Using WebSphere Enterprise Service Bus for z/OS

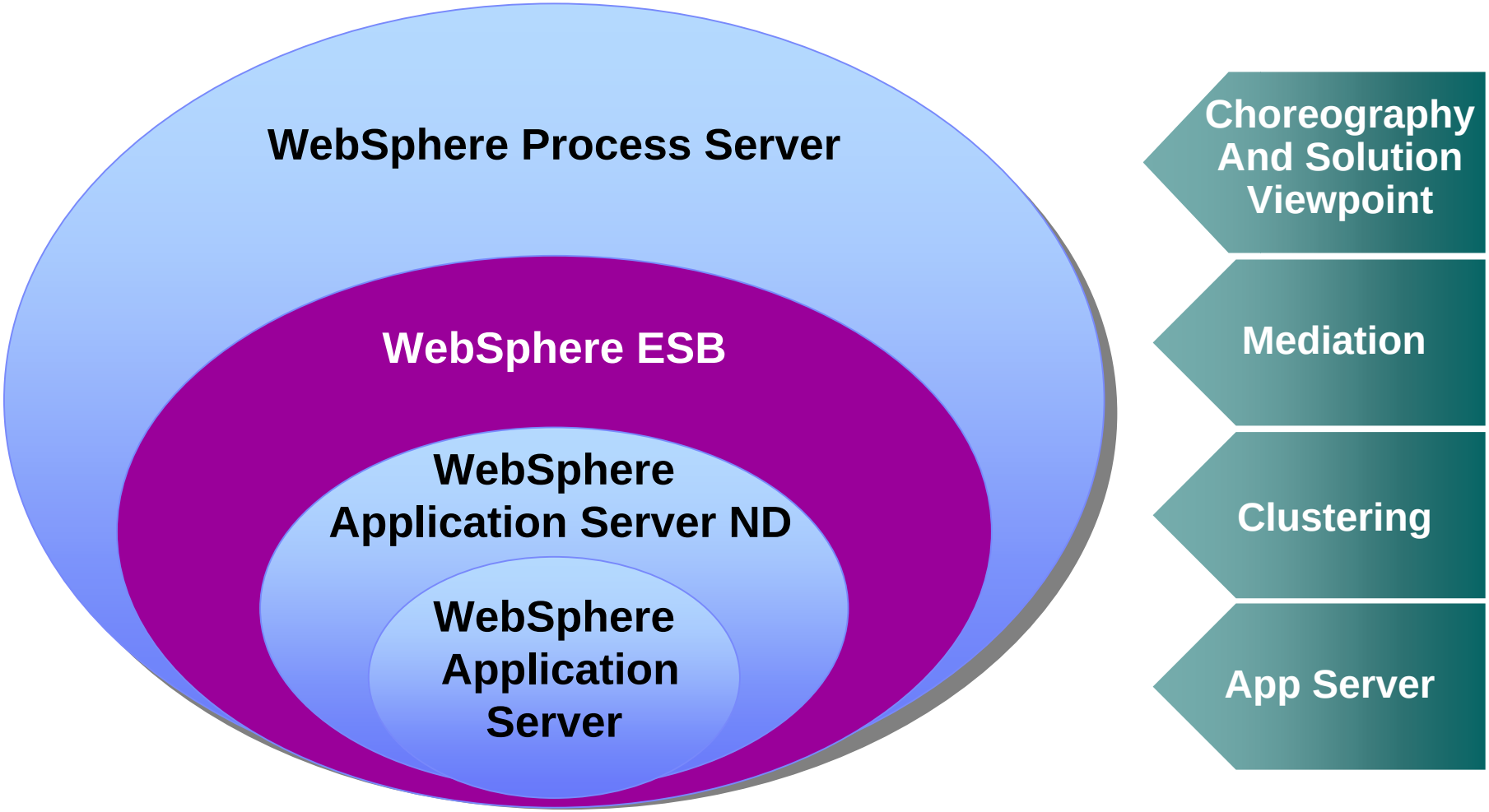
Andrew Mead

Hursley, UK

Agenda

- What is WebSphere Enterprise Service Bus
- What was new with V6.1 & V6.1.2 and looking to 6.2
- A System z customer scenario

WebSphere Application Server, ESB, and Process Server



WebSphere ESB

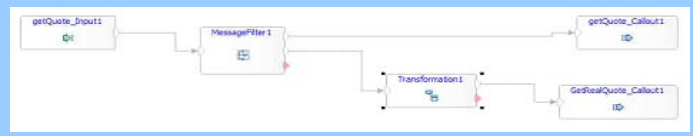
Adapters

Flat File
JDBC
Email
FTP
SAP R/3
Siebel Business Applications
JD Edwards
Oracle E-Business Suite
PeopleSoft

WebSphere TX Integration

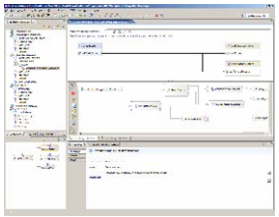
Any JMS 1.1
WebSphere MQ
HTTP

WebSphere ESB

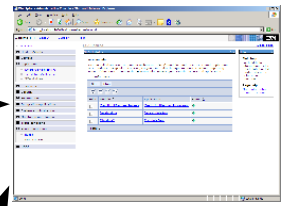


XSLT
WSRR Endpoint Lookup
Message Logger
DB Lookup
Set Msg Type
Service Retry
Msg Element Setter
Message Router
CEI
ITCAM Mgmt
Message Splitter
Message Aggregation
Custom

WebSphere Application Server



WebSphere Integration Developer



WebSphere Admin Console

Web Services:

SOAP/ HTTP
SOAP/ JMS
UDDI Registry 3.0
WS-*

AIX, HP-UX (PA-RISC, Itanium), i5/OS, Linux (Intel, p-Series,linux on System z)
Solaris (SPARC, X86-64), Windows, z/OS

SCA Programming Model

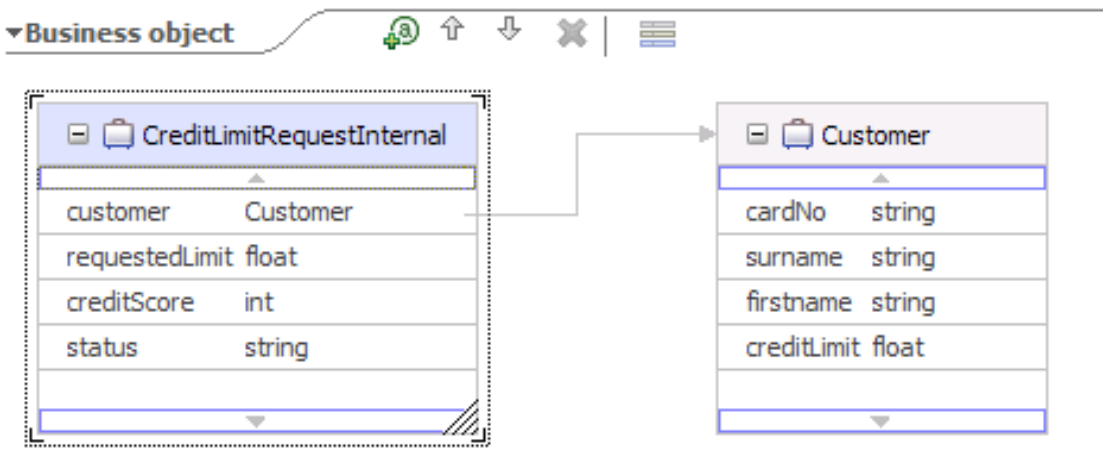
SCA
SDO
SMO

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

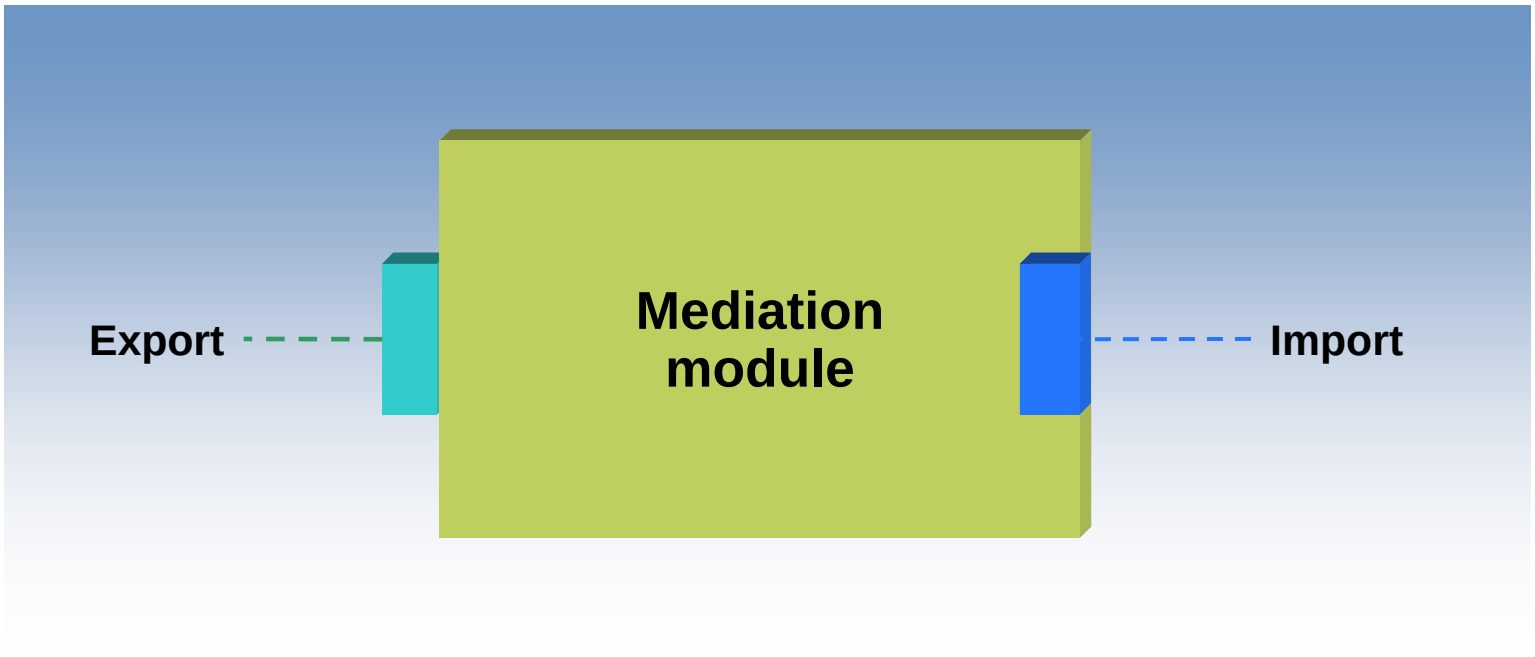
Business object definition (SDO/XML schema)



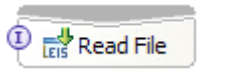
- Business Objects represent structured application data
- Consistent logical representation, independent of data source or wire format
- Based upon SDO standard
- Multiple data bindings can be supported

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



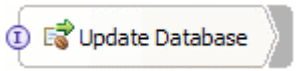
Export/Import Bindings



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

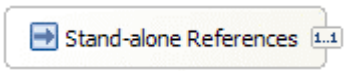
Web services (SOAP/HTTP, SOAP/JMS)

JMS / WebSphere MQ JMS / WebSphere MQ



EJB

Native (SCA)



Java invocation

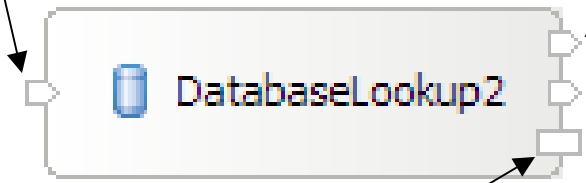
Concepts: Integrating request-response interactions using WebSphere ESB Mediation Flow Components

- **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

One or more typed input Terminal (shipped primitives only have one)

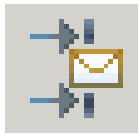
0..n output typed output terminals



Fail terminal fired if processing fails

- Acts like a Java catch block if wired
- If not wired then causal exception is thrown up causing flow failure
- Receives original message from input terminal

WebSphere ESB v6.0.1 Mediation Primitives



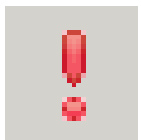
MessageFilter – routes within a flow

XSLTransformation – transforms content

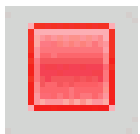
MessageLogger – logs to database

DatabaseLookup – adds content from database

CustomMediation – executes custom logic (Java)

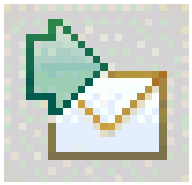


Fail – terminates flow with an exception

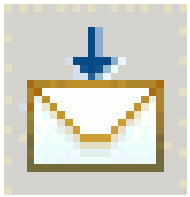


Stop – terminates flow normally

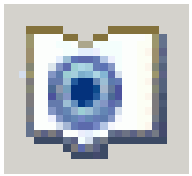
WebSphere ESB v6.0.2 Additional Primitives



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

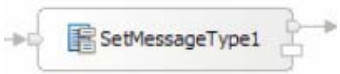


Endpoint Lookup - queries the WebSphere® Service Registry and Repository and retrieves service endpoints, which it places in the message context. The retrieved endpoints can then be used to dynamically invoke a service

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)

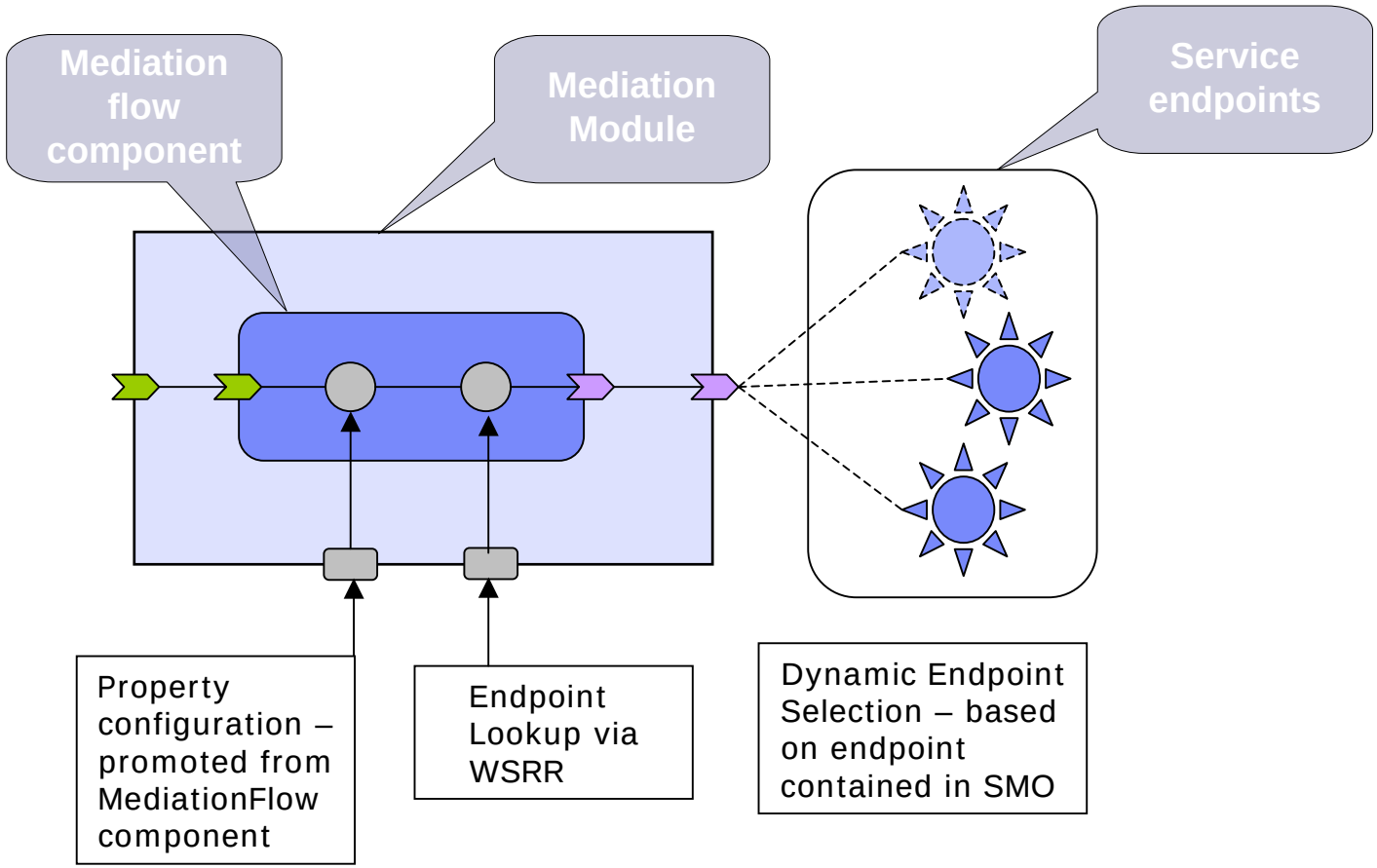


Set Message Type: allows a weakly typed element (xs:any, xs:anyType, ...) to have a type asserted for it at runtime

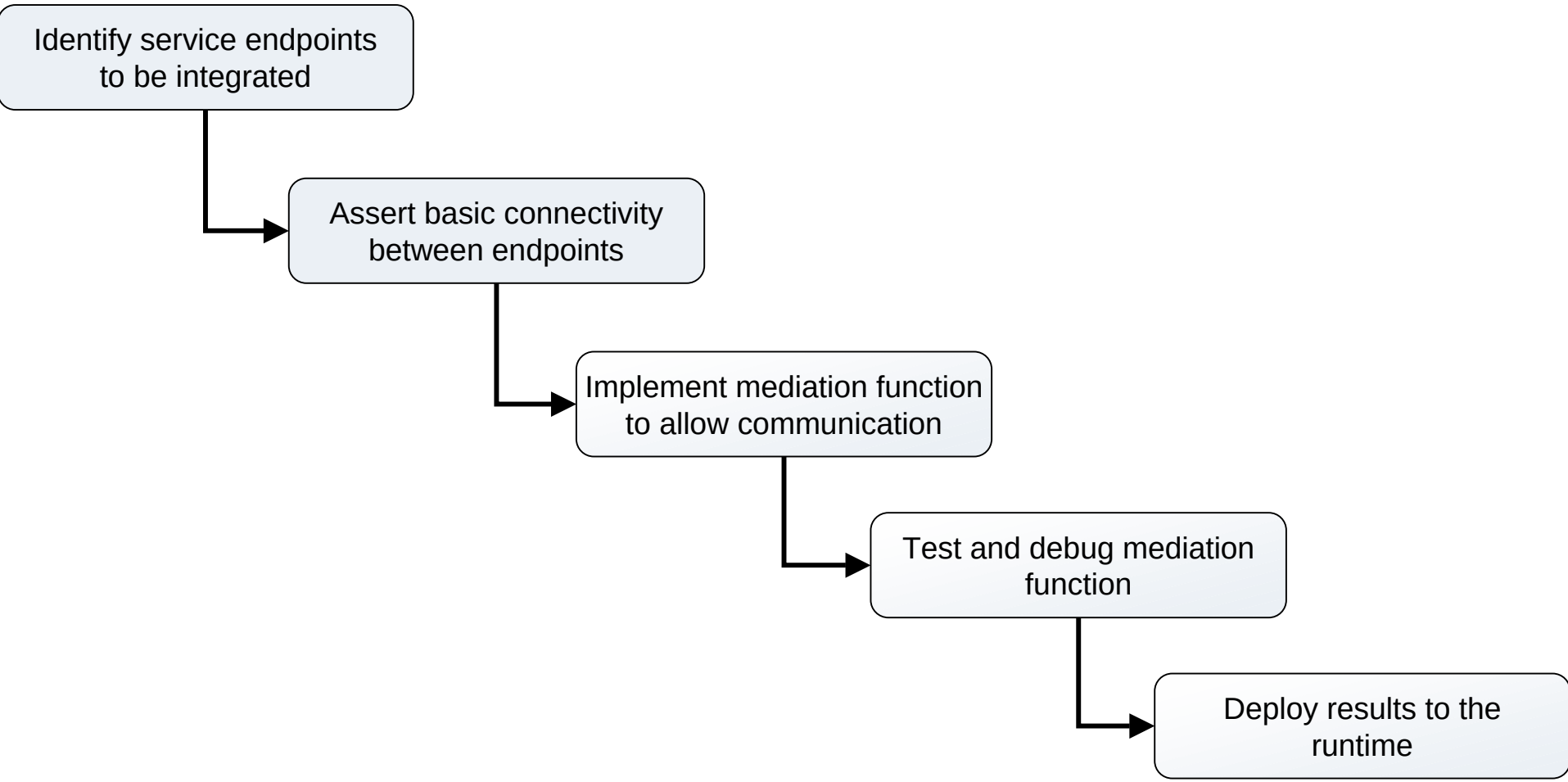
Fanout: allows a flow to be split - and to operate on repeating elements of the message, and may be paired with **Fanin**

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

Dynamic Service Invocation

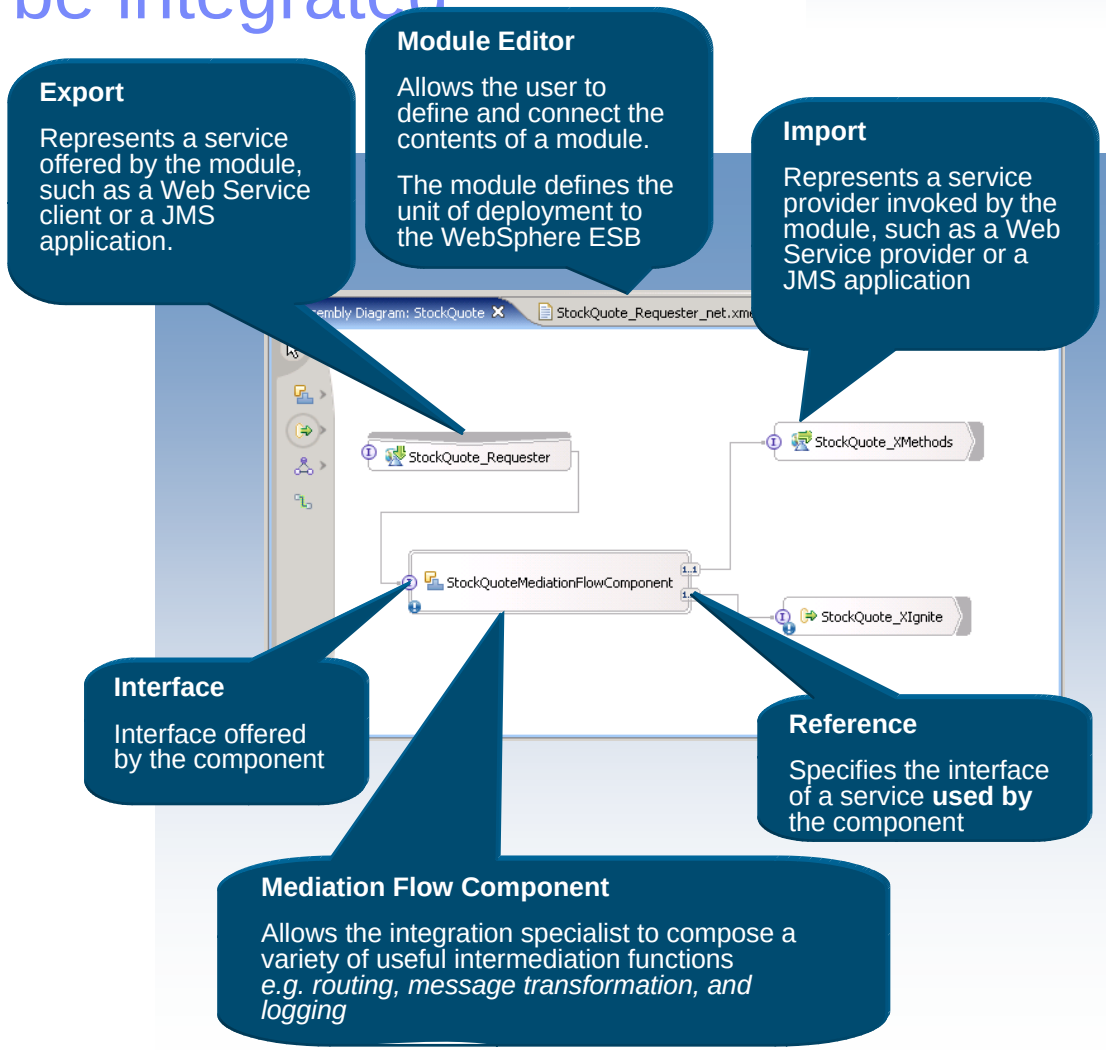


Typical Integration Developer Task Flow



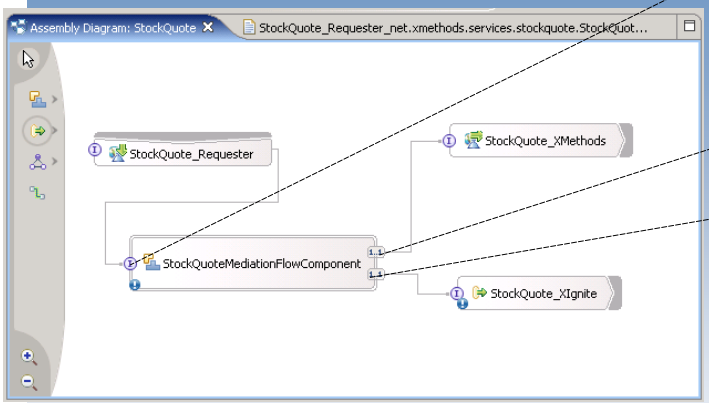
1. Integration developer specifies the service endpoints that need to be integrated

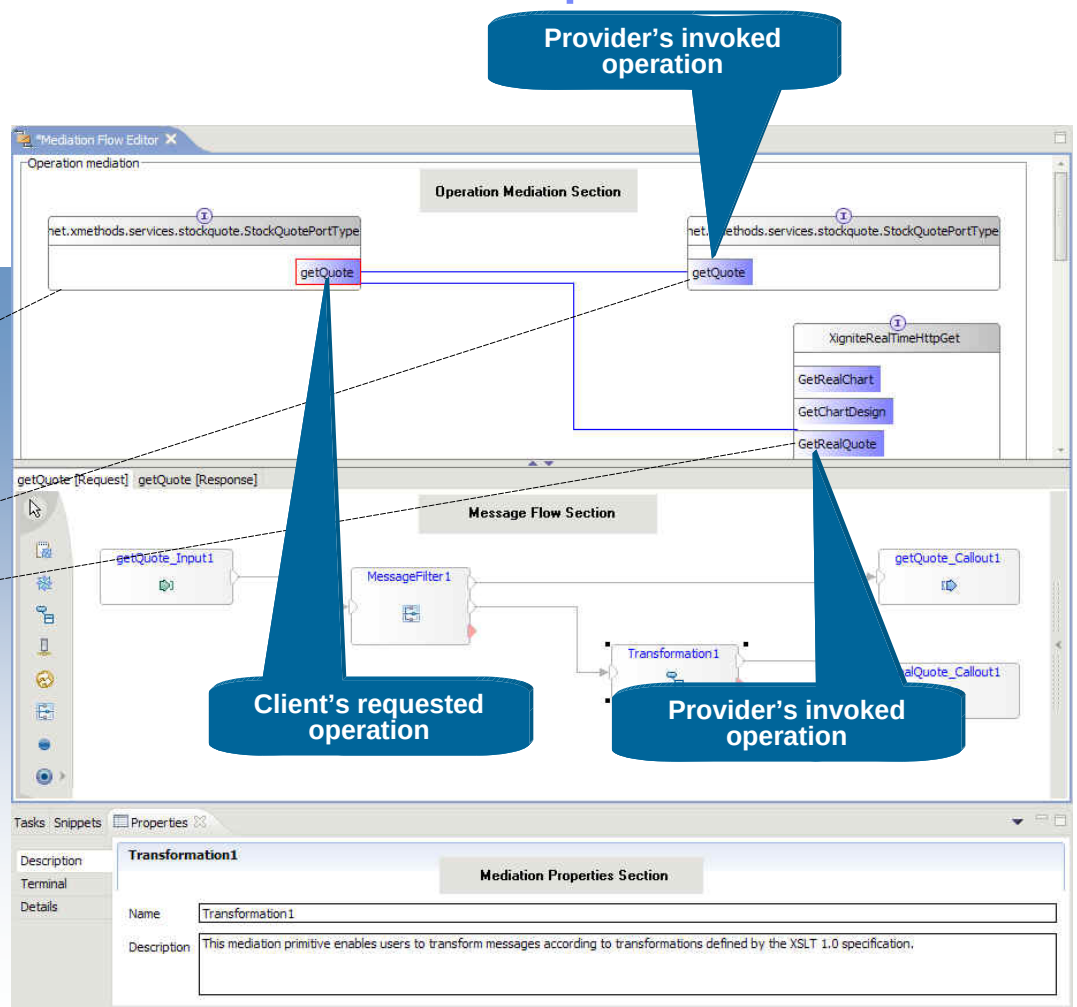
- **Uses the 'module editor' to construct a mediation module**
  - specifies how a subset of WebSphere ESB's service requesters and service providers interact
- **Within the module**
  - Service requesters are represented as 'exports'
  - Service providers are represented as 'imports'
  - The integration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

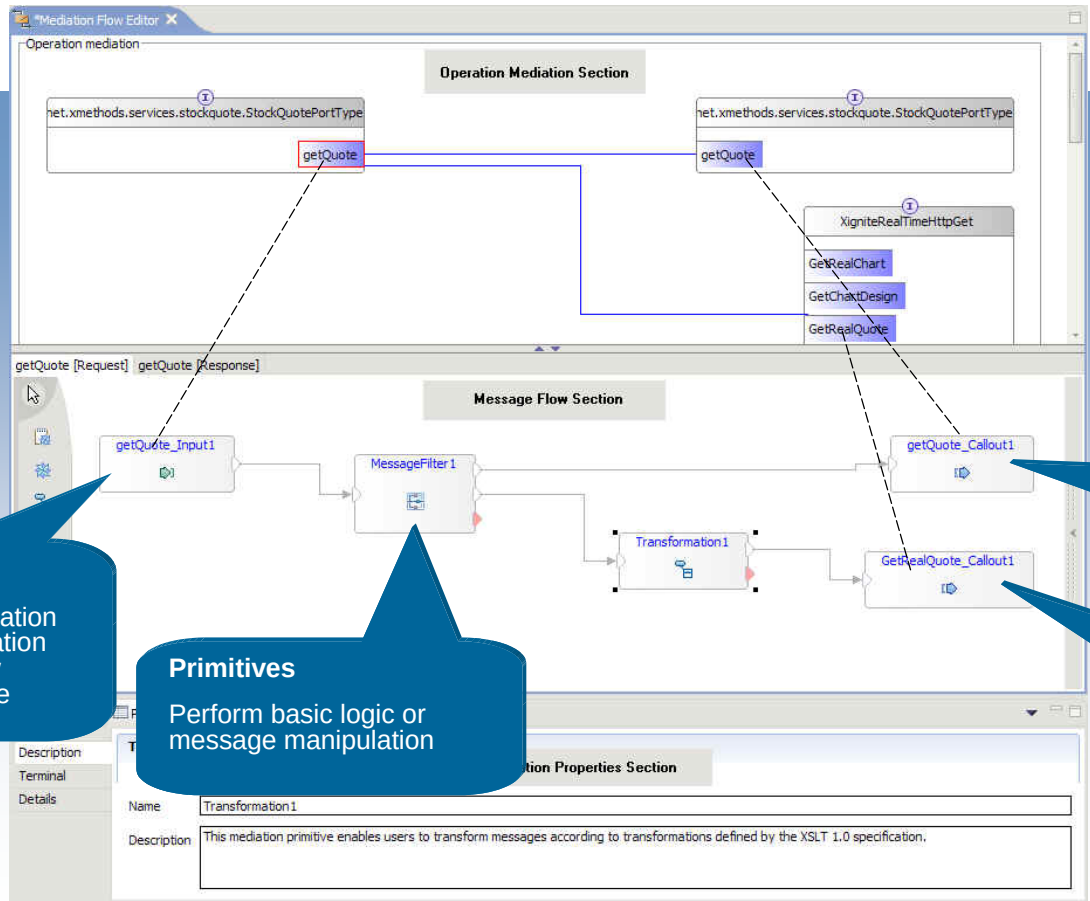




Integration Developer

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'

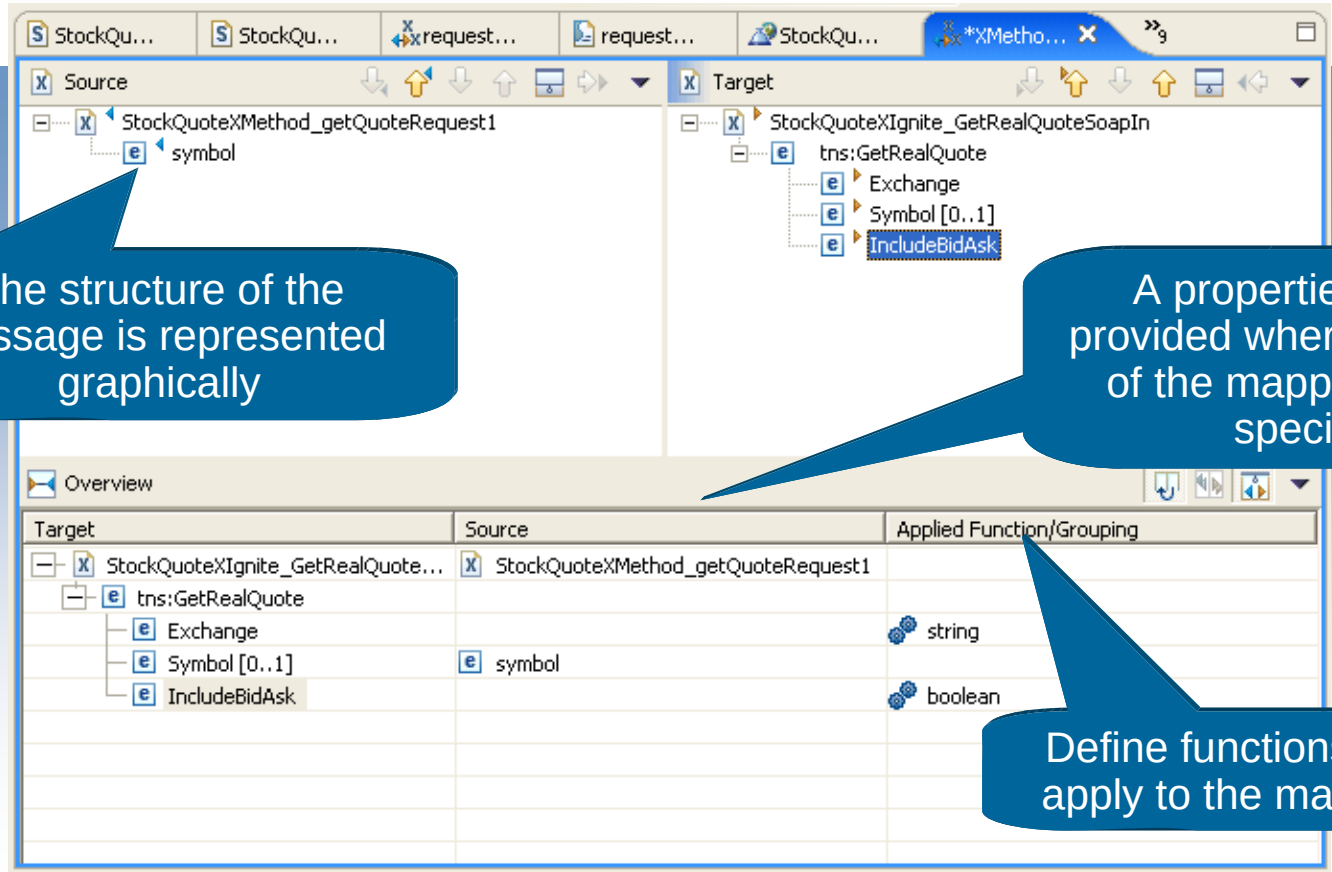
Causes the specified operation to be invoked

'Callout'

Causes the specified operation to be invoked

4. Integration developer customizes the elements of the mediation flow

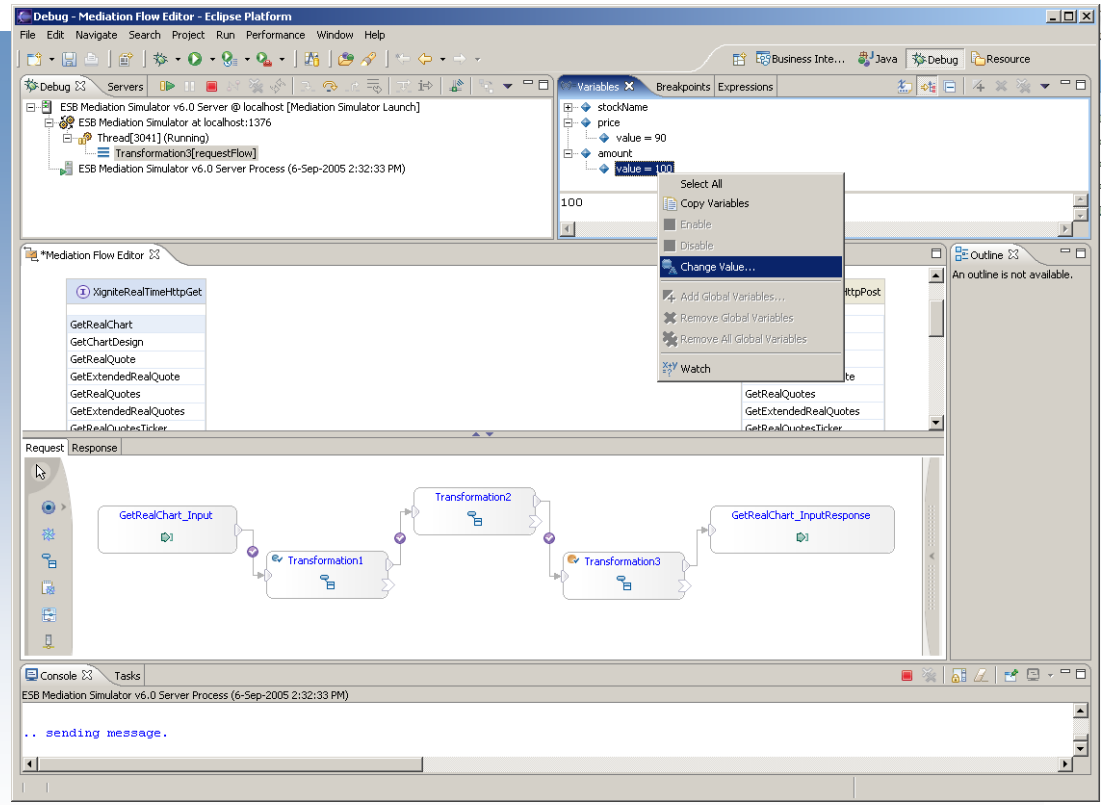
- e.g. Customizes the XSLT transform mediation primitive by using mapping tool to construct an XSLT transform



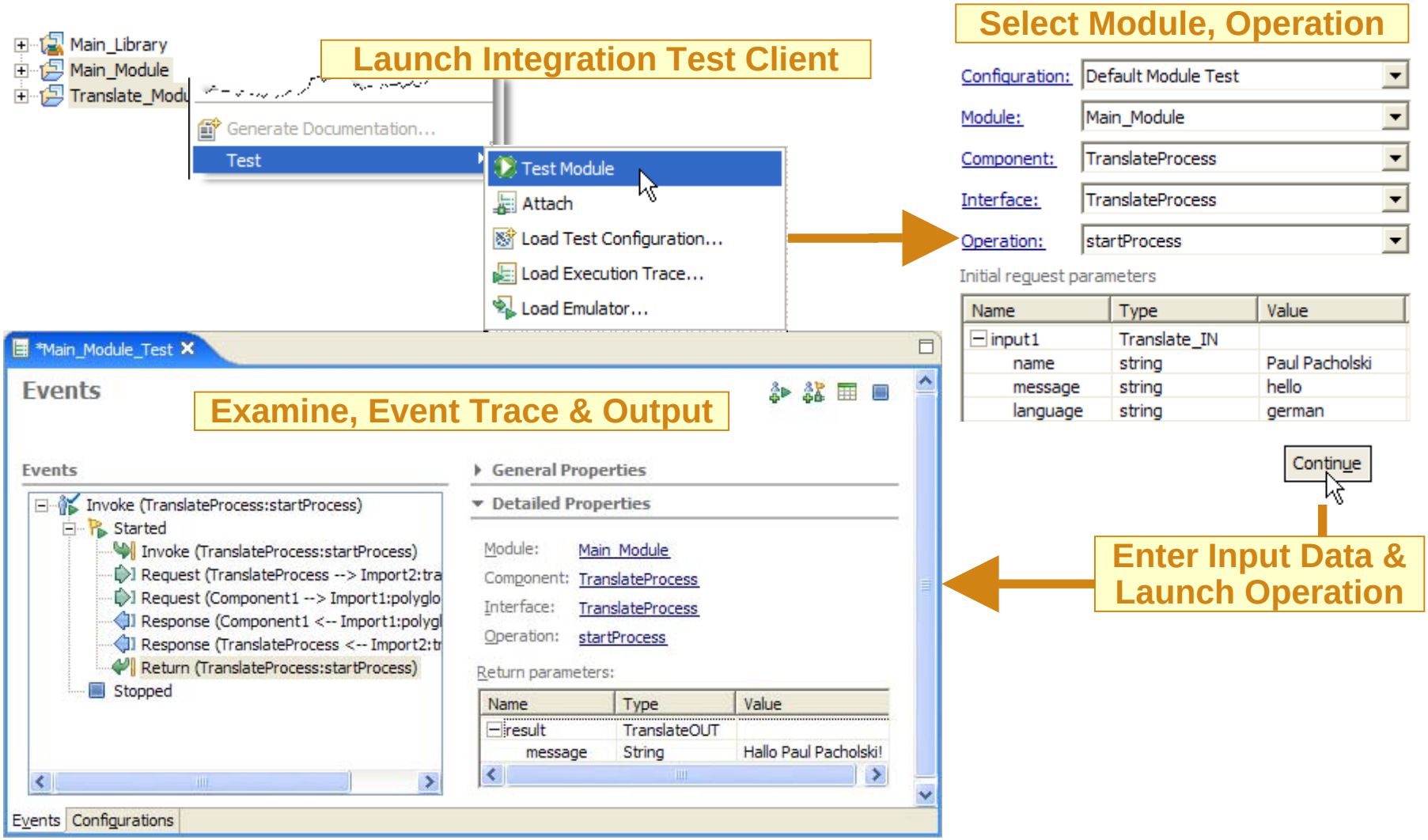
5. Debugs the composed/configured mediation function: WID Mediation Visual Debug

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

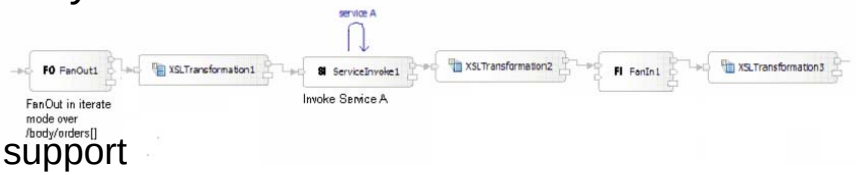


Test and Debug – Integration Test Client

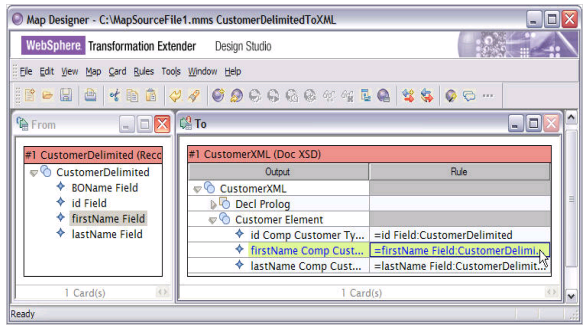


What was new in WebSphere Enterprise Service Bus v6.1

- 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

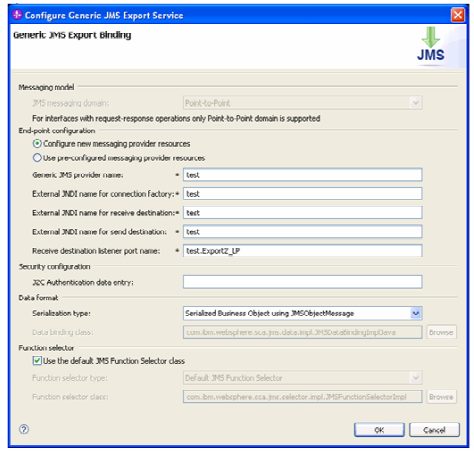


- Integrates a broader range of services with expanded connectivity
  - New WebSphere TX integration
  - New WS-Notification support
  - New generic HTTP support
  - Enhanced 3rd party JMS support



- Enables streamlined operational infrastructure with platform currency
  - Updated WAS 6.1 based runtime
  - 64 bit enabled

- 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

WESB 6.2 on z/OS GA Jan 16th 2009

- Increased flexibility to administratively configure service mediations through policies
- Full set of mediation capabilities manageable via policies
- Integration with WebSphere Service Registry and Repository for policy management and governance
- Based on WAS 6.1

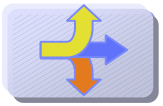
WebSphere ESB for z/OS – Extends the value of WebSphere Application Server for z/OS

- 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
  - Integration with z/OS Workload Manager so you can set performance goals for user requests.
  - Built-in servant scalability, so WLM can start them up when performance goals are being missed.
  - Exploitation of RRS for transaction management.
  - Cuts SMF records to support monitoring and management.
  - 64-bit support for larger heaps.

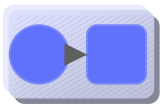
WebSphere ESB for z/OS

Built on WebSphere Application Server for an integrated SOA platform

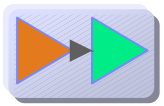
- Integrates seamlessly with WebSphere platform
- Delivers business-critical qualities of service
- Easily extended to WebSphere Process Server
- Integrated solution for service mediation and hosting



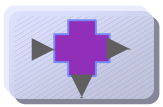
Delivers leadership in SOA standards for service composition, and leverages the embedded messaging and web services engines from WebSphere



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

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 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

Mediation Updates

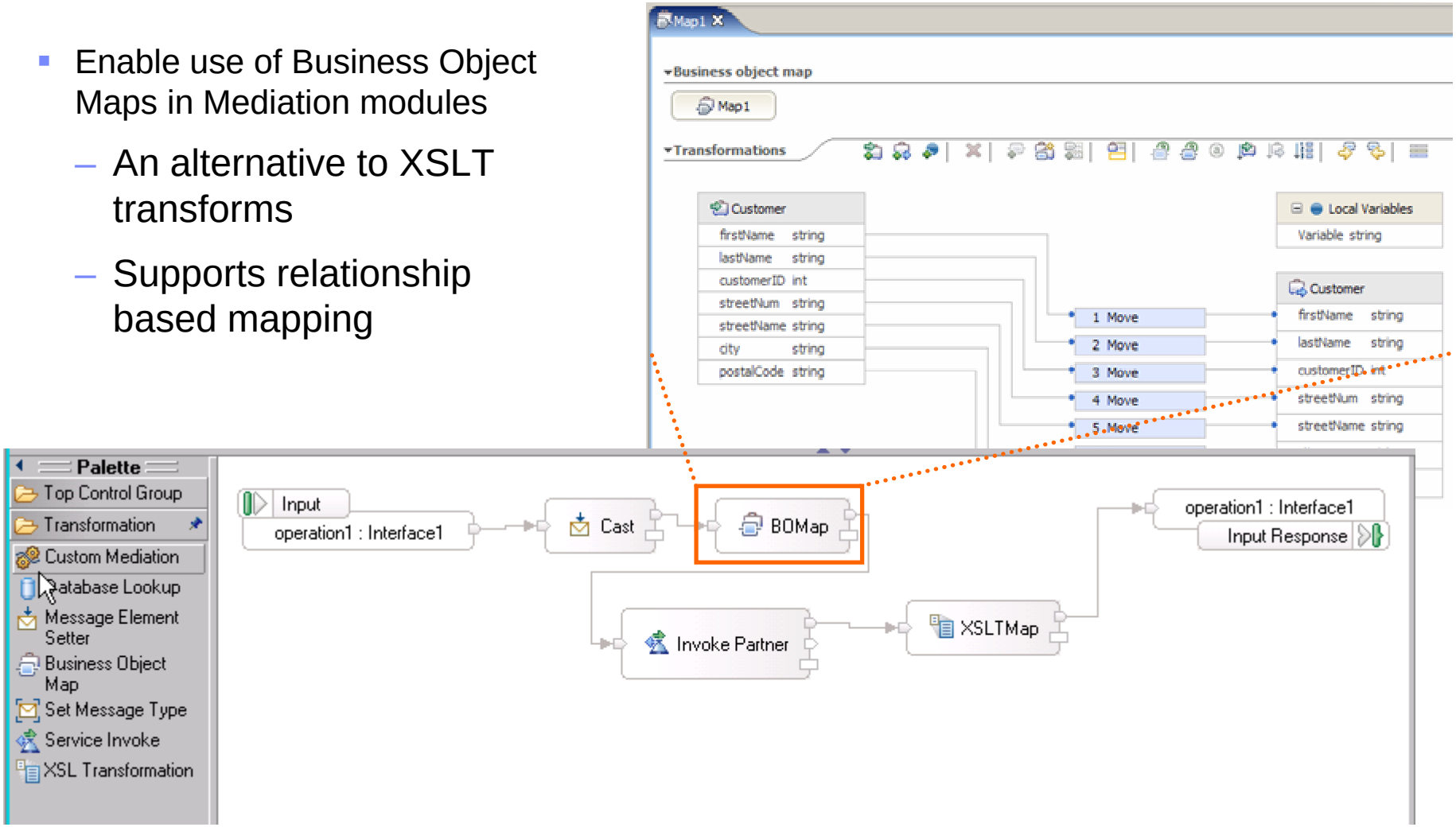
New and changed mediation primitives
Service Message Object Updates

WebSphere software

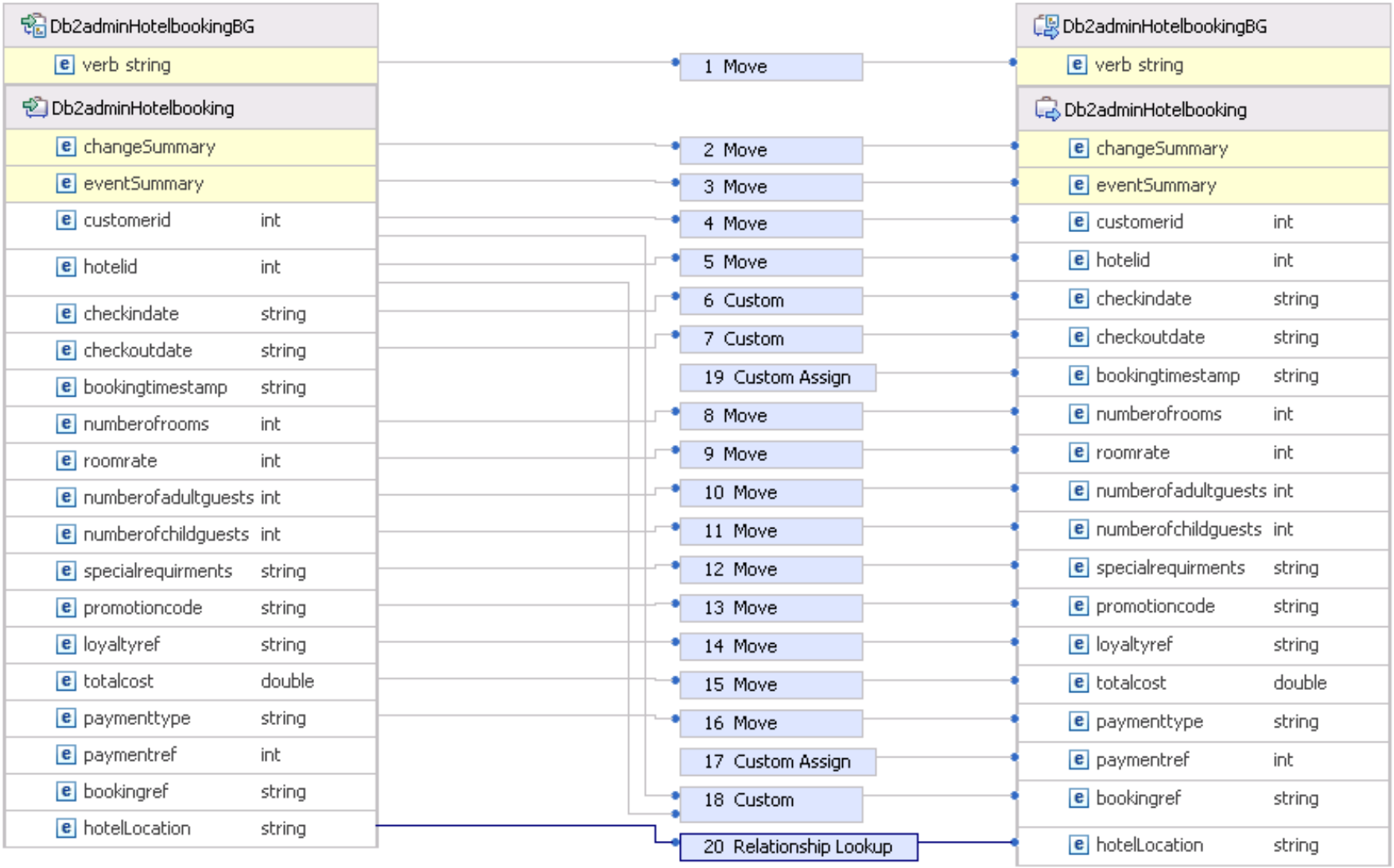
WESB Enhancements

BO Map Primitive

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

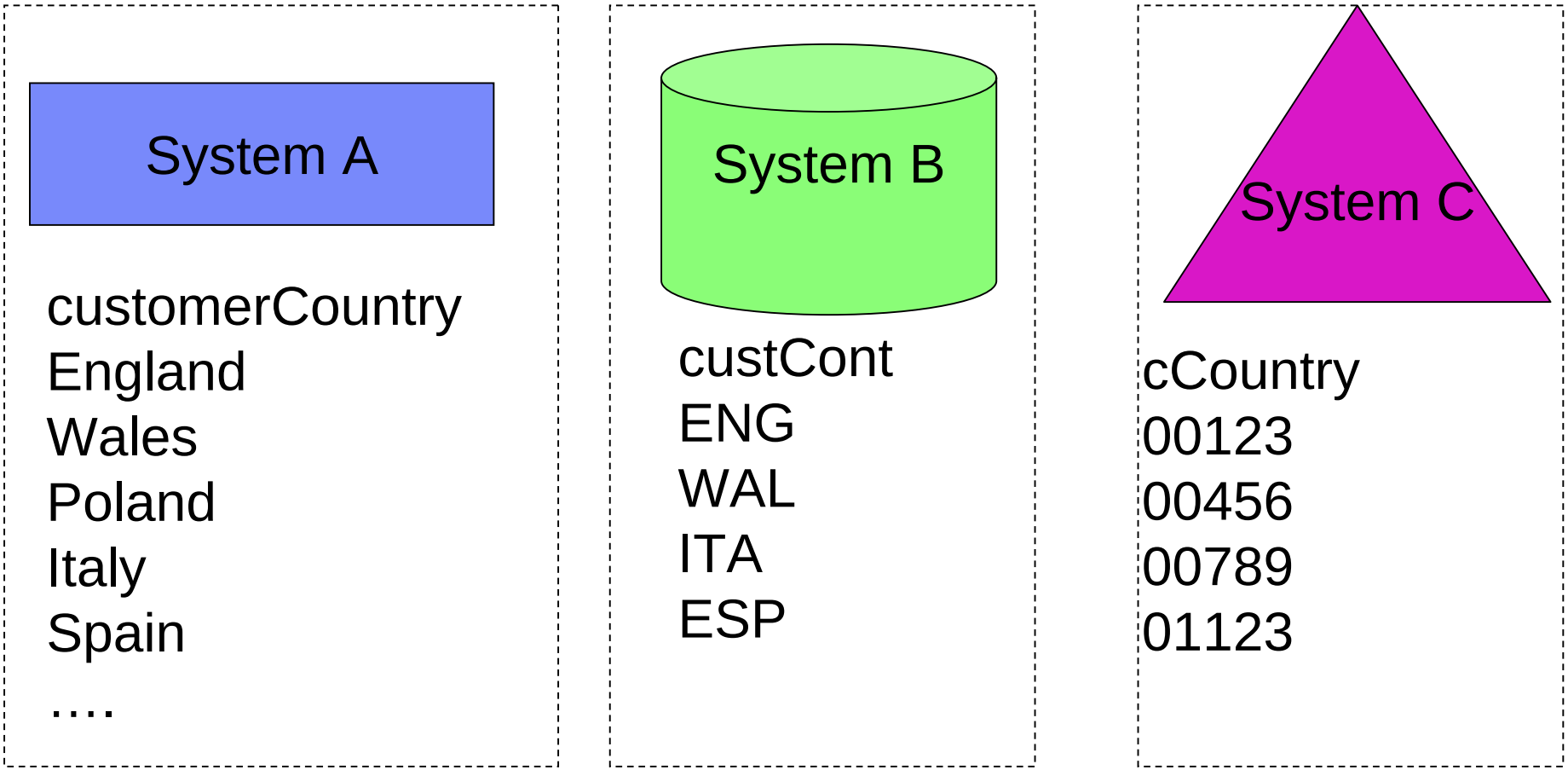


SVT example


Db2adminHotelbookingBG
verb string
Db2adminHotelbooking
changeSummary
eventSummary
customerid int
hotelid int
checkindate string
checkoutdate string
bookingtimestamp string
numberofrooms int
roomrate int
numberofadultguests int
numberofchildguests int
specialrequirments string
promotioncode string
loyaltyref string
totalcost double
paymenttype string
paymentref int
bookingref string
hotelLocation string
1 Move
2 Move
3 Move
4 Move
5 Move
6 Custom
7 Custom
19 Custom Assign
8 Move
9 Move
10 Move
11 Move
12 Move
13 Move
14 Move
15 Move
16 Move
17 Custom Assign
18 Custom
20 Relationship Lookup
Db2adminHotelbookingBG
verb string
Db2adminHotelbooking
changeSummary
eventSummary
customerid int
hotelid int
checkindate string
checkoutdate string
bookingtimestamp string
numberofrooms int
roomrate int
numberofadultguests int
numberofchildguests int
specialrequirments string
promotioncode string
loyaltyref string
totalcost double
paymenttype string
paymentref int
bookingref string
hotelLocation string

Relationships

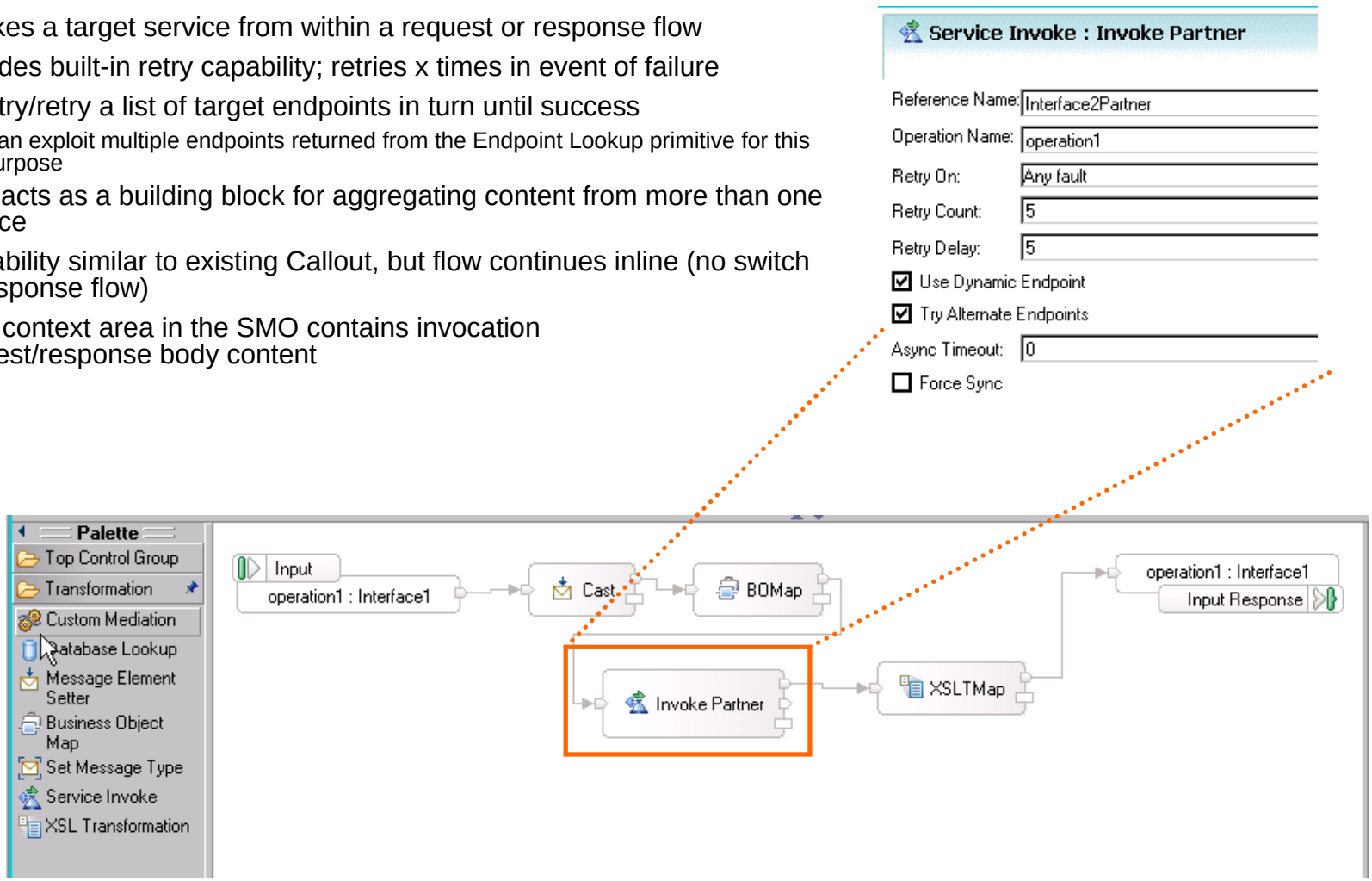
- Used to map same data that has different representations in different systems.



WESB Enhancements

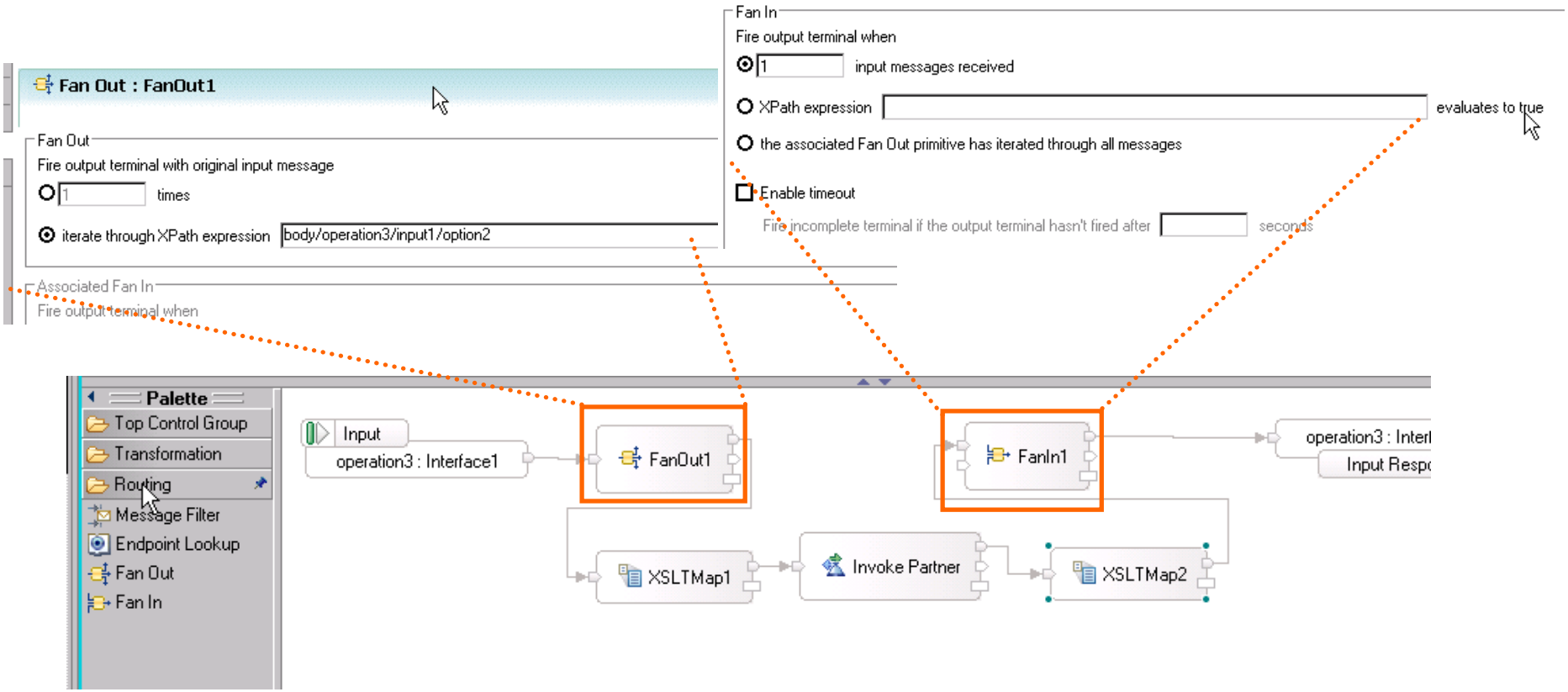
Service Invocation and Retry Primitive

- 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



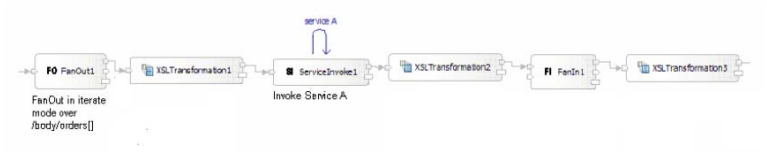
New: FanOut and FanIn Mediation Primitive

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



Splitting and Aggregating - details

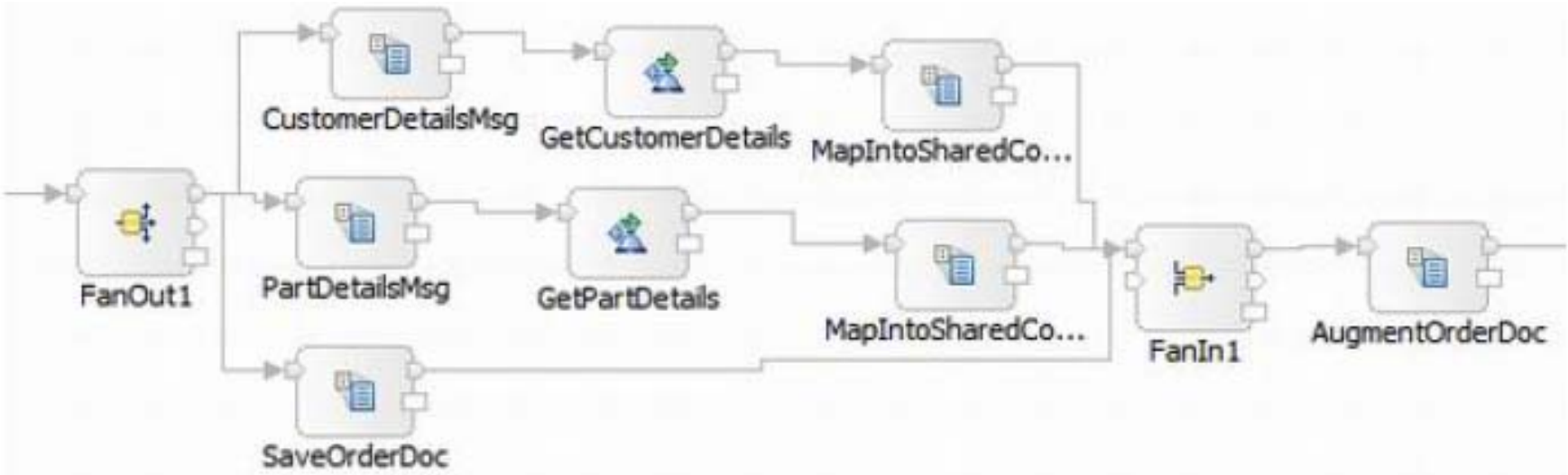
- New FanOut and FanIn primitives
  - FanOut splits an incoming message based upon a repeating element



  - ...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

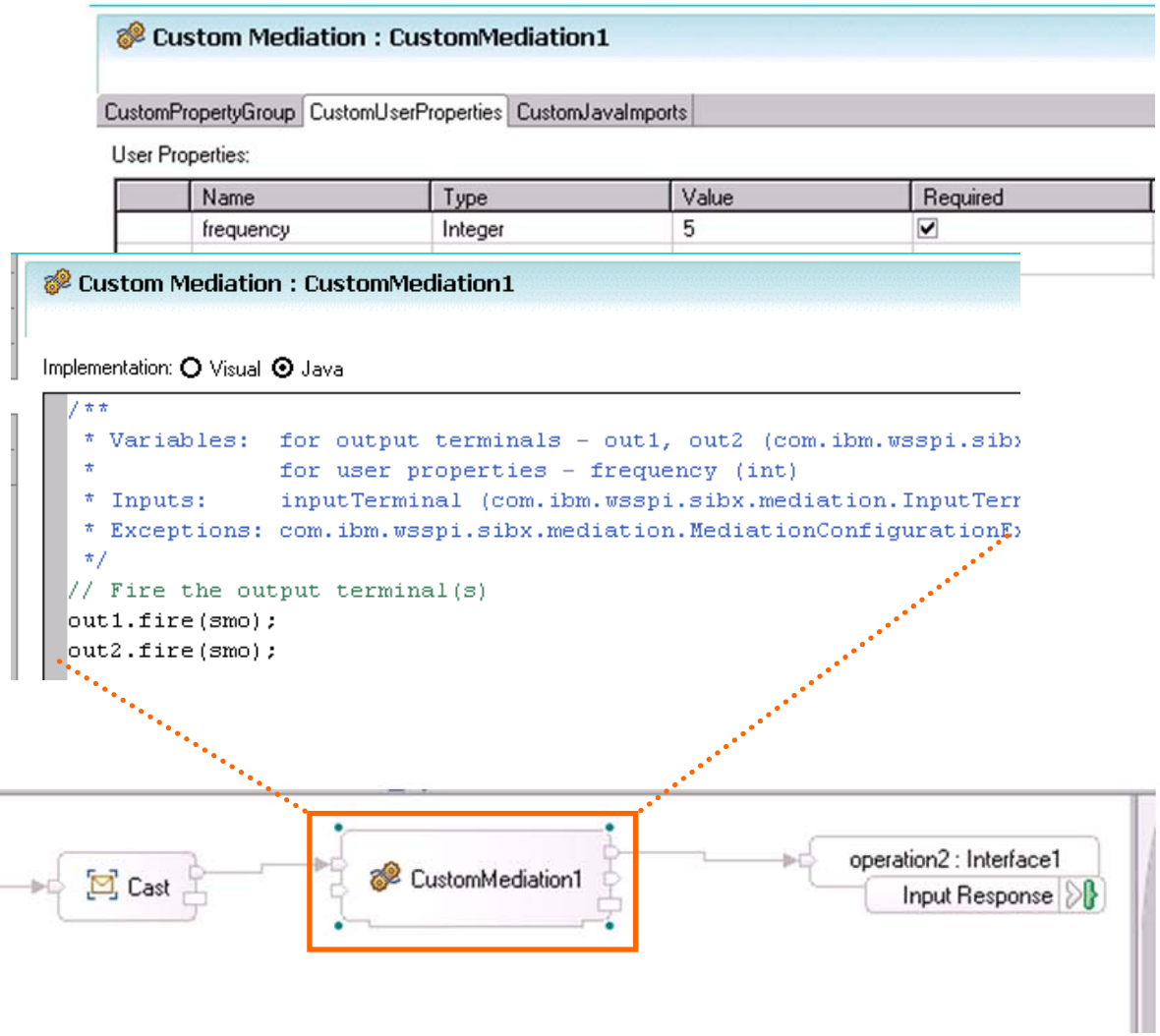
Splitting and Aggregating - example



WESB Enhancements

Custom Mediation Primitive

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



Summary and Conclusion

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


Thank
You