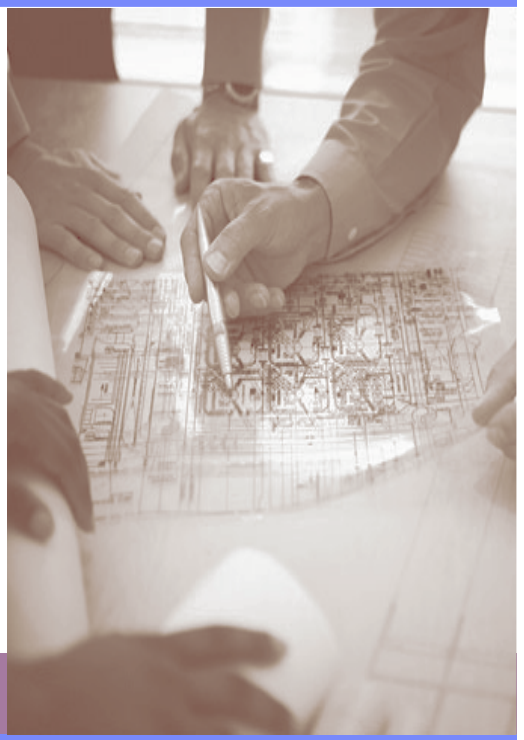
IBM Software Group



SCA-based Enterprise Service Bus
WebSphere ESB

WebSphere software

Soudabeh Javadi, WebSphere Software
IBM Canada Ltd
sjavadi@ca.ibm.com

Agenda

- **What is WebSphere ESB**
- **WebSphere ESB Functions**
- **Positioning**

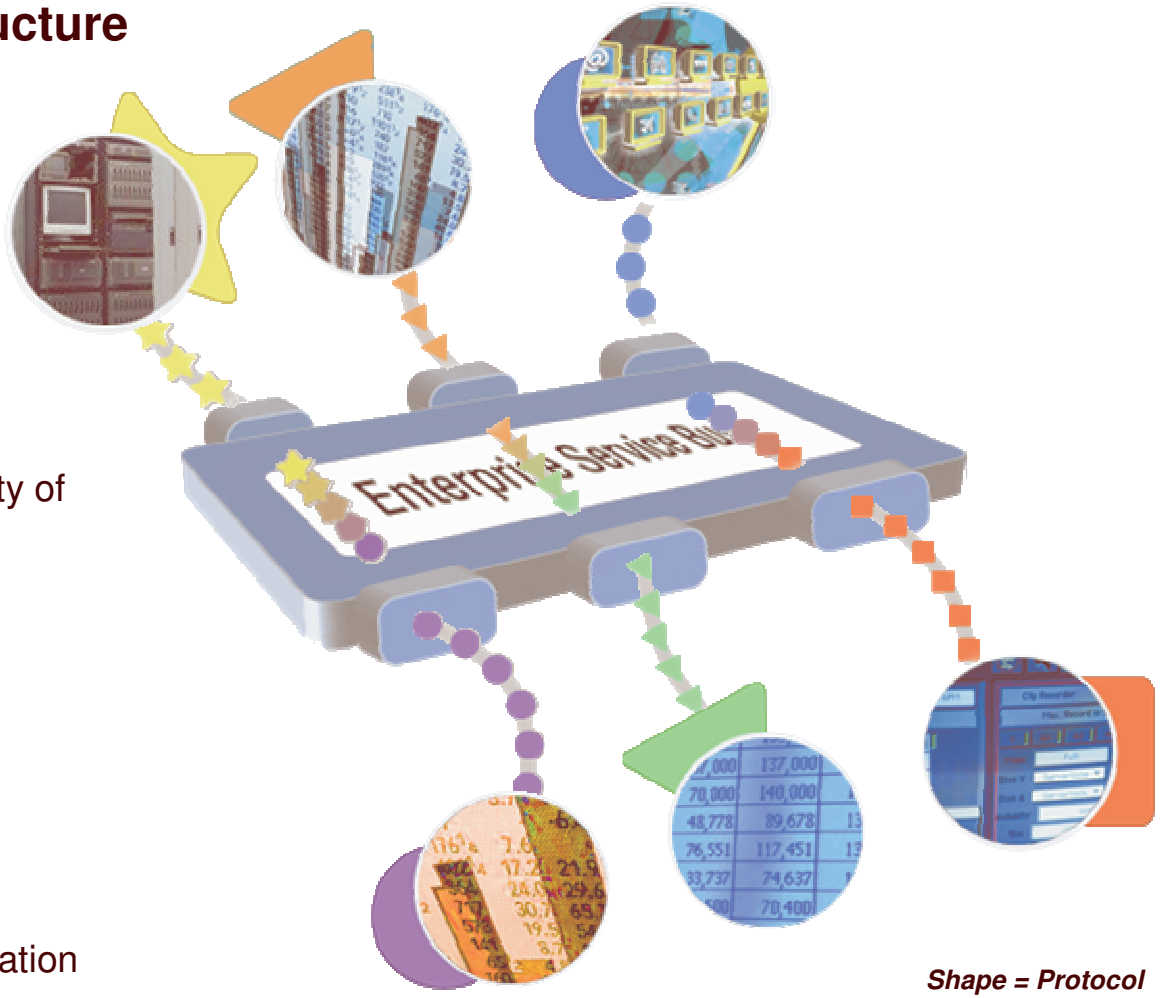
What is an Enterprise Service Bus (ESB)?

A flexible connectivity infrastructure for integrating applications as services…

……which reduces the number, size, and complexity of interfaces.

An ESB:

- **VIRTUALIZES** the location and identity of participants
- **CONVERTS** between different transport protocols used by the participants
- **TRANSFORMS** message formats between participants
- **APPLIES** appropriate qualities of service for the given interaction
- **DISTRIBUTES** business event information to/from disparate sources.

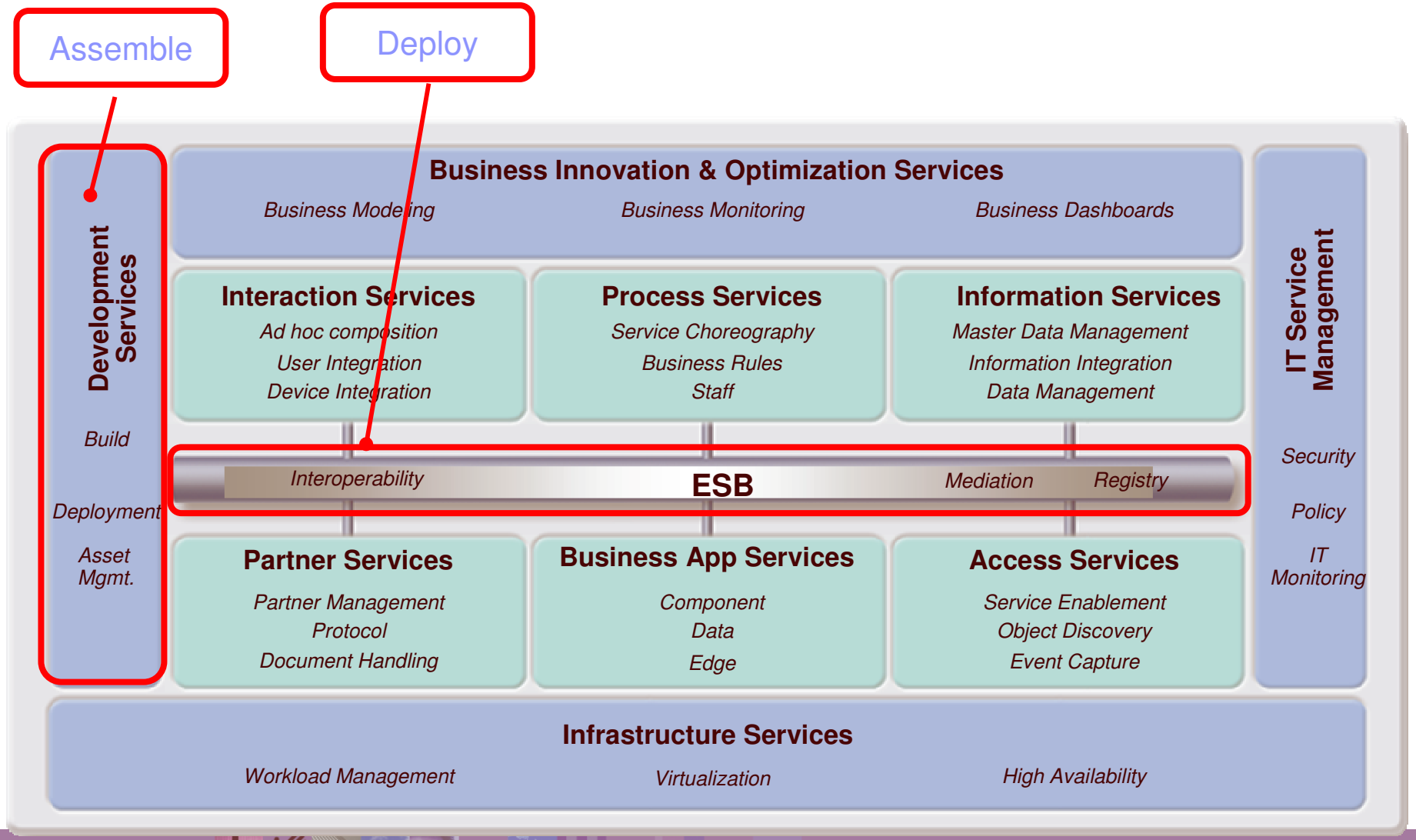


Shape = Protocol

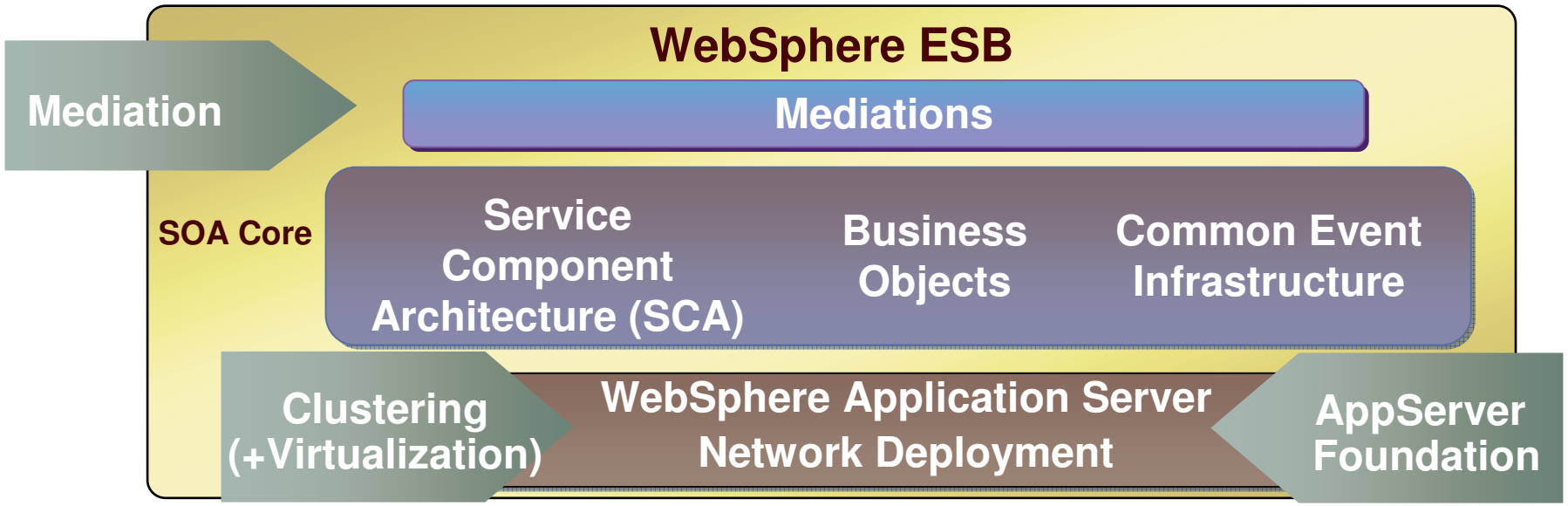
Color = Data type

SOA Reference Architecture

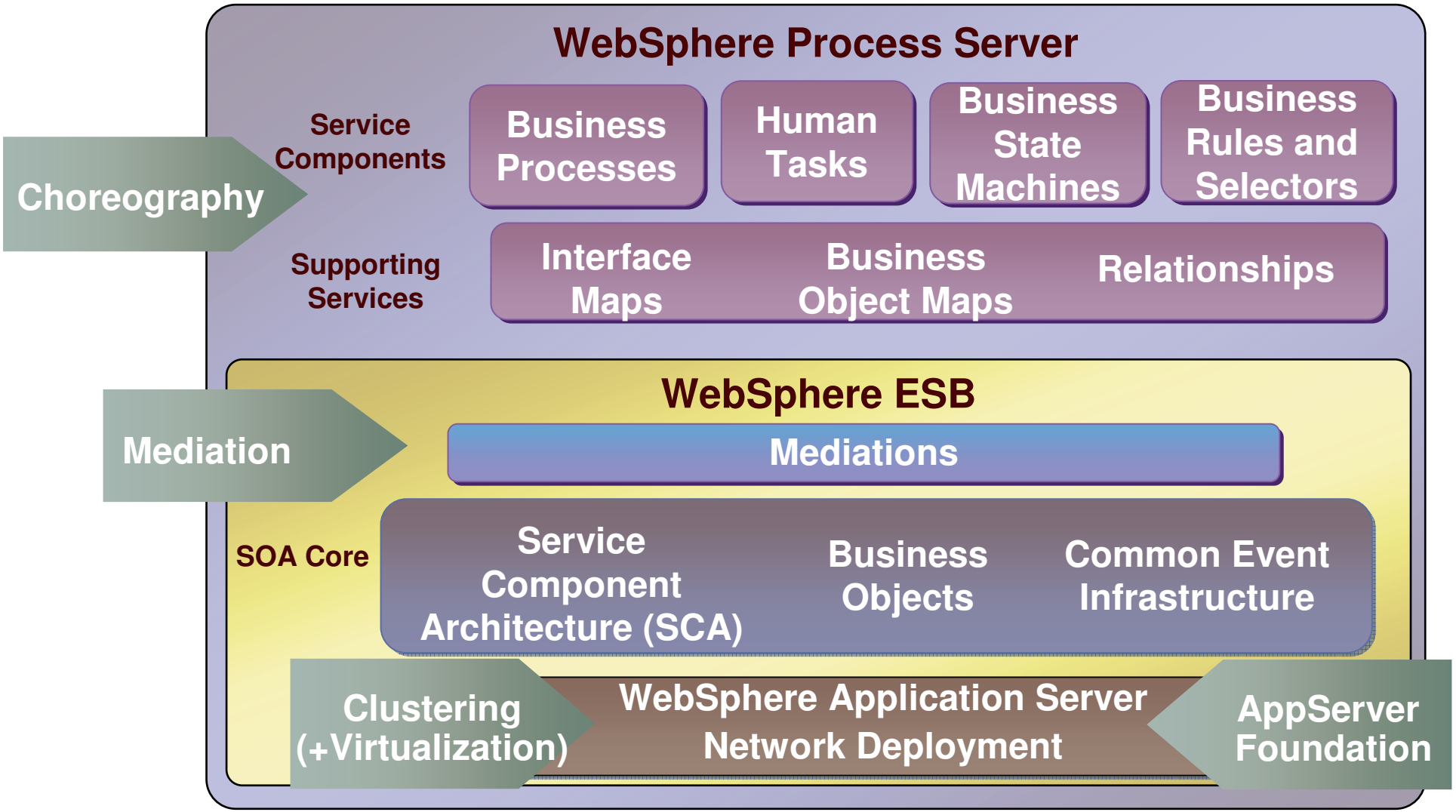
Comprehensive services in support of your SOA



WebSphere Enterprise Service Bus (WESB) Component Architecture



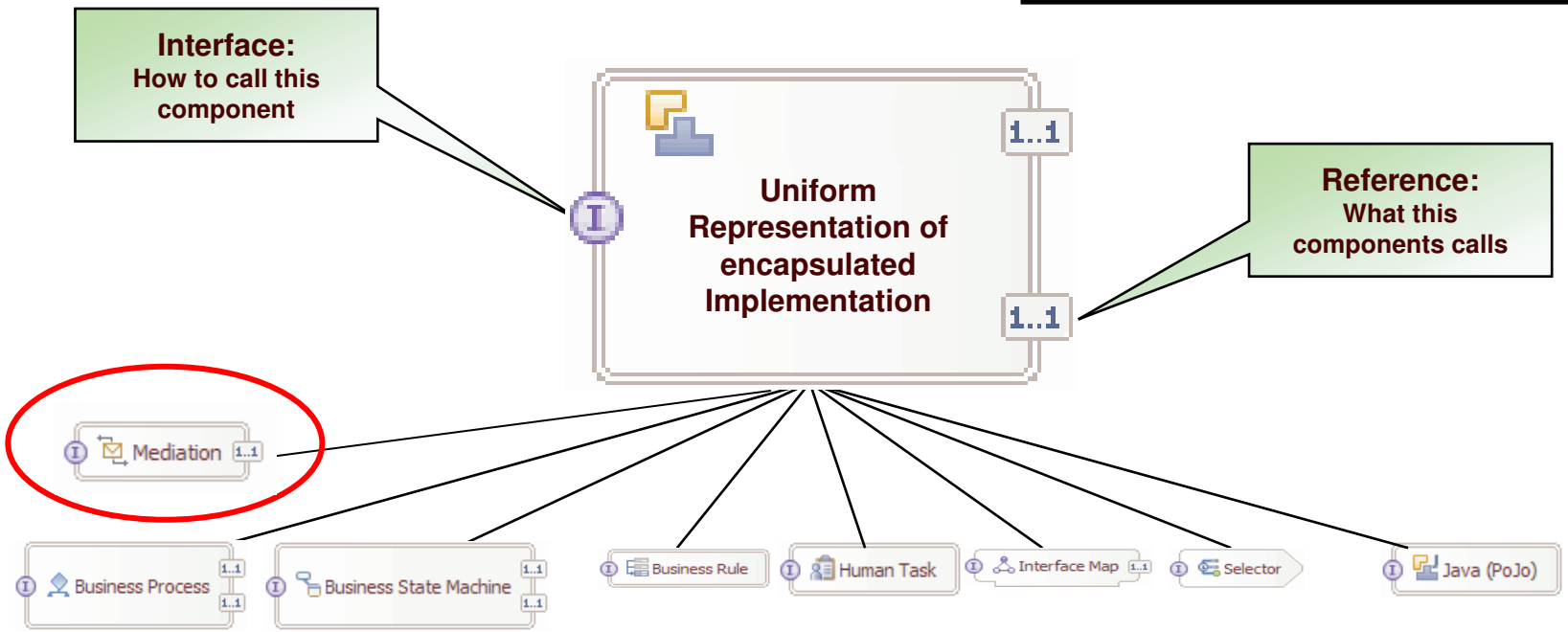
WESB – A Component of WebSphere Process Server



SOA: Common Invocation Model

Service Component Architecture

IBM, along with BEA, Oracle, SAP, IONA, Siebel and Sybase have announced the new specifications for SCA

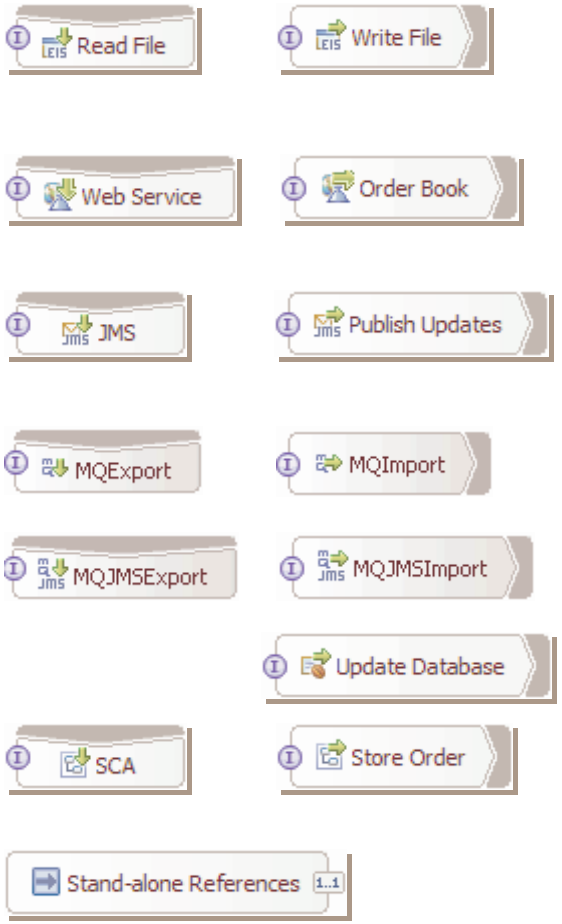


Business Value

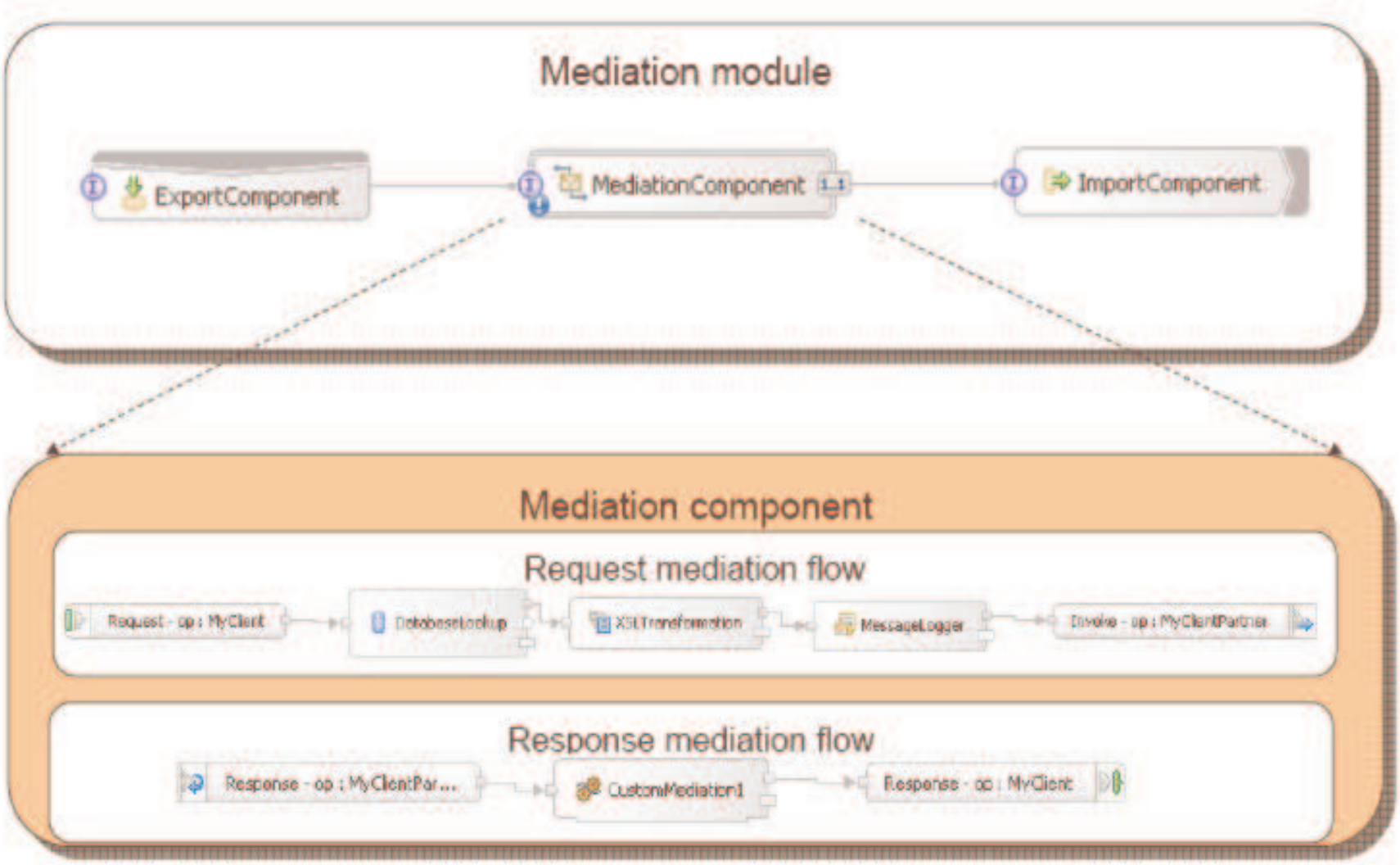
- **Encapsulate components for reuse**
  - Service Components are wired together to form deployable solutions
  - Business Objects are the data flowing between Service Components
- **All components (e.g., services, rules, human interactions) are represented consistently and invoked identically - encapsulation and reuse will reduce development costs**
- **Increased productivity, reduced cost**

SOA: Common Invocation Model
Imports and Exports

- **WebSphere Adapters**
  - JCA 1.5
  - WBI “Legacy” Adapters
- **Web Services**
  - SOAP over HTTP, SOAP over JMS
- **JMS (WebSphere Messaging Resources)**
  - Point-to-Point and Publish/Subscribe
- **MQ**
  - MQ native
  - MQ/JMS (MQ-JMS Provider)
- **EJB (Session Beans)**
- **SCA**
  - Connect modules to each other without exposing the interface outside of WebSphere Process Server
- **Standalone Reference**
  - Enables an SCA API Client to call a Module



WESB SCA Concepts

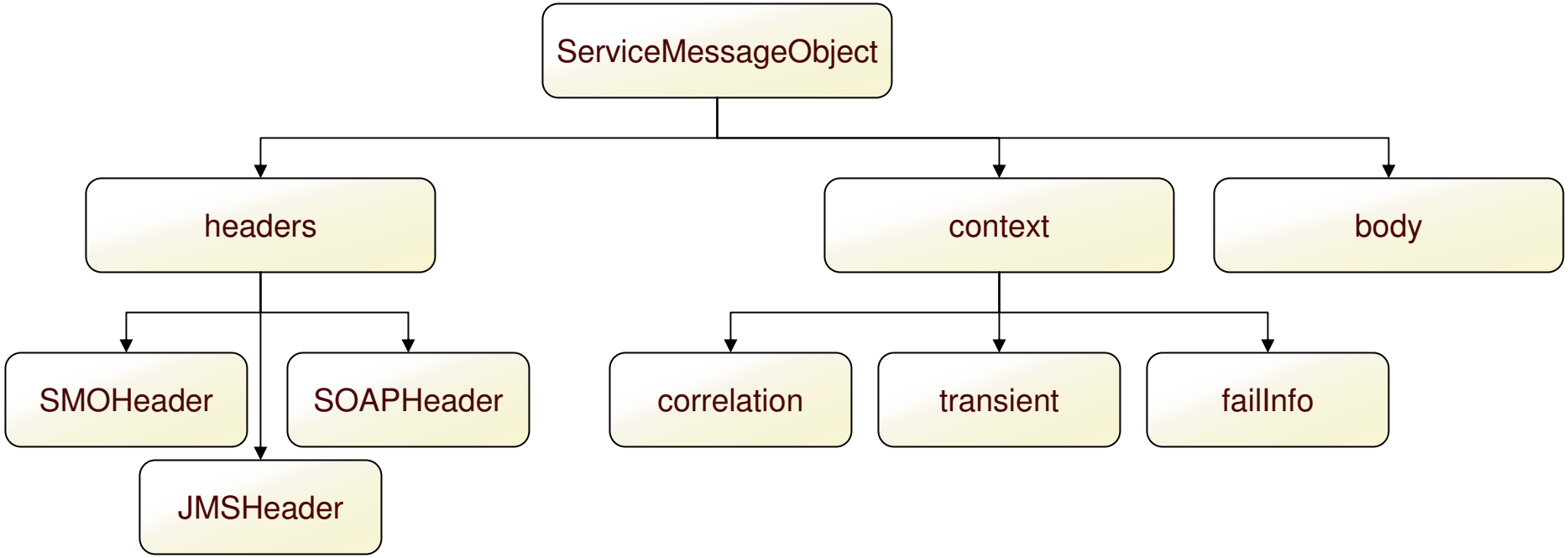


Mediation Primitives

Primitive	Symbol	Description
Message logger		Log/store message information to a database.
Message filter		Provides content-based routing.
DB lookup		Access info in a database and store it in the SMO.
XSL transform		Transform messages using XSL transformation.
Registry lookup	6.0.2	Select an endpoint via registry lookup.
Event emitter	6.0.2	Emit CBE events.
Element setter	6.0.2	Allows in-line updates to SMO elements.
Stop		Stop a path in the flow - no exception.
Fail		Stop a path in the flow - generate an exception.
Custom		For custom processing of message. It uses a SCA Java component for custom message processing.
(build your own)		Write it using SCA Java APIs (similar to custom).

WESB Message Model - Service Message Object

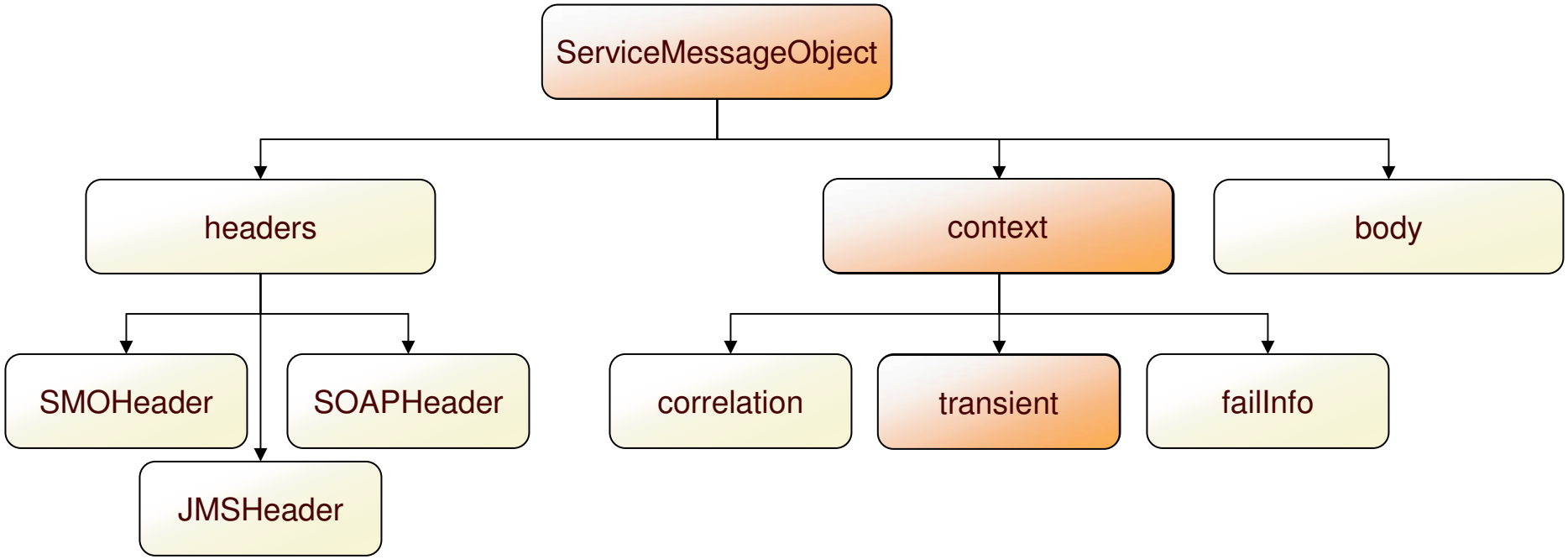
- Mediation primitives process messages as SMOs
- The SMO is an extension of SDO
- It contains: context, header, fault, and body information



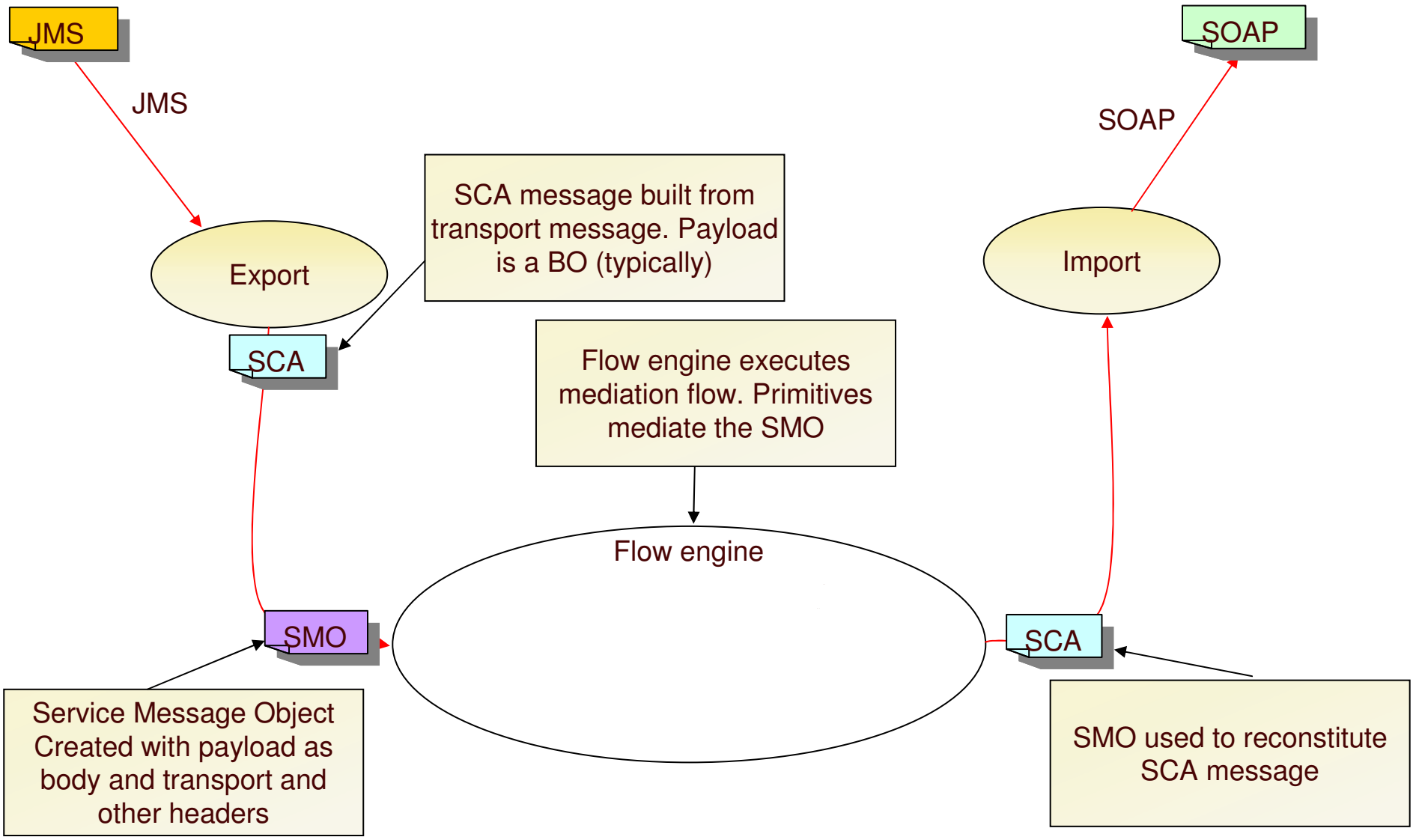
Service Message Object

- Addressable by XPath: /context/transient
- Accessible by API (com.ibm.websphere.sibx.smo.*)

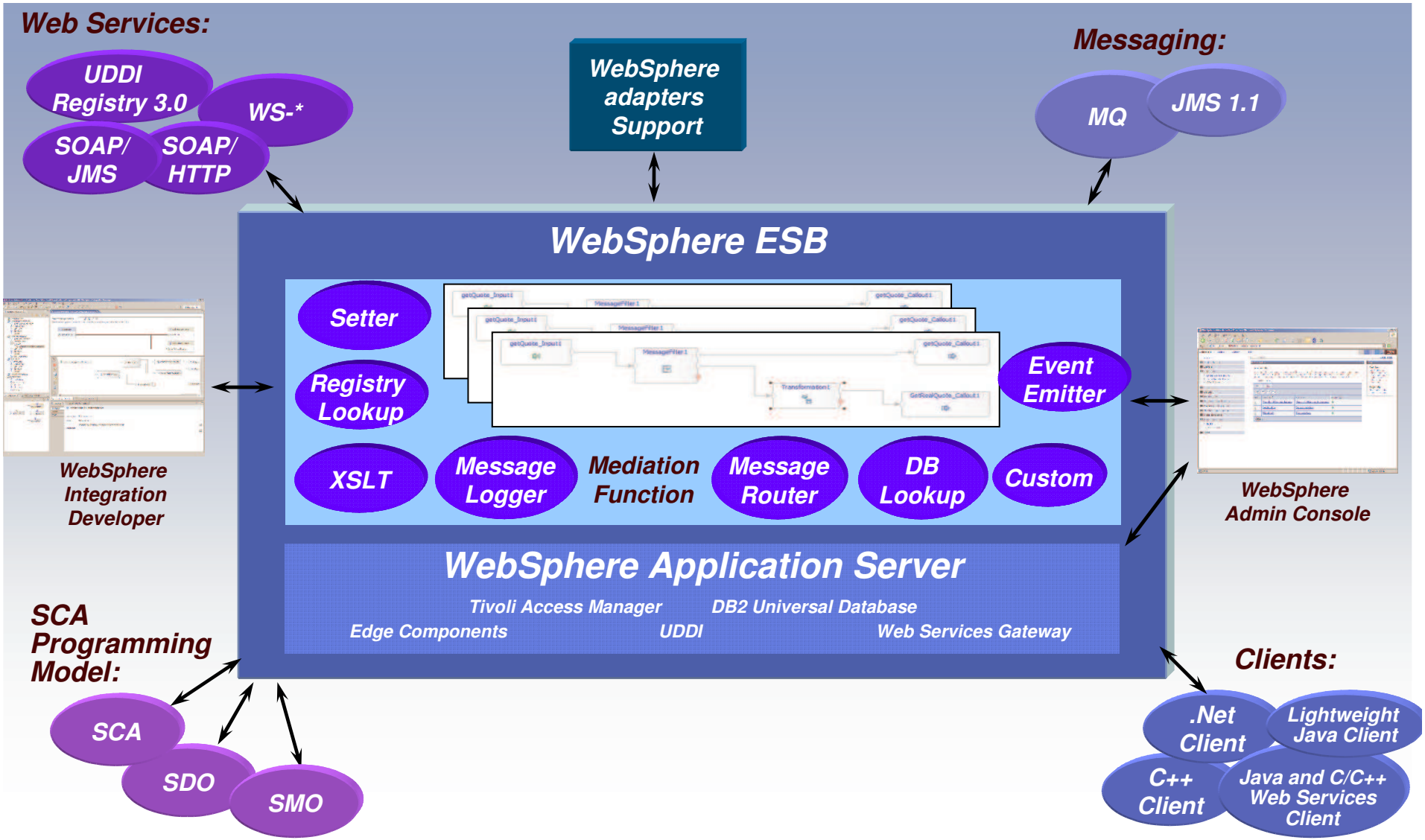
```
ServiceMessageObject smo = (ServiceMessageObject)a_type;
DataObject transient = smo.getContext().getTransientContext();
```



Invocation of a mediation flow



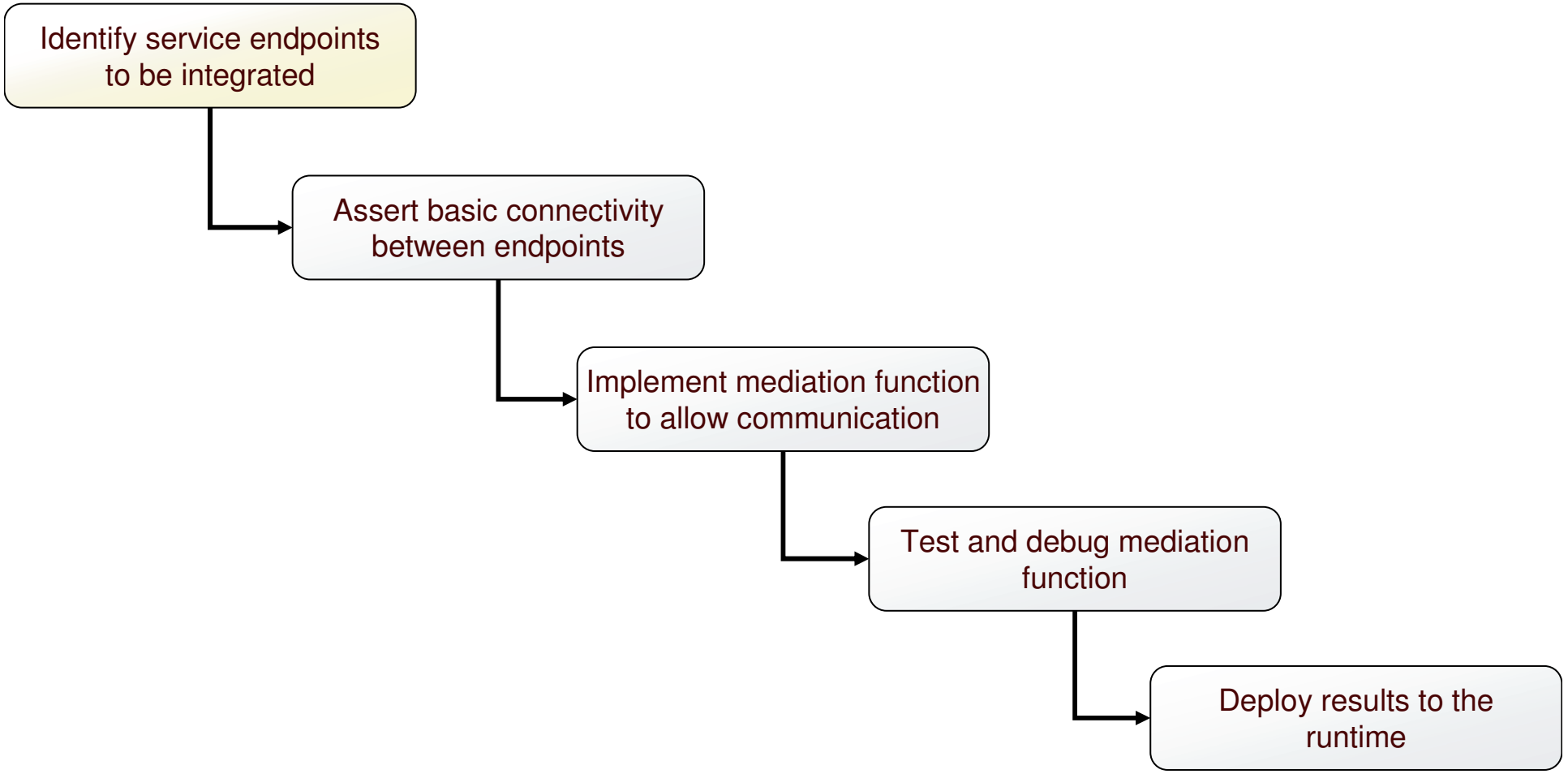
WebSphere ESB – At a Glance



Integration Developer: Typical Task Flow

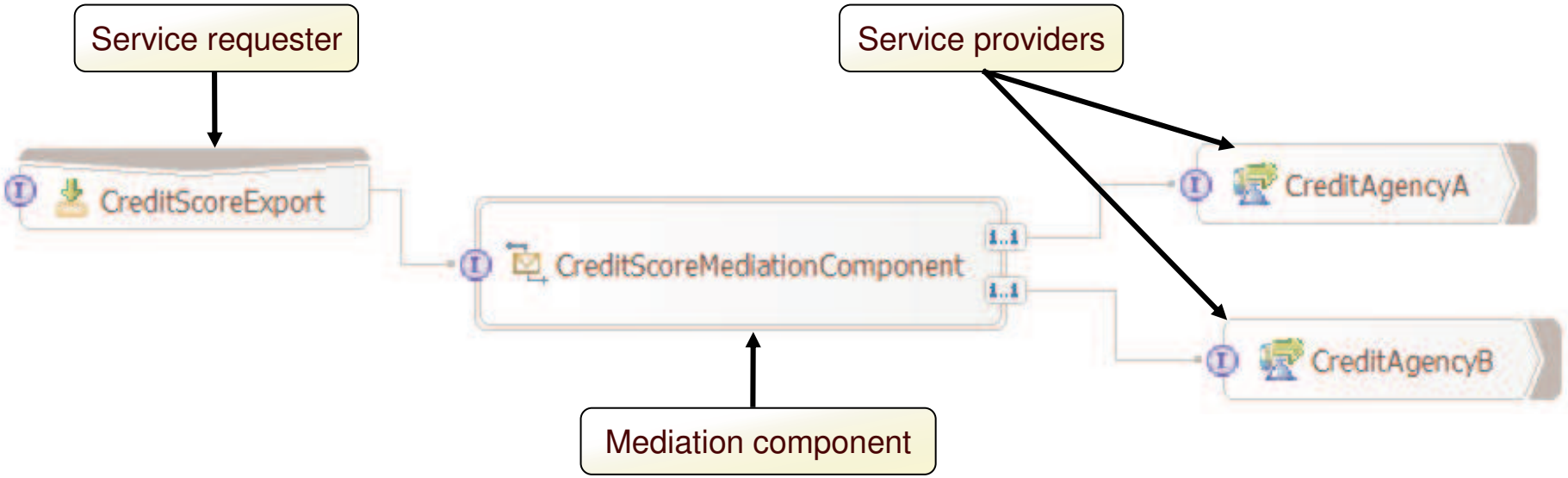
1. **Identifies the service endpoints that need to be integrated**
   - Service requesters and Service providers
2. **Asserts the basic connectivity between these endpoints**
   - Which requester operation is linked to which provider operation
3. **Implement the mediation function required to allow endpoints to communicate effectively**
   - Selects from supplied function
   - Customizes selected function
   - Optionally: constructs and integrates custom-written function
4. **Test and Debug mediation function**
5. **Deploy to the runtime**

Typical Integration Developer task flow

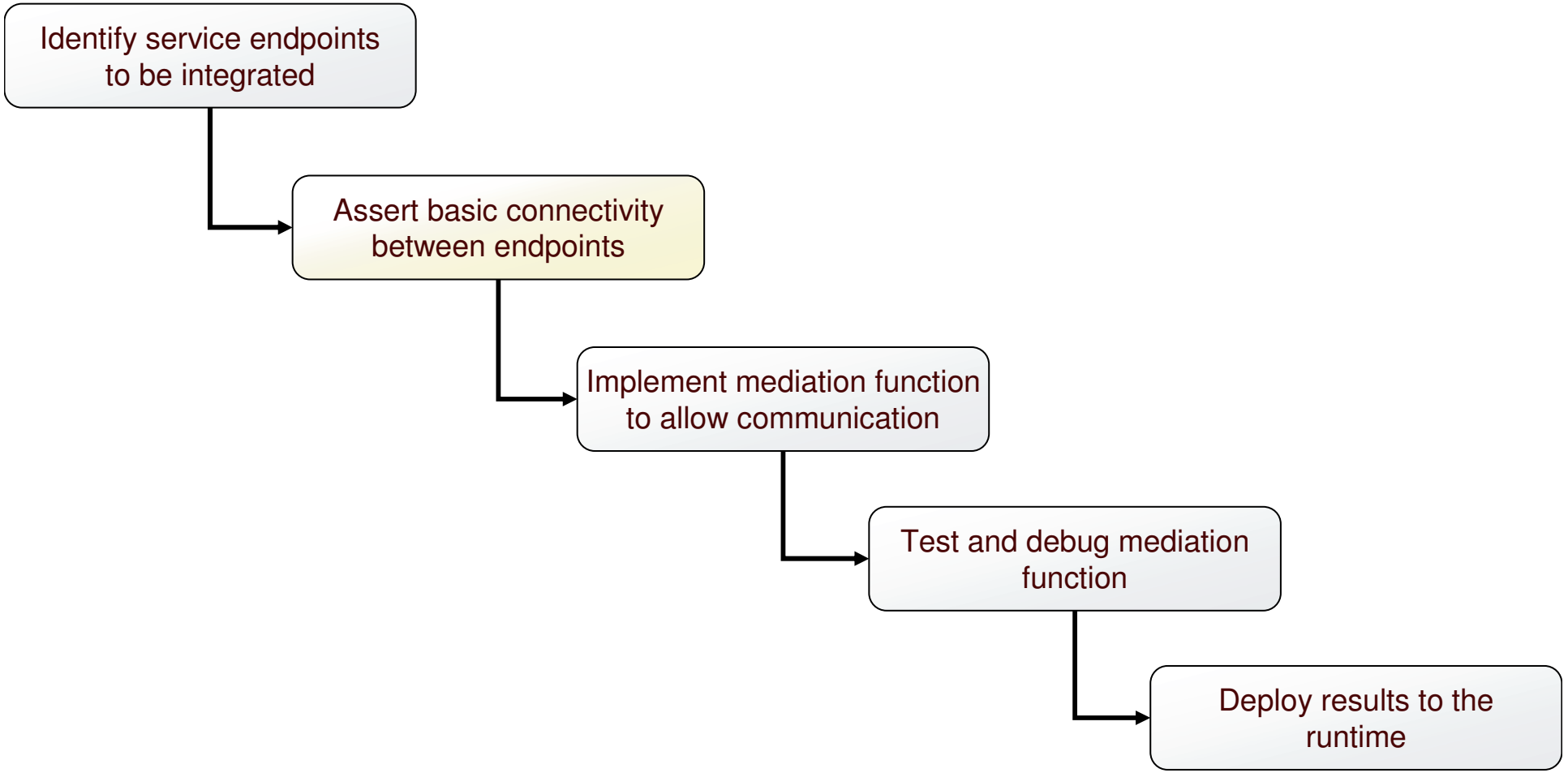


Identify service endpoints to be integrated

- Concept
  - The mediation module defines a mediation component and the endpoints it mediates in the form of imports and exports
- Task
  - Define mediation module and component
  - Define imports for service providers
  - Define exports for service requesters

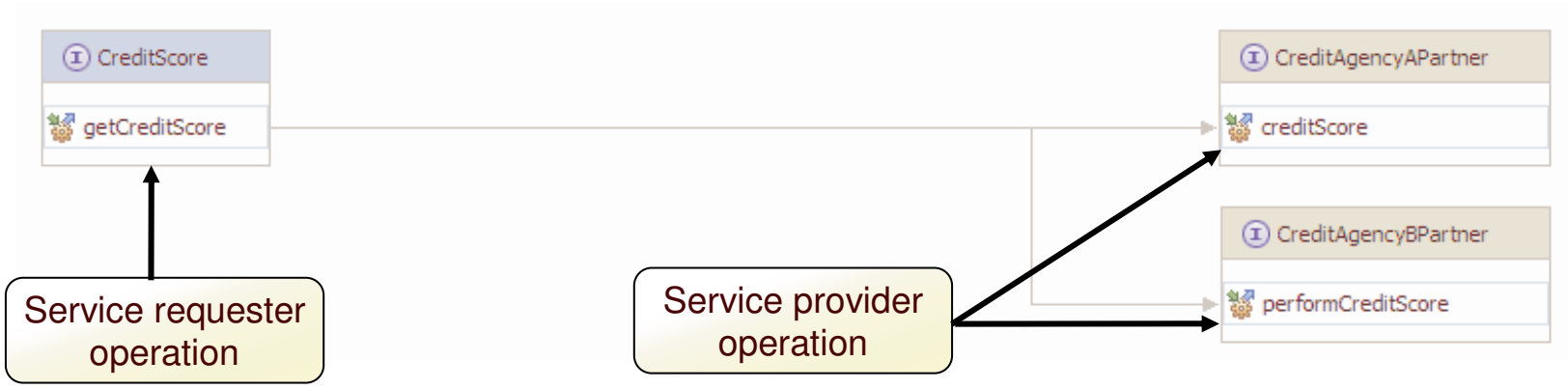


Typical Integration Developer task flow

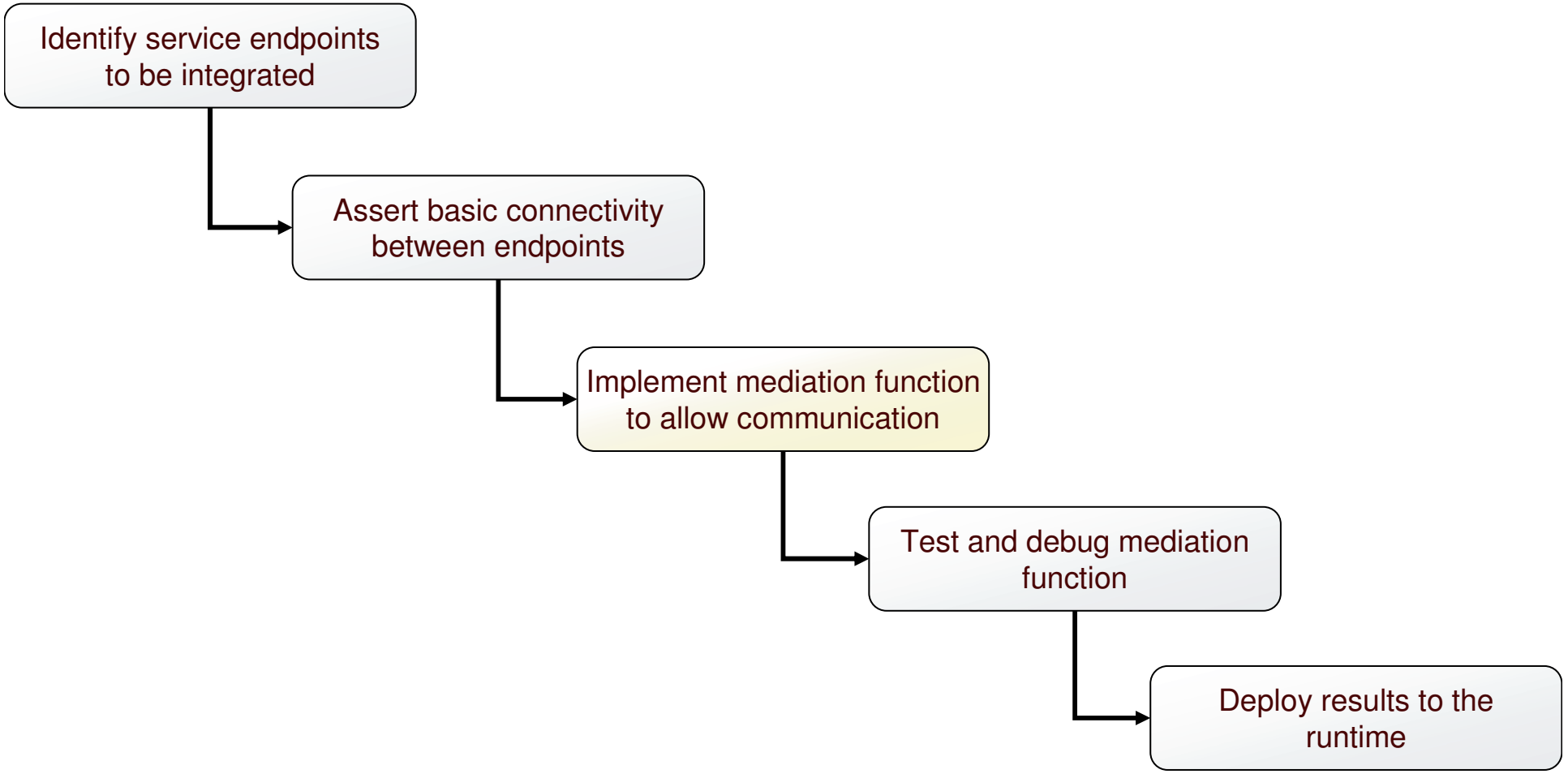


Assert the basic connectivity between endpoints

- Concept
  - Operation connections define links between service requester interfaces and service provider interfaces
  - Define paths along which mediation can occur
- Task
  - Identify paths between, and connect requester operations to provider operations

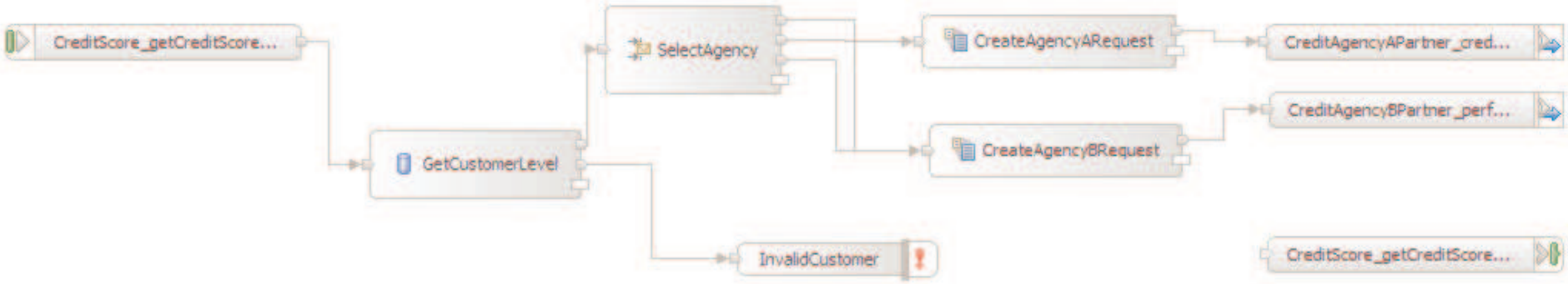


Typical Integration Developer task flow

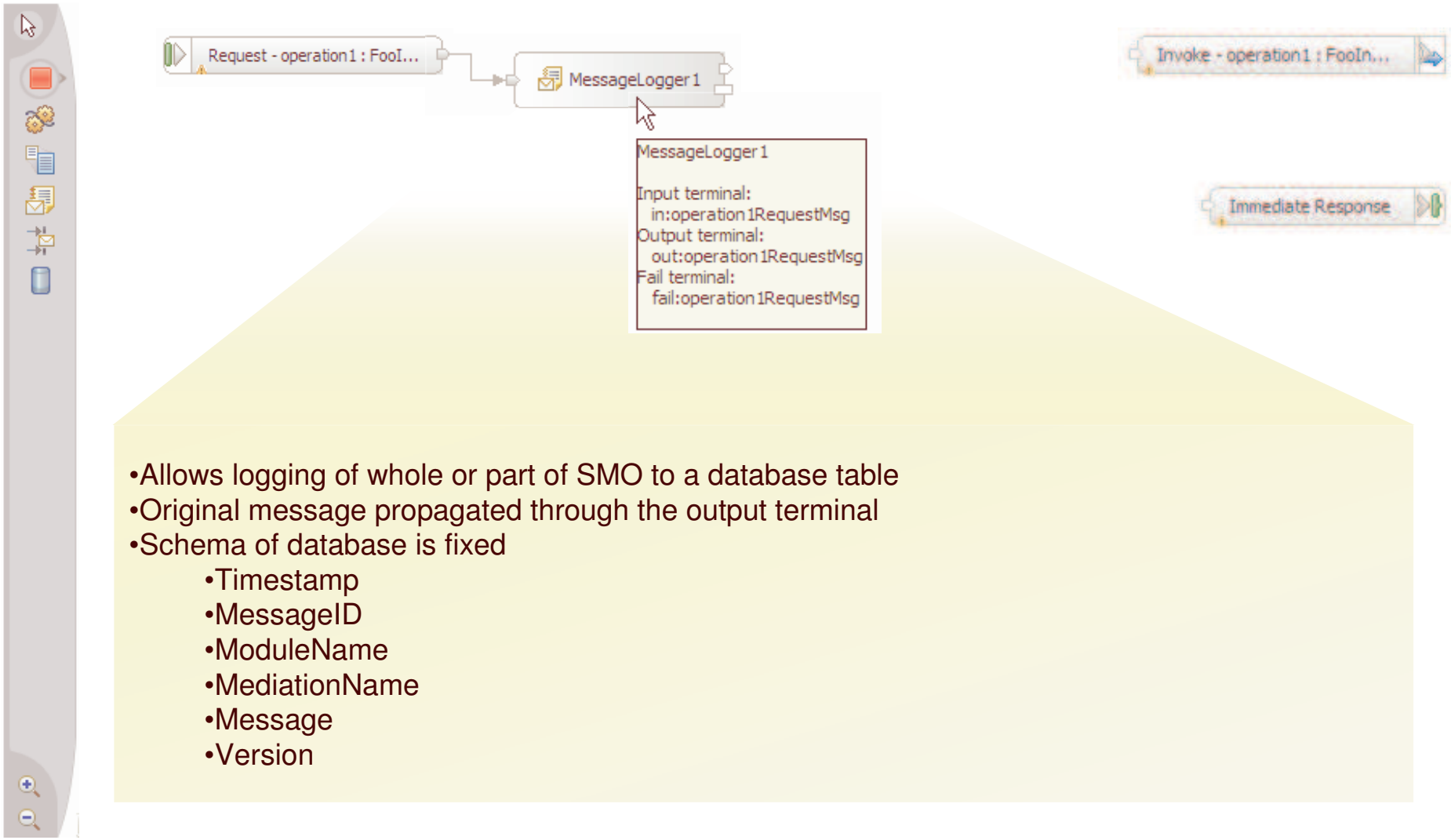


Implement mediation function to allow communication

- Concept
  - Mediate message using supplied primitives and/or by implementing custom mediations
  - Mediate the message in the form of a Service Message Object
- Task
  - Construct a mediation flow by selecting and connecting mediation primitives together

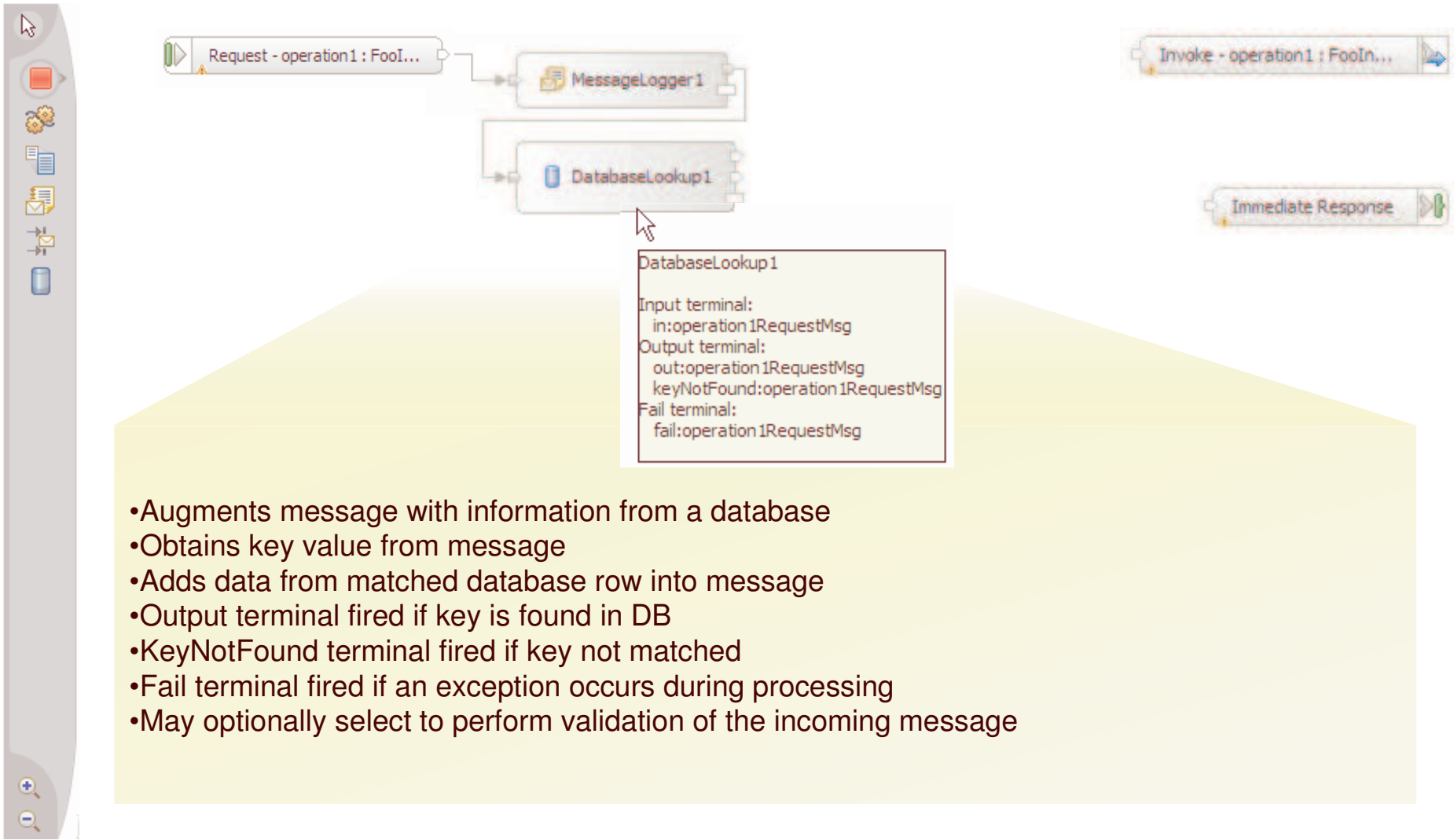


Mediation primitives – Message Logger



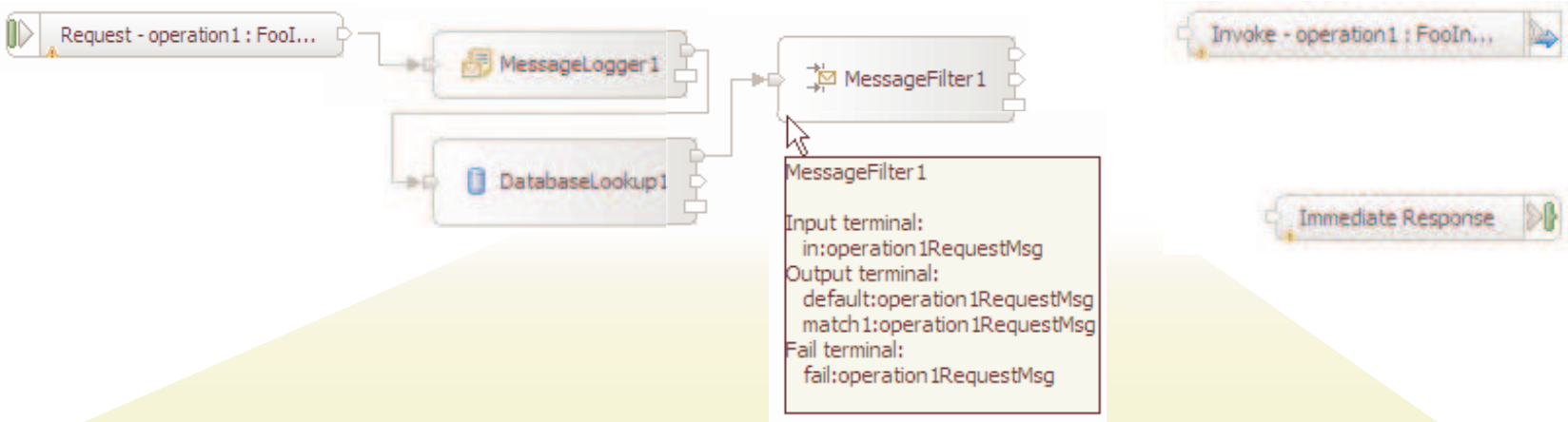
- Allows logging of whole or part of SMO to a database table
- Original message propagated through the output terminal
- Schema of database is fixed
  - Timestamp
  - MessageID
  - ModuleName
  - MediationName
  - Message
  - Version

Mediation primitives – DB Lookup



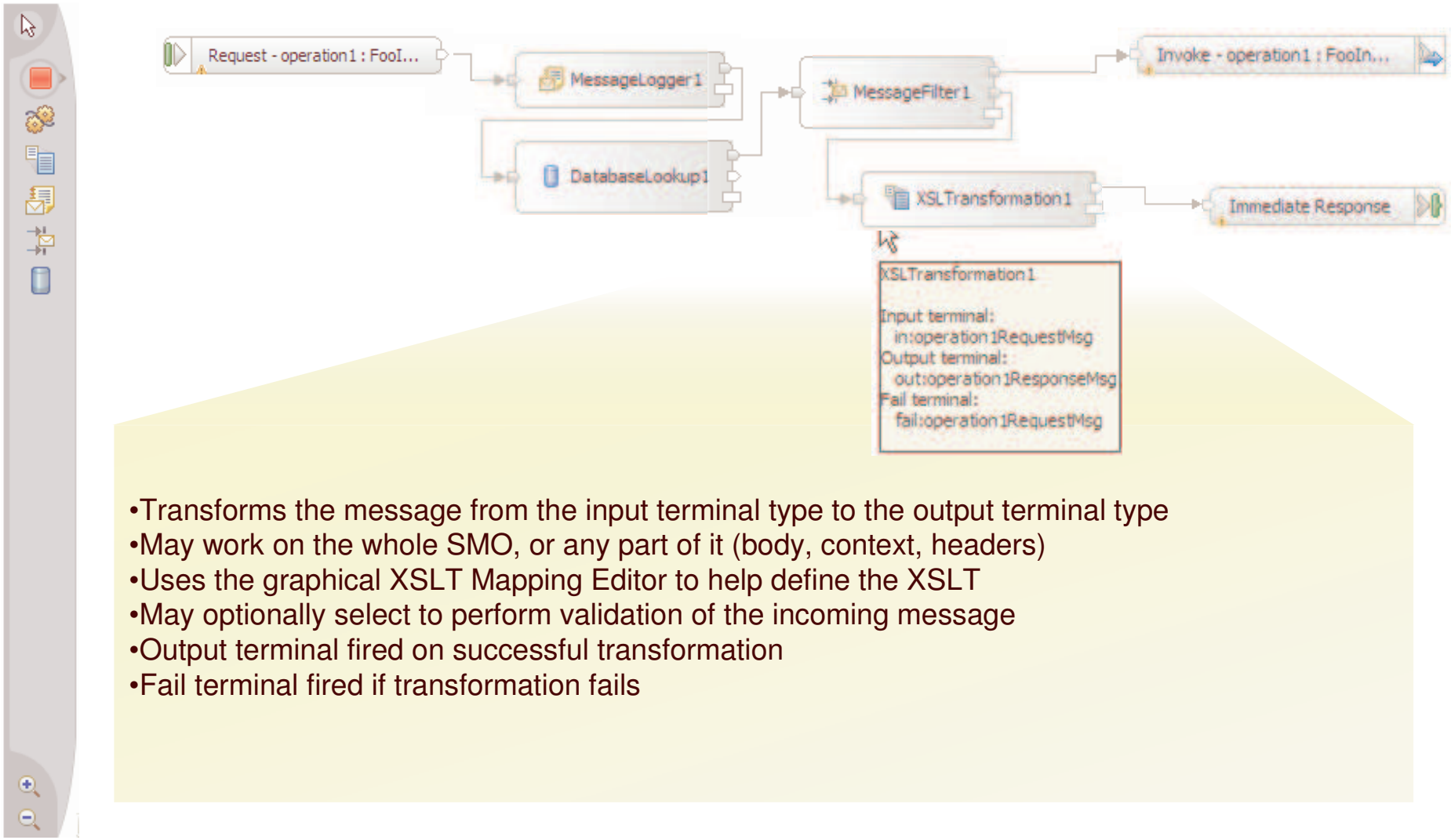
- Augments message with information from a database
- Obtains key value from message
- Adds data from matched database row into message
- Output terminal fired if key is found in DB
- KeyNotFound terminal fired if key not matched
- Fail terminal fired if an exception occurs during processing
- May optionally select to perform validation of the incoming message

Mediation primitives – Message Filter



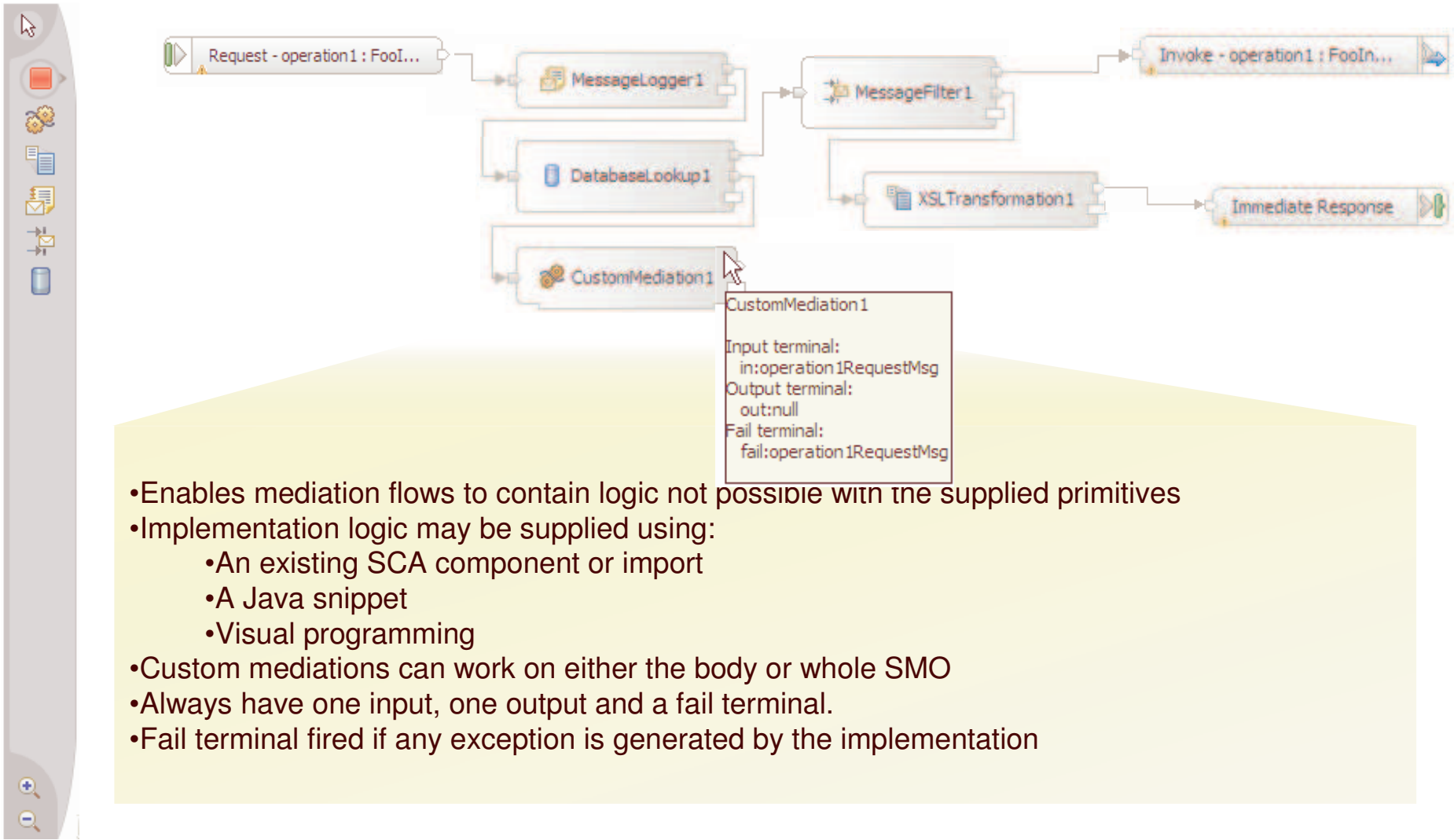
- Allows messages to be routed along different paths depending on contents
- Has a dynamic number of output terminals
- Allows filters to be specified which determine
  - An XPath pattern to match in the message
  - Which terminal to fire if the pattern matches
- Can fire either the first filter to match, or all matching filters
- If no filters match the default terminal is fired
- If an exception occurs during processing the fail terminal is fired

Mediation primitives – XSLT



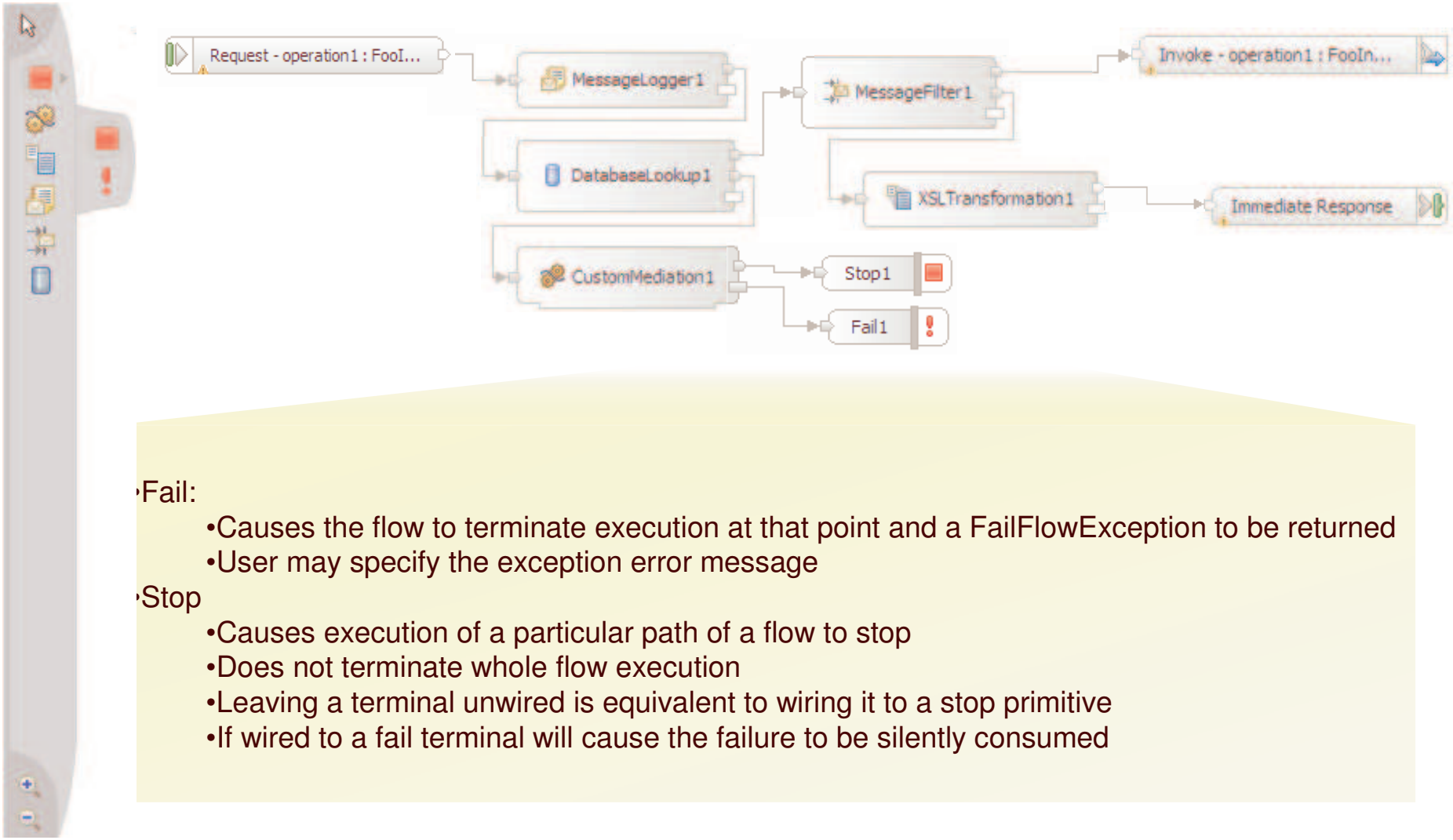
- Transforms the message from the input terminal type to the output terminal type
- May work on the whole SMO, or any part of it (body, context, headers)
- Uses the graphical XSLT Mapping Editor to help define the XSLT
- May optionally select to perform validation of the incoming message
- Output terminal fired on successful transformation
- Fail terminal fired if transformation fails

Mediation primitives – Custom



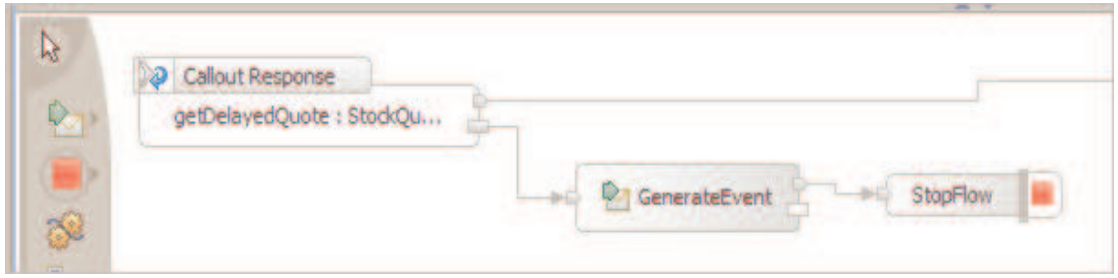
- Enables mediation flows to contain logic not possible with the supplied primitives
- Implementation logic may be supplied using:
  - An existing SCA component or import
  - A Java snippet
  - Visual programming
- Custom mediations can work on either the body or whole SMO
- Always have one input, one output and a fail terminal.
- Fail terminal fired if any exception is generated by the implementation

Mediation primitives – Fail and Stop



- Fail:
  - Causes the flow to terminate execution at that point and a FailFlowException to be returned
  - User may specify the exception error message
- Stop
  - Causes execution of a particular path of a flow to stop
  - Does not terminate whole flow execution
  - Leaving a terminal unwired is equivalent to wiring it to a stop primitive
  - If wired to a fail terminal will cause the failure to be silently consumed

Mediation primitives – Event Emitter

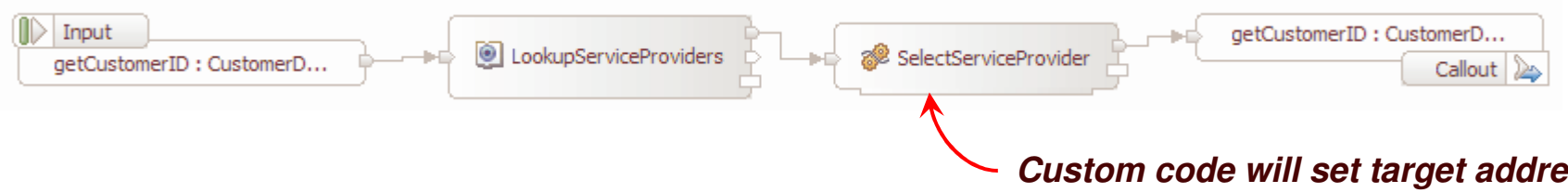


- Emits CBE events from within a Mediation Flow
- Enables reporting of significant events within the flow
- Fully integrated with Common Event Infrastructure (CEI)
  - Events contain all the common elements of CBEs used with CEI
  - Generated events are sent to CEI server, therefore:
    - Can be written to the event database
    - Can be forwarded using JMS topics or queues
  - Can be used by monitoring applications, for example:
    - CBE Browser
    - WebSphere Business Monitor
    - User written event processing application
- Events can be configured to contain a section of the SMO
  - Data objects in SMO are expanded to extended data elements

Mediation primitives – Message Element Setter

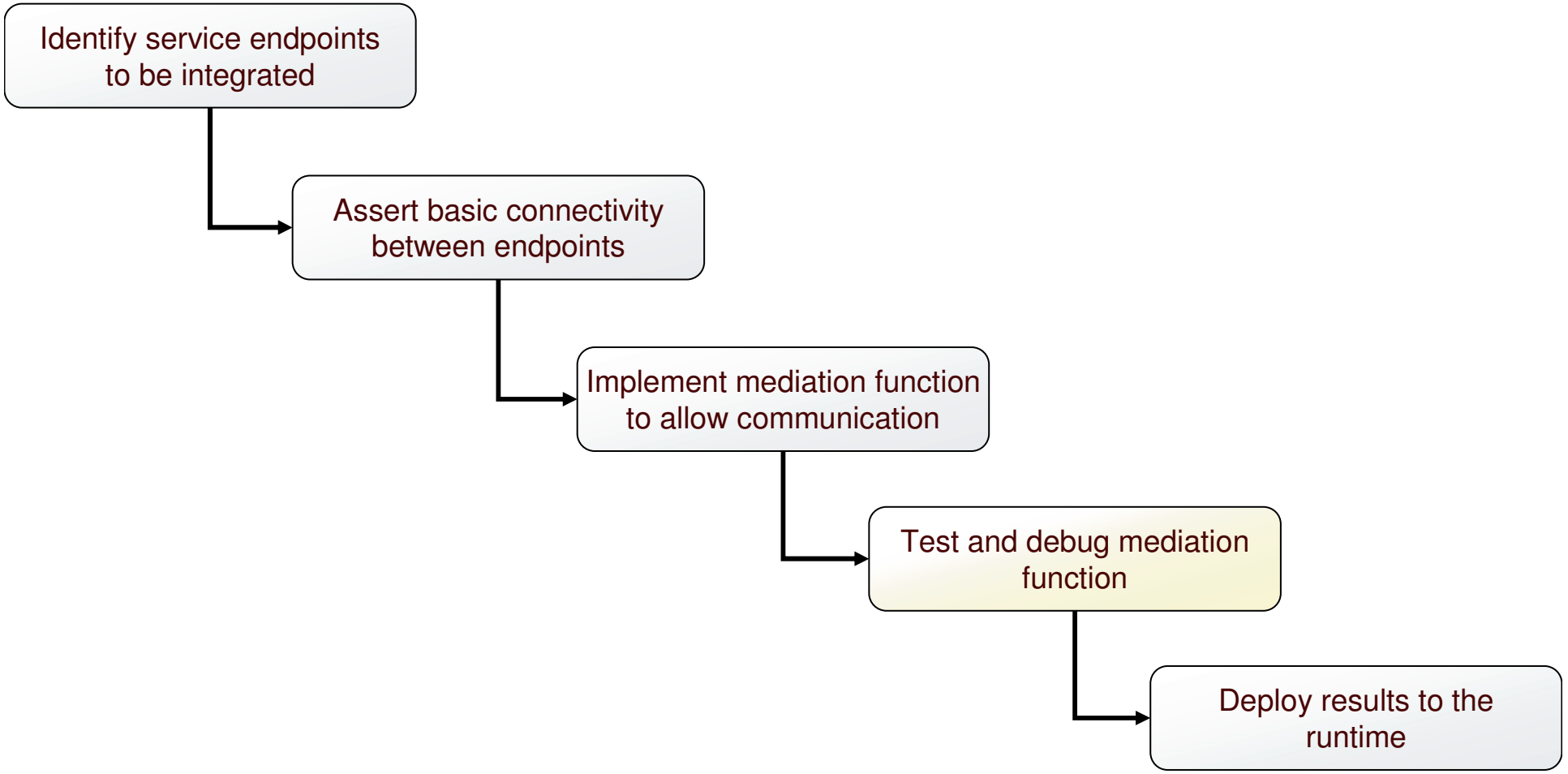
- Makes “in-place” updates to the SMO
  - Assignment of a constant value
  - Copying from one part of an SMO to another
    - Leaf nodes
    - Sub-trees
    - Source and target must match
  - Deleting elements
    - Setting the element value to “null”
- XPath expressions used to identify elements
  - Target elements
  - Source elements of a copy operation
- Multiple elements can be set within the same primitive

Mediation primitives – Endpoint Lookup

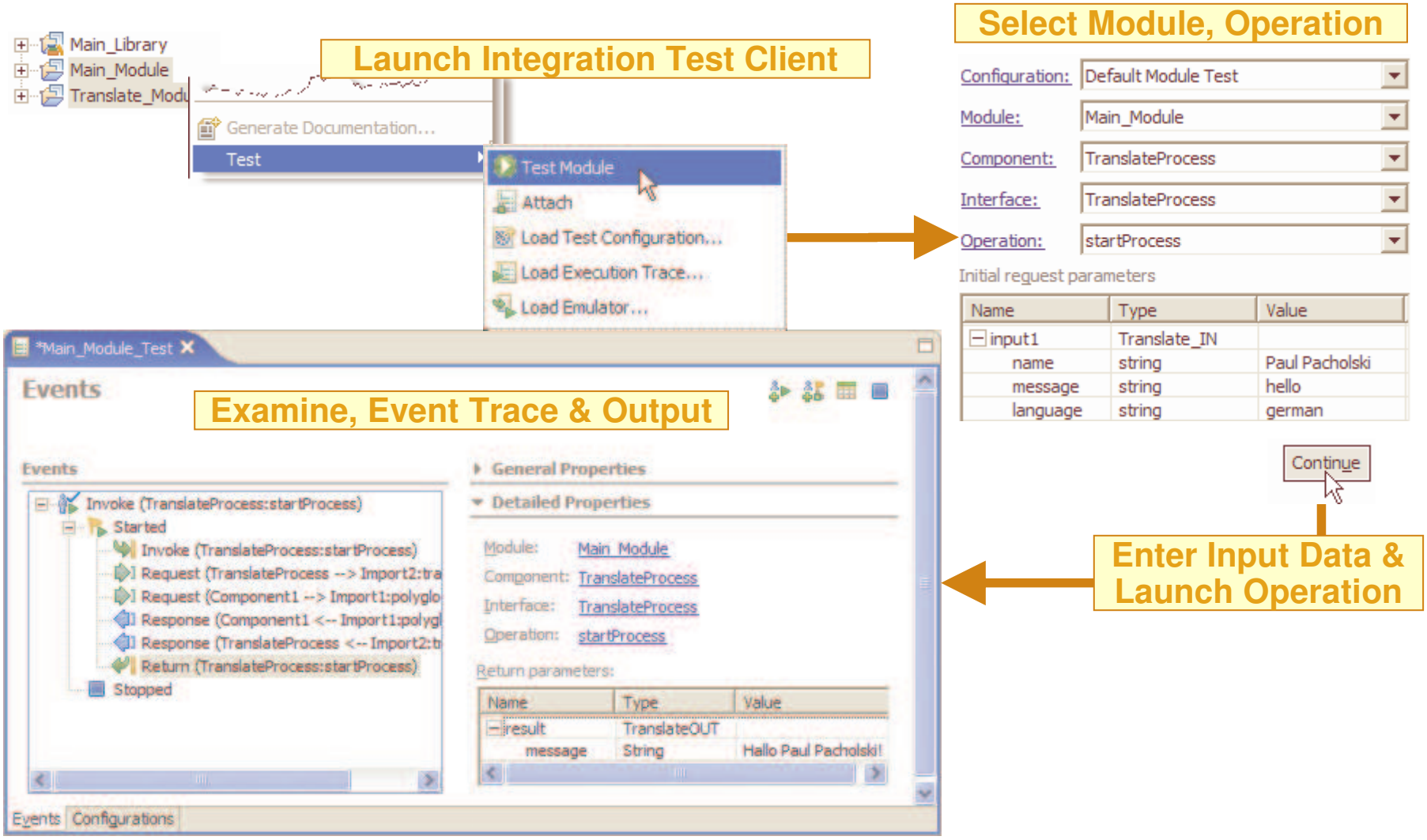


- New primitive uses a registry to find service endpoints
  - Performs the lookup based on selection criteria
  - Initializes SMO with results for downstream use by mediation flow
- Numerous criteria can be used for selection
  - Which registry to use for the lookup
    - Available registries are administratively defined within a WebSphere cell
  - Specifics of the requested service port type
    - Name
    - Namespace
    - Version
  - Associated classification (Based on OWL Web Ontology Language)
  - Associated properties and property values
  - Match policy
    - Defines if only one or all matching services should be returned

Typical Integration Developer task flow

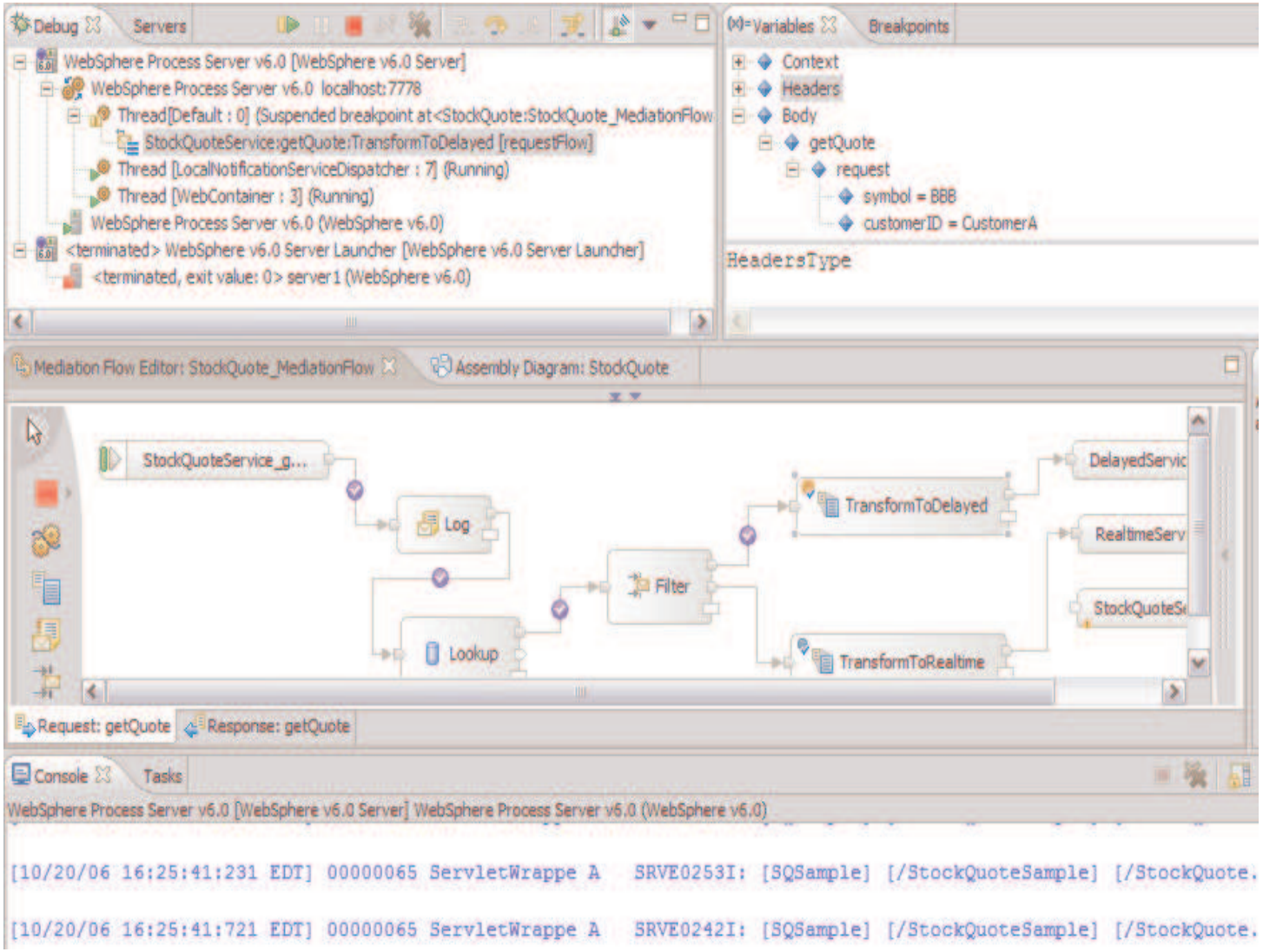


Test and Debug – Integration Test Client

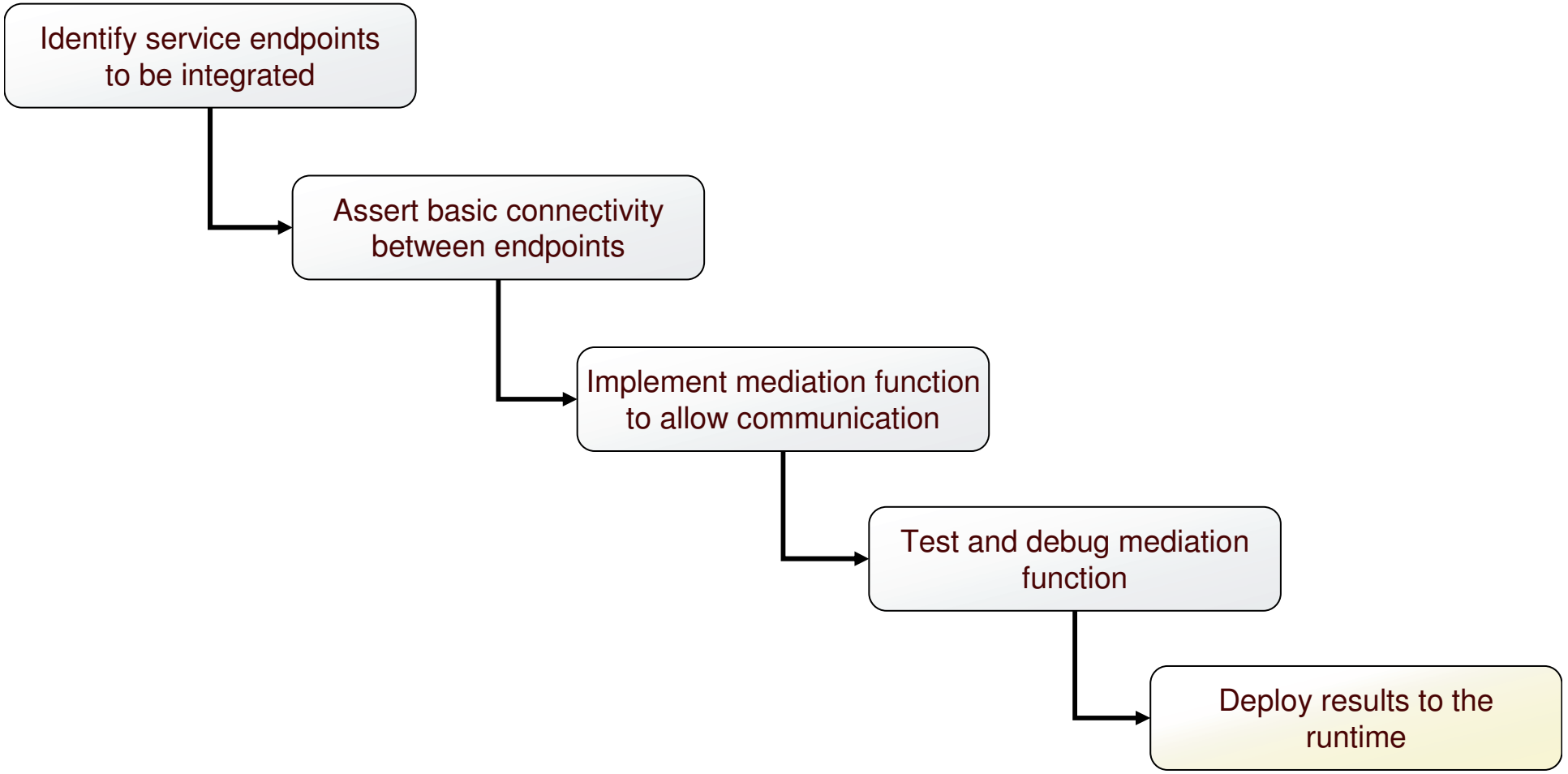


Test and Debug – Integration Debuggers

- Server must be started in the Debug Mode
- Debugger runs in the Debug Perspective
- Capabilities
  - Set breakpoints in a component
  - Step through the component
  - Change the values of its variables
  - Step into source code



Typical Integration Developer task flow

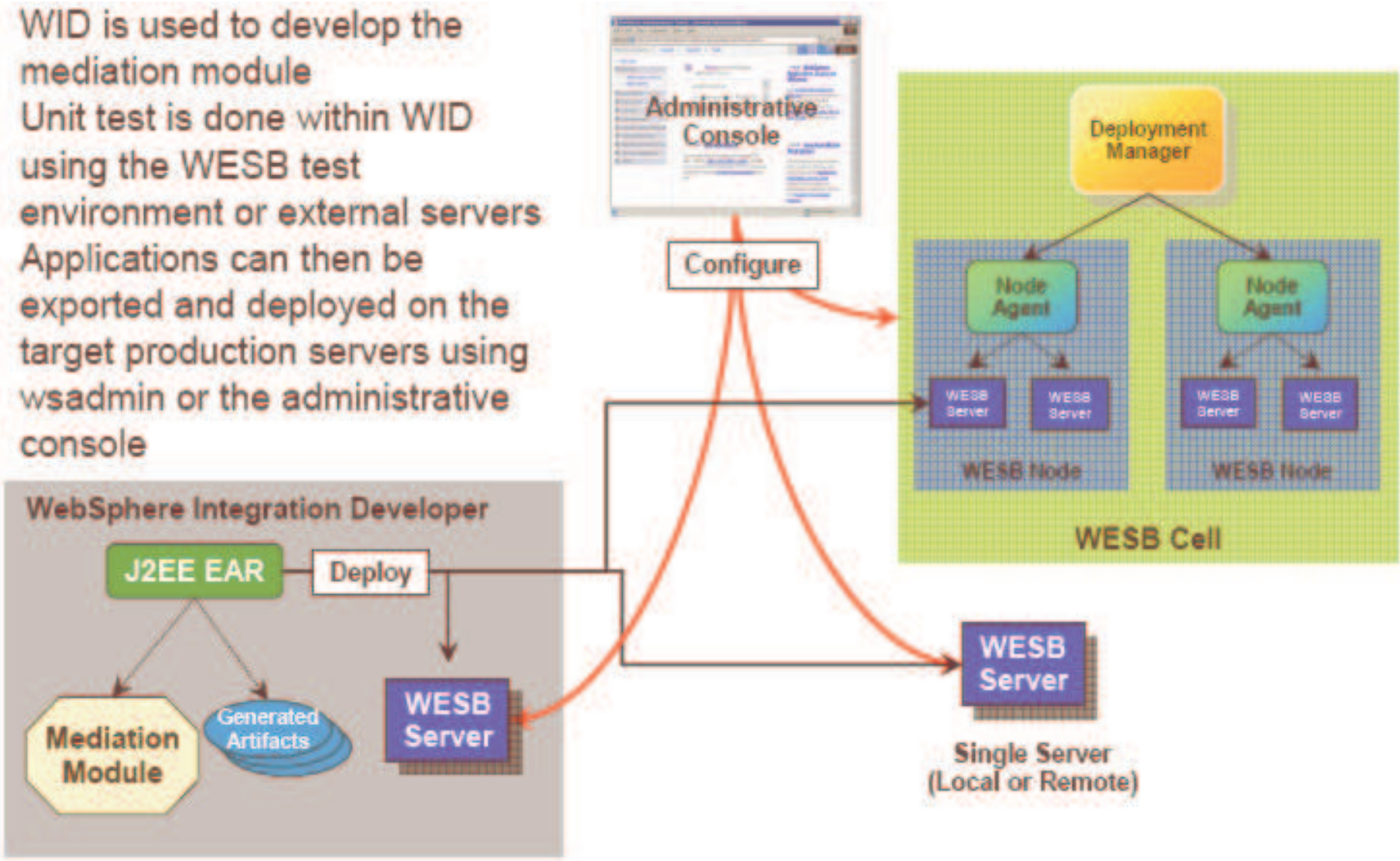


Deploy results to the runtime

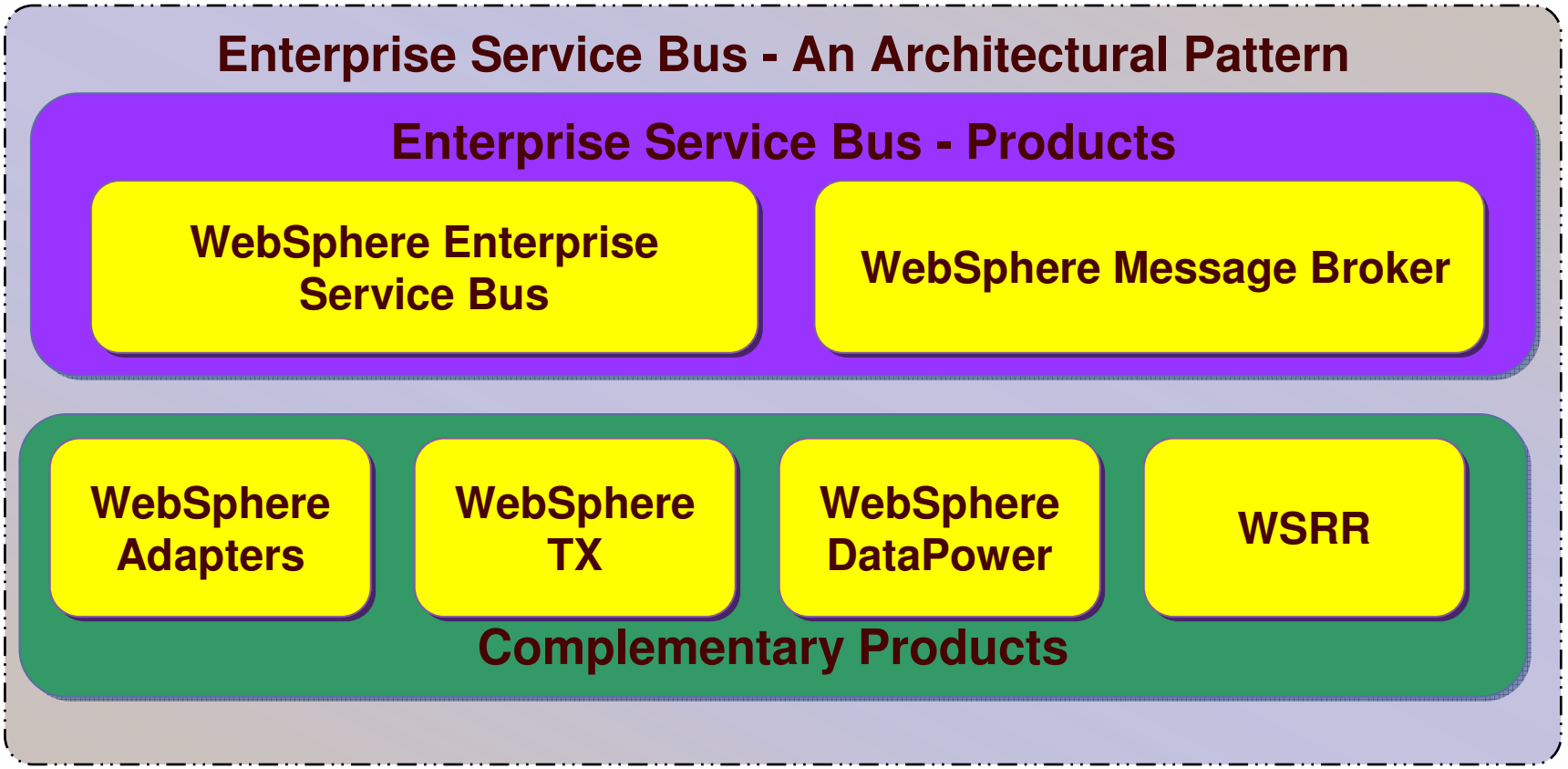
- Concept
  - Deploy and manage mediation modules
- Task
  - View deployed modules
  - Administratively modify module wirings for SCA bindings and Web Service bindings
  - Administratively modify certain properties of mediation primitives or of callouts
  - Perform via administrative console and CLI/scripting interfaces

The Development Cycle

- WID is used to develop the mediation module
- Unit test is done within WID using the WESB test environment or external servers
- Applications can then be exported and deployed on the target production servers using wsadmin or the administrative console



Solutions Supporting the ESB Pattern



WESB & WebSphere Message Broker Support

ESB:
WebSphere ESB

Advanced ESB:
WebSphere Message Broker

Web Services connectivity and data transformation

HTTP **JMS**
WebSphere MQ
Web Services **XML**
WebSphere Adapters

Universal connectivity and data transformation

HTTP **JMS** **WebSphere MQ**
Web Services **XML** **WebSphere Adapters**

Plus the following:

Weblogic JMS® Biztalk® TIBCO Rendezvous®
MQe Multicast Tuxedo® FTP TIBCO EMS JMS®
COBOL Copybook HIPAA EDI-FACT HL7 SonicMQ JMS®
ACORD Real-time IP AL3 Word/Excel/PDF
SWIFT FIX ebXML EDI-X.12 MQTT Custom Formats

providing high-speed data movement and universal mediation…
…enabling non-SOA applications to plug into the IBM SOA platform

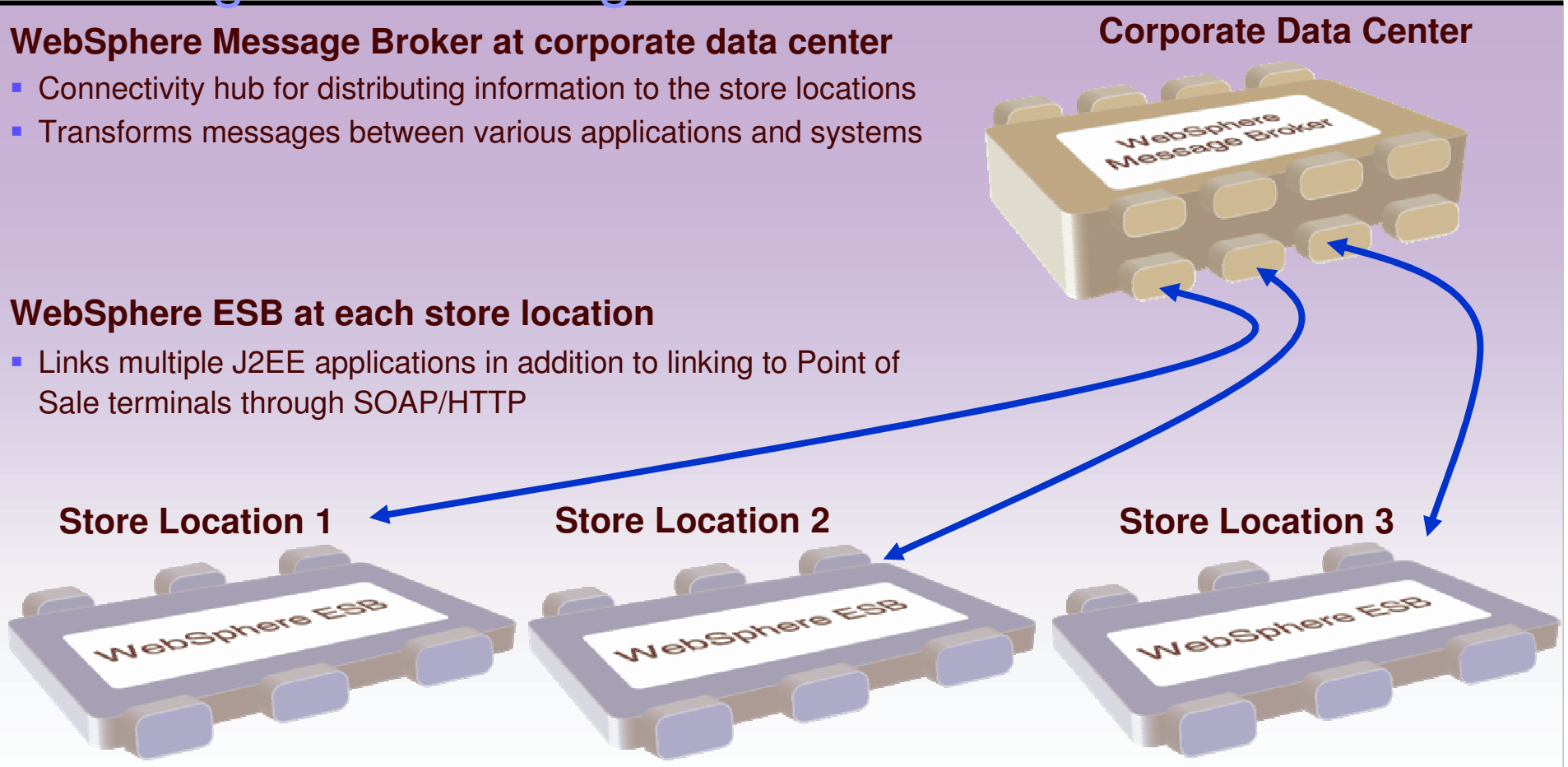
A Sample Integration Scenario - WESB & WebSphere Message Broker Integration

WebSphere Message Broker at corporate data center

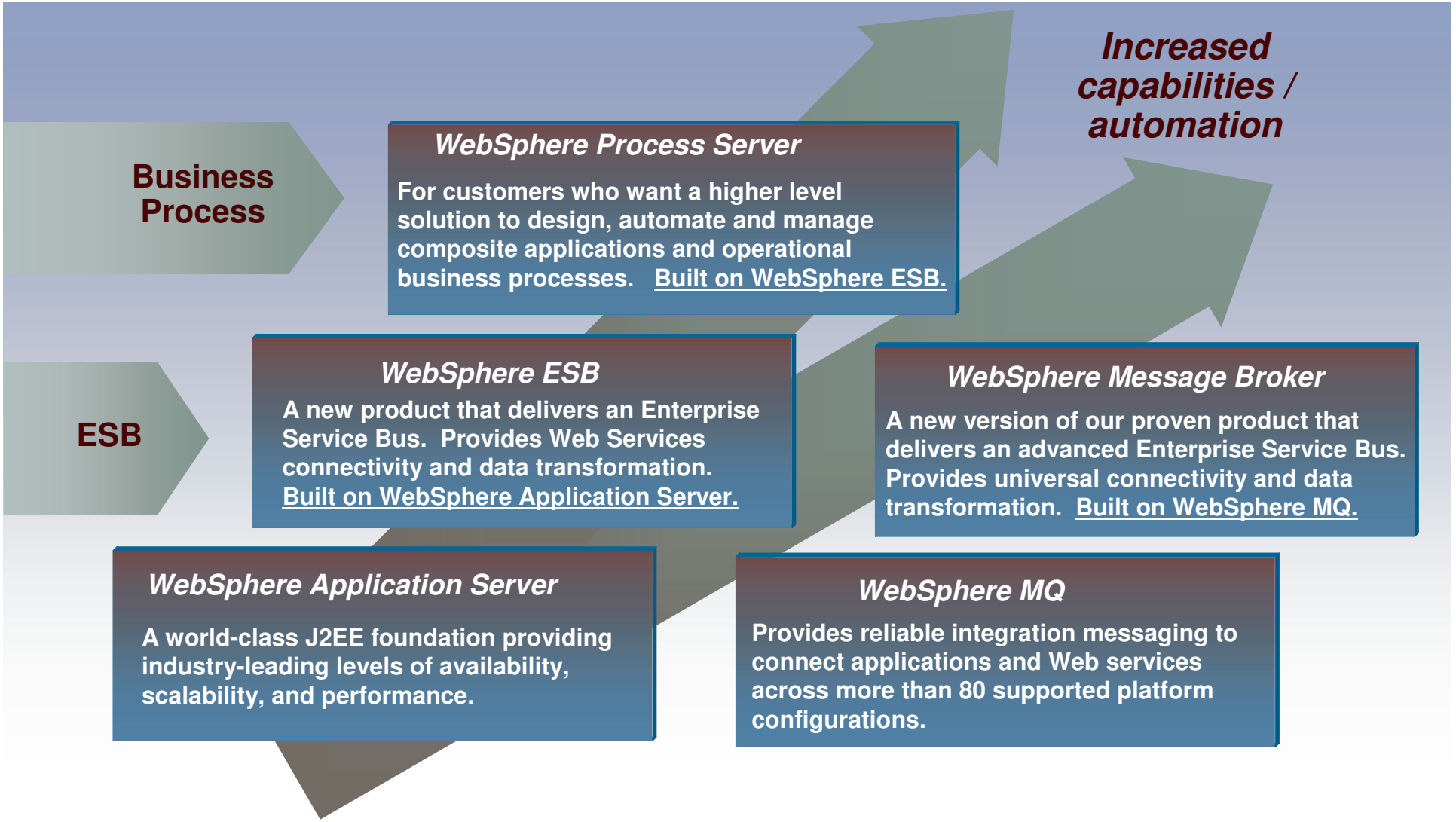
- Connectivity hub for distributing information to the store locations
- Transforms messages between various applications and systems

WebSphere ESB at each store location

- Links multiple J2EE applications in addition to linking to Point of Sale terminals through SOAP/HTTP



Summary


Increased capabilities / automation
Business Process
WebSphere Process Server
For customers who want a higher level solution to design, automate and manage composite applications and operational business processes. Built on WebSphere ESB.
ESB
WebSphere ESB
A new product that delivers an Enterprise Service Bus. Provides Web Services connectivity and data transformation. Built on WebSphere Application Server.
WebSphere Message Broker
A new version of our proven product that delivers an advanced Enterprise Service Bus. Provides universal connectivity and data transformation. Built on WebSphere MQ.
WebSphere Application Server
A world-class J2EE foundation providing industry-leading levels of availability, scalability, and performance.
WebSphere MQ
Provides reliable integration messaging to connect applications and Web services across more than 80 supported platform configurations.

Thank you!