sup>-party JMS 1.1 ASF-compliant providers
    - Oracle AQ, TIBCO, SonicMQ, WebMethods, BEA WebLogic, etc.
  - Avoids Deployment Descriptor editing
  - Automatic setup of WAS Generic JMS resources
    - Manual setup of provider's JMS resources
- Admin visibility and modification of binding properties
  - Now available across all JMS and MQ bindings
- Dynamic JMS header updates via SMO
- Works with existing JMS Data Bindings

Configure Generic JMS Import Service

Generic JMS Import Binding

JMS

Messaging model

JMS messaging domain: Point-to-Point

For interfaces with request-response operations only Point-to-Point domain is supported

End-point configuration

Configure new messaging provider resources

Use pre-configured messaging provider resources

Generic JMS provider name: \*

External JNDI name for connection factory: \*

External JNDI name for send destination: \*

External JNDI name for receive destination: \*

Receive destination listener port name: test.Import3\_RESP\_LP

Security configuration

J2C Authentication data entry:

Data format

Serialization type: Serialized Business Object using JMSObjectMessage

Data binding class: com.ibm.websphere.sca.jms.data.impl.JMSDataBindingImplJava Browse

Function selector

Generate "TargetFunctionName" message header property for default JMS Function Selector

OK Cancel





## ***WTX Data Bindings [New]***

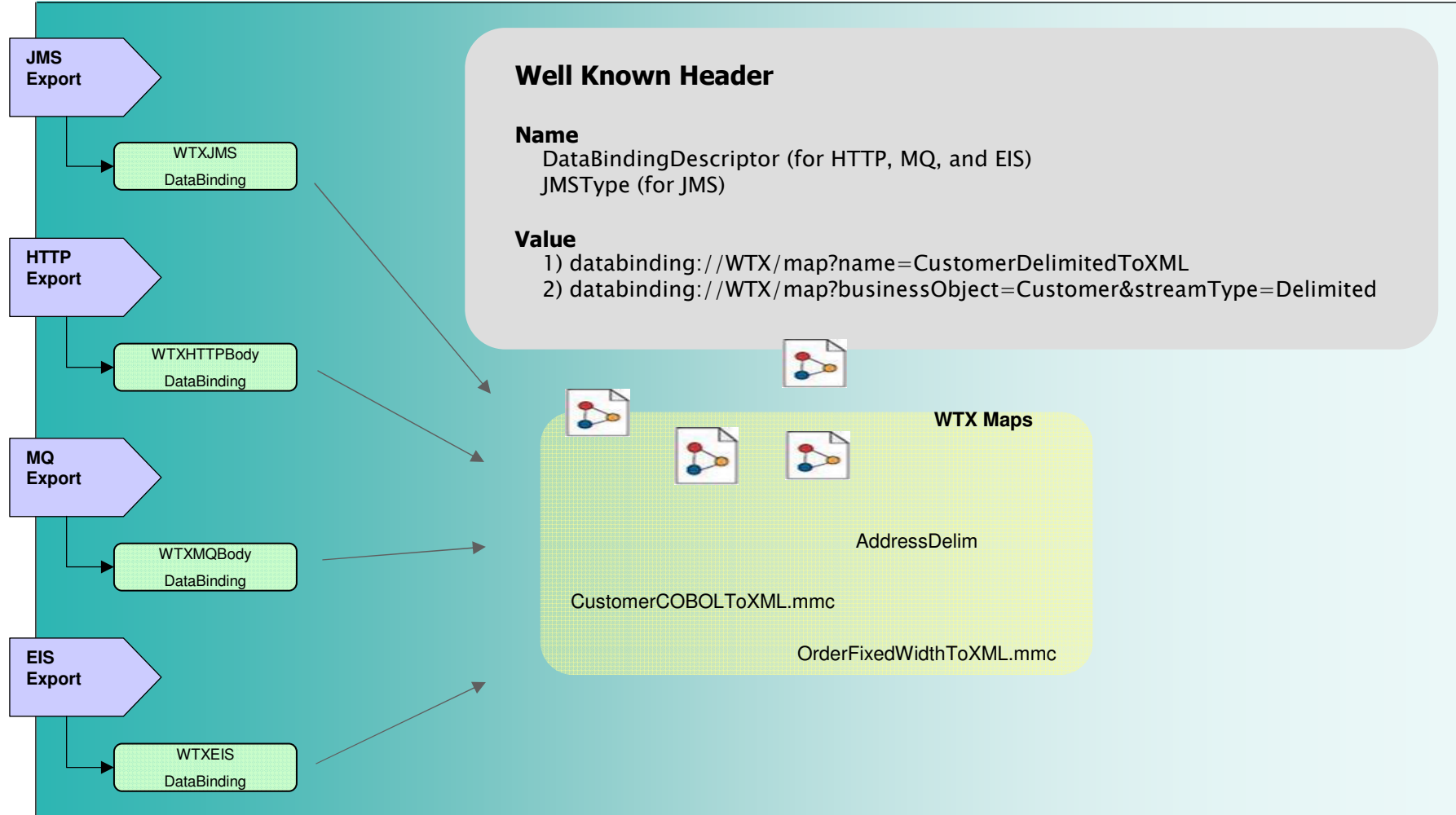
- WebSphere Transformation Extender Integration
  - Tooled Data Bindings
  - Semi-structured to Business Object Transformations
- WPS/WESB & WTX
  - WPS/WESB will provide the data bindings
  - WTX 8.2 must be installed in order to use the data binding
- Exports & Imports
  - JMS, MQ, HTTP, EIS
  - Flexible bindings (any data any format)



# WTX Data Bindings

## Any Business Object, Any Format

SCA Module





## ***Mediation Updates***

**New and changed mediation primitives  
Service Message Object Updates**

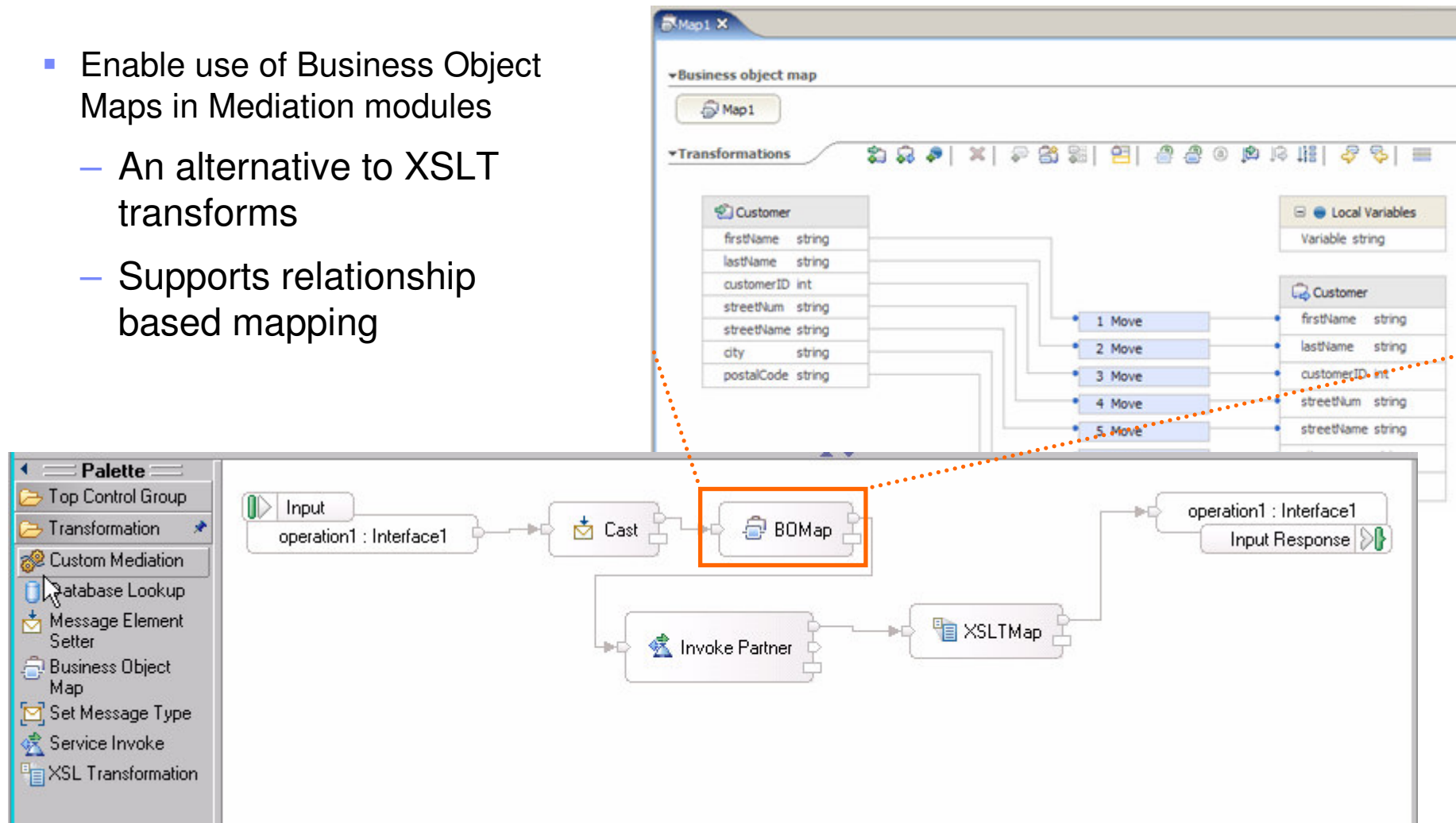
**WebSphere** software



## WESB Enhancements

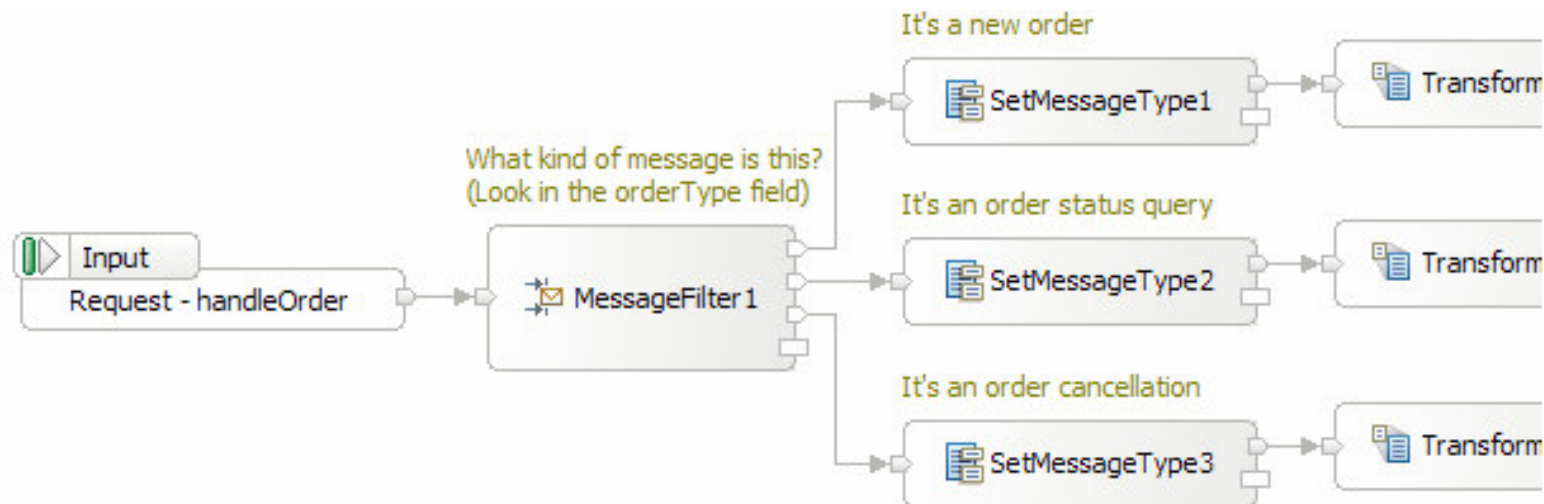
### BO Map Primitive

- Enable use of Business Object Maps in Mediation modules
  - An alternative to XSLT transforms
  - Supports relationship based mapping



## Mediation Flow Component Type Refinement [New]

- Ability to assert concrete types for message elements described by ‘weak types’ such as anyType, anySimpleType or any
- New SetMessageType mediation primitive
  - Conceptually performs a ‘cast’ operation on any part of the message
  - Works with other primitives and XPath support to provide full access to weakly typed message content in mediation flows
  - Weakly typed content can be visualized, mapped and manipulated as if its structure were fully defined in the original BO definition



# WESB Enhancements

## Service Invocation and Retry Primitive

- Invokes a target service from within a request or response flow
- Includes built-in retry capability; retries x times in event of failure
- Can try/retry a list of target endpoints in turn until success
  - Can exploit multiple endpoints returned from the Endpoint Lookup primitive for this purpose
- Also acts as a building block for aggregating content from more than one service
- Capability similar to existing Callout, but flow continues inline (no switch to response flow)
- New context area in the SMO contains invocation request/response body content
- Service A invocation can be asynchronous

**Service Invoke : Invoke Partner**

Reference Name:

Operation Name:

Retry On:

Retry Count:

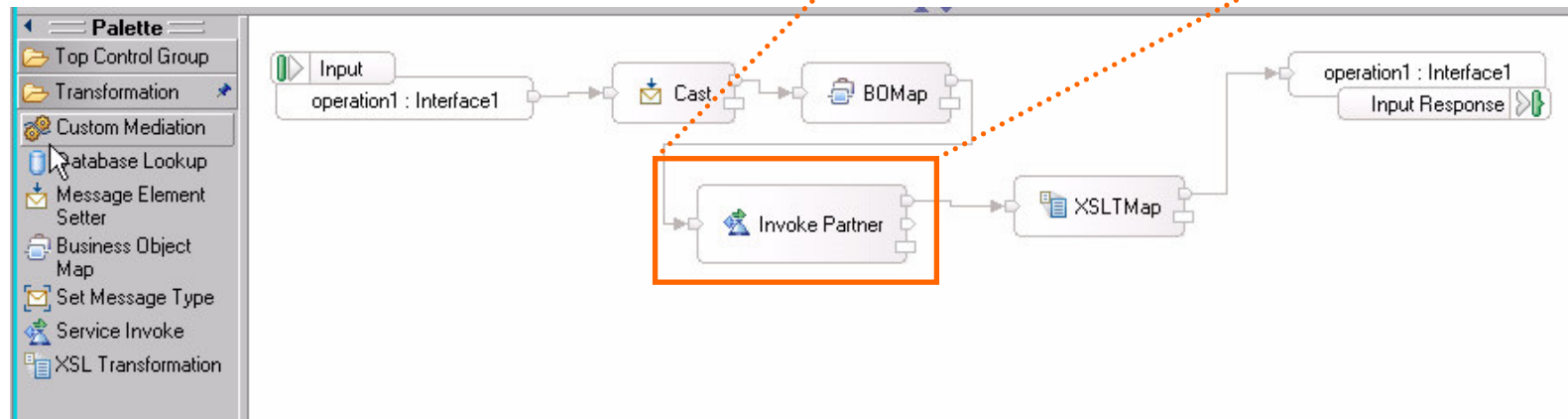
Retry Delay:

Use Dynamic Endpoint

Try Alternate Endpoints

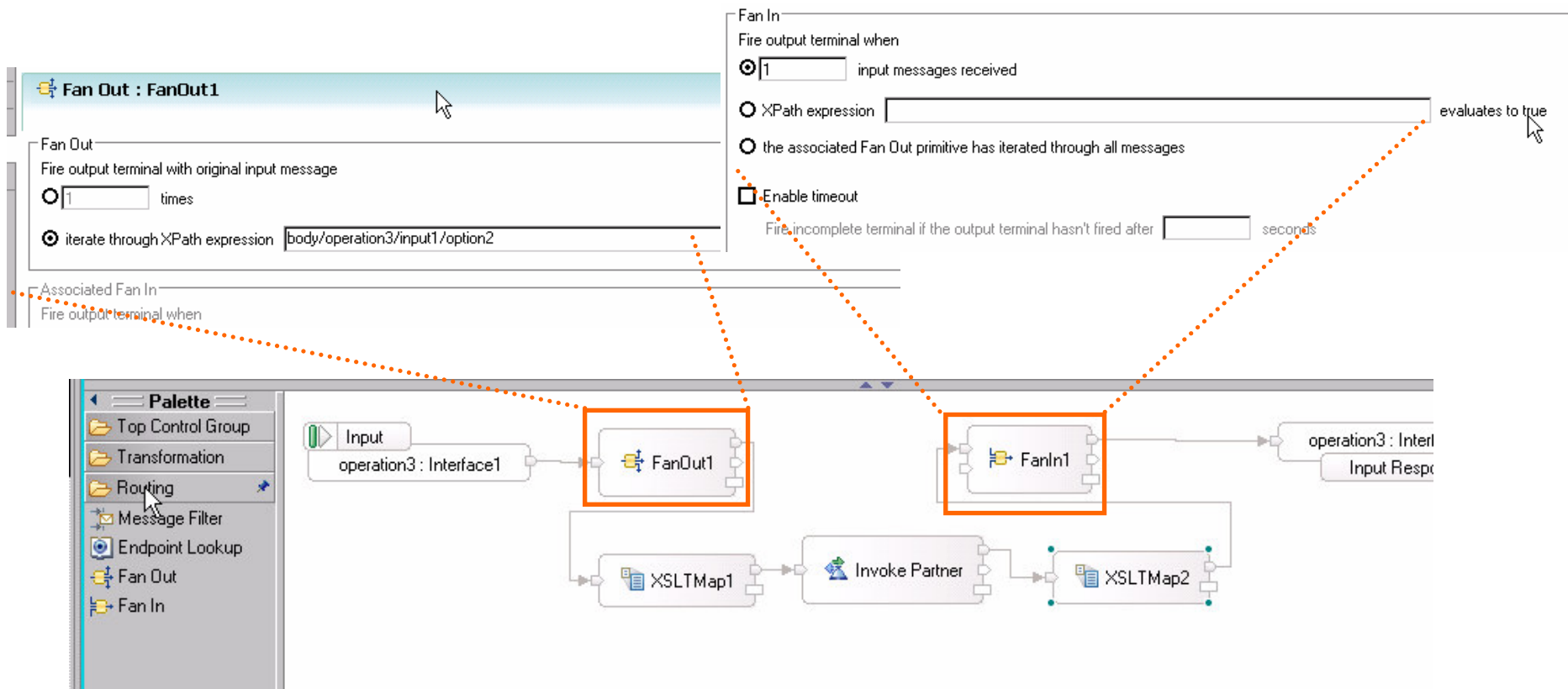
Async Timeout:

Force Sync



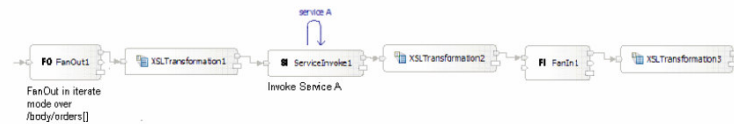
## New: FanOut and FanIn Mediation Primitive

- Messages can be split for further processing
- Results can be aggregated

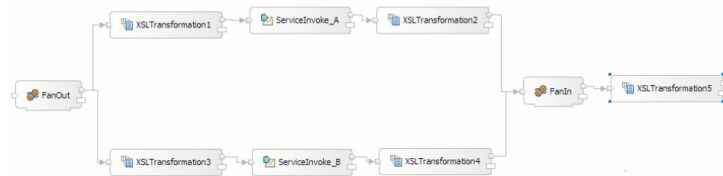


## Splitting and Aggregating - details

- New FanOut and FanIn primitives
  - FanOut splits an incoming message based upon a repeating element



- ...or sets up branching paths that are later joined by FanIn



- Allows composite messages to be split up for individual processing of the parts; and assembly of composite messages
- Aggregation supported using ServiceInvoke primitive
  - Invoke multiple services and combine the results





# WESB Enhancements

## Custom Mediation Primitive

- Multiple input and output terminals
- User-defined properties
- Easy access to ESB mediation primitive programming model

**Custom Mediation : CustomMediation1**

CustomPropertyGroup CustomUserProperties CustomJavaImports

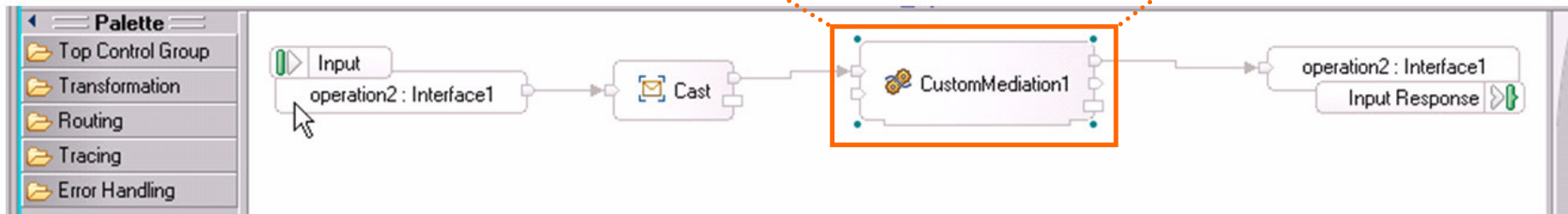
User Properties:

Name	Type	Value	Required
frequency	Integer	5	<input checked="" type="checkbox"/>

Implementation:  Visual  Java

```

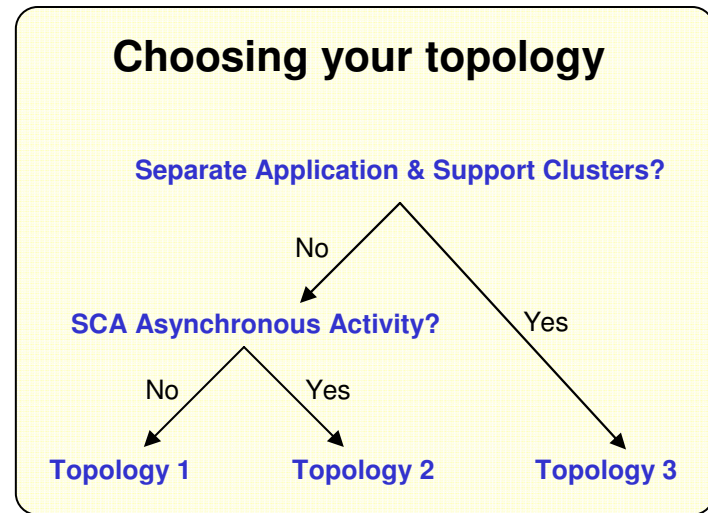
/**
 * Variables: for output terminals - out1, out2 (com.ibm.wsspi.sib)
 *           for user properties - frequency (int)
 * Inputs:    inputTerminal (com.ibm.wsspi.sibx.mediation.InputTerm)
 * Exceptions: com.ibm.wsspi.sibx.mediation.MediationConfigurationE
 */
// Fire the output terminal(s)
out1.fire(smo);
out2.fire(smo);
    
```



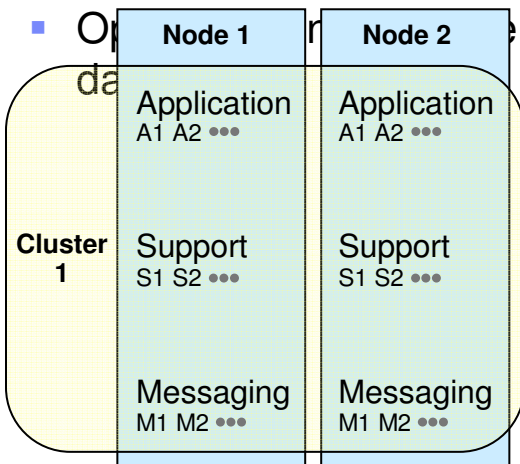


# Template-driven ND topology

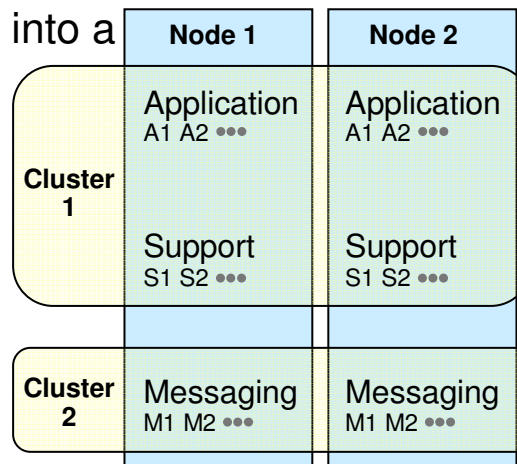
- Wizard driven approach to configuring your ND topology
  - Install support
  - Administration support
- Three primary roles nodes can play
  - Customer applications
  - WPS/WESB Support applications
  - Messaging (Destinations)



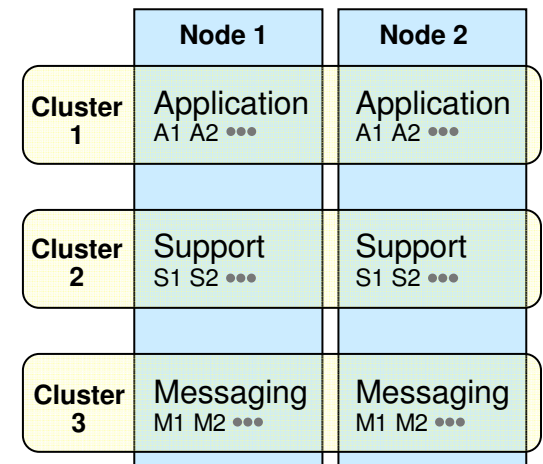
■ On data everything into a



Topology 1: Single Cluster



Topology 2: Remote Messaging



Topology 3: Remote Messaging and Support



## *Summary and Conclusion*

WebSphere software



## ***Conclusion***

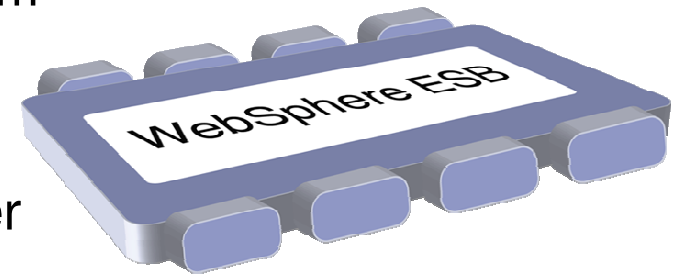
- Key evolution in 6.1, centering on
  - Consumability
    - both administration and development interfaces
  - Completeness of capability
    - Additional connectivity
    - Additional mediation function
- Continued maturation and evolution



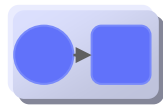
## **WebSphere ESB for z/OS**

**Built on WebSphere Application Server for an integrated SOA platform**

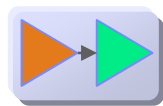
- Integrates seamlessly with WebSphere platform
- Delivers business-critical qualities of service
- Easily extended to WebSphere Process Server
- Integrated solution for service mediation and hosting



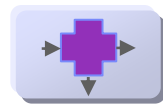
Delivers leadership in SOA standards for service composition, and leverages the embedded messaging and web services engines from WebSphere



Integrates everything with WebSphere Adapters for enterprise applications, the breadth of the WebSphere ecosystem, and support for standard protocols



Optimized for standard XML and web services formats, with basic support for other common formats



Provides business visibility with embedded event engine for Business Activity Monitoring solutions

