

Basics of Message Flow

Objectives

After completing this lesson, you should be able to:

- Identify message exchange patterns
- Describe the message flow process
- Explain the functionality of Pipeline and Split-join
- Describe the usage patterns of different variables in a message flow
- Create a pipeline template and use it to create a pipeline
- Debug Service Bus pipelines



Agenda

- Message flow
- Context variables
- Pipeline template
- Debugging

Message Exchange Pattern

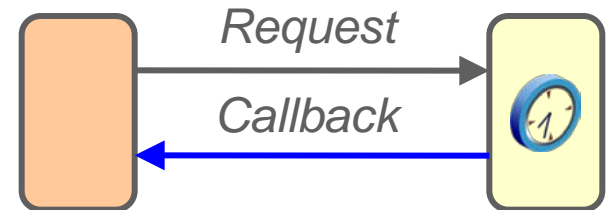
Synchronous request/response

- Real-time response or error feedback
- Client in waiting mode



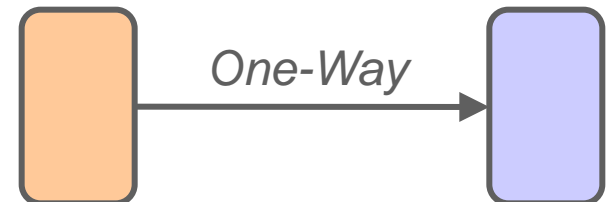
Asynchronous request/callback

- Client free after request submission
- Separate service invocation for response



Publish

- Also known as “fire and forget”
- Client free after request submission
- No response message (ACK only)

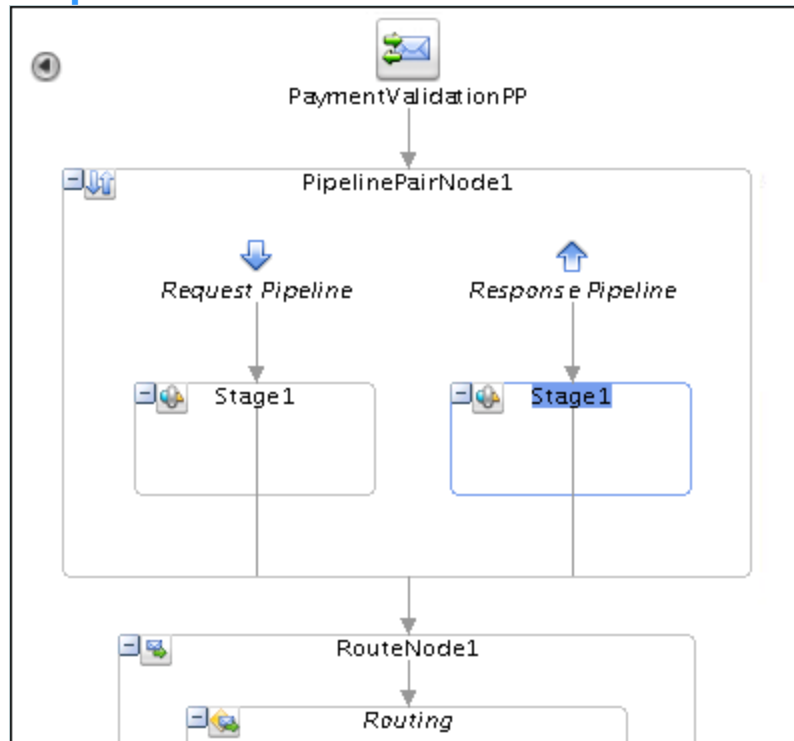


Message Flow

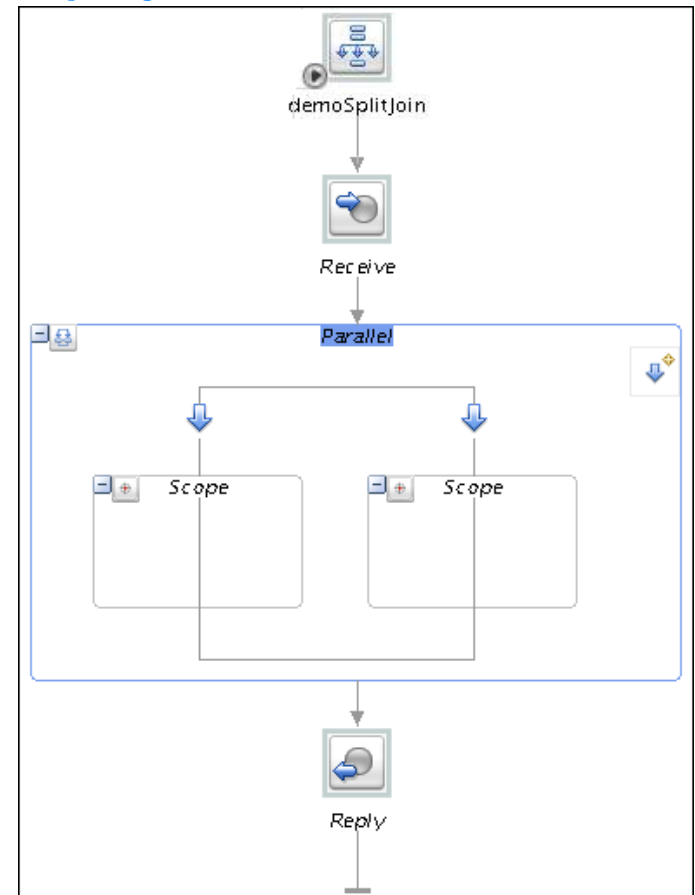
Message flow can be defined in a:

- Pipeline
- Split-join for parallel processing

Pipeline



Split-join



Pipeline

- Provides message processing capability
- Separated from Proxy Service, thereby allowing for better structured development
- A named sequence of stages containing *actions*
- Has its own unique message context and variables

Definitions:

- *Stages*: A group of actions
- *Actions*: Instructions, created graphically, with the occasional mix of XPath, XQuery, and/or XSLT










Pipeline Components

Primary elements for pipeline are:










- **Start node:** Mark the entry point of the message flow
- **Pipeline pair node:** Node for request and response pair processing
- **Stage node:** Container of actions to manipulate messages passing through the pipeline
- **Branch node:** Route the message based on condition either on type of operation or values
- **Route node:** Set the destination of message otherwise by default echo the request
- **Error Handler:** Handle errors either on stage or node

Note: Minimum start node and route node are required for a pipeline.










Message Processing Actions

	Assign	Assign the result of an XQuery expression to a context variable
	Delete	Delete a context variable or set of nodes specified by XPath expression
	Insert	Insert the result of an XQuery expression at an identified place relative to nodes selected by an XPath expression
	Rename	Rename an element selected by an XPath expression without modifying the contents of the element
	Replace	Replace a node or a node's contents specified by an XPath expression
	Validate	Validate elements selected by an XPath expression against an XML schema element or a WSDL resource
	Java Callout	Invoke a Java method from within the pipeline
	MFL Transform	Convert non-XML to XML or XML to non-XML within the pipeline
	nXSD Translate	Convert message content from XML to native format data, or vice versa

Communication Actions

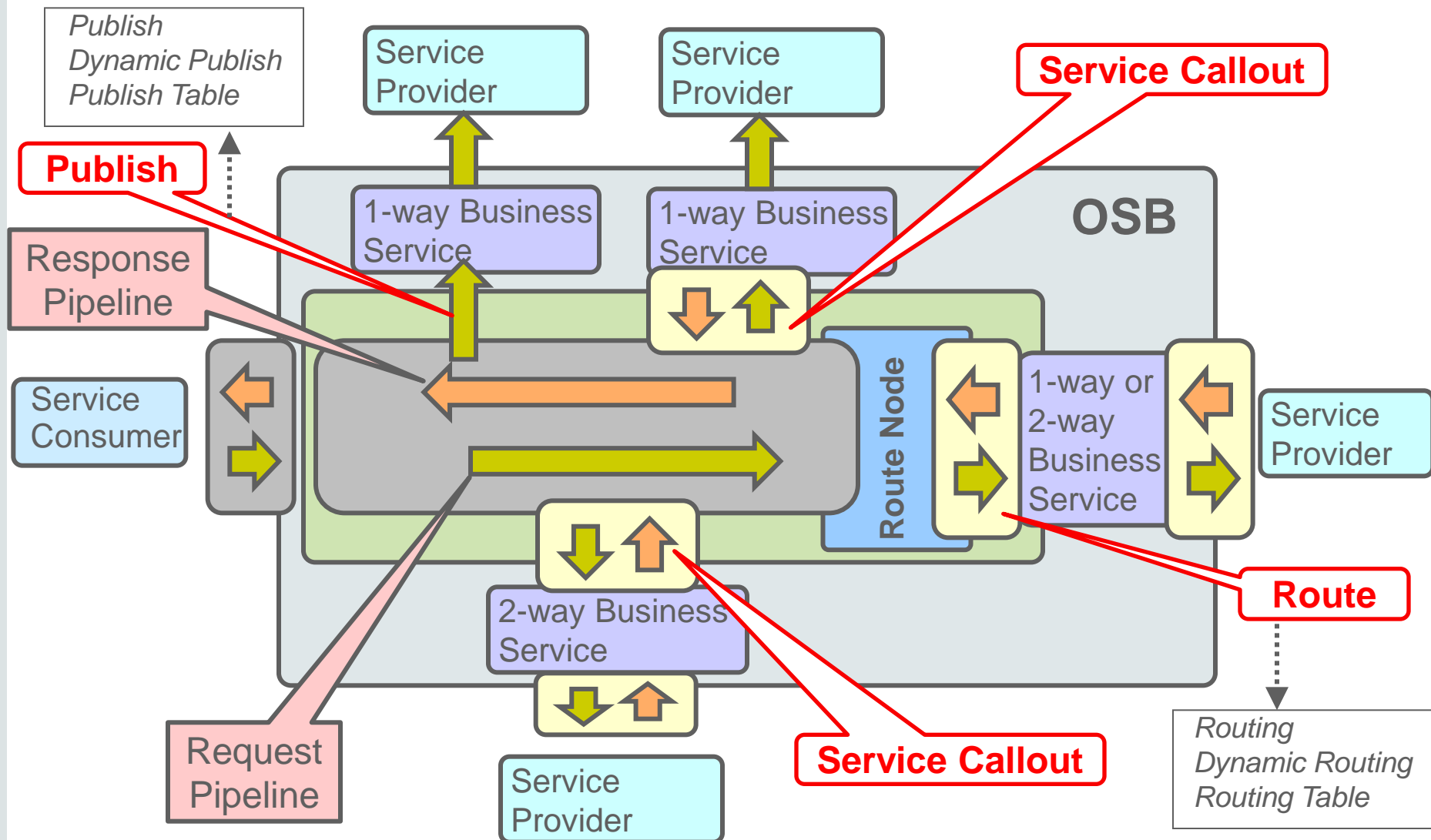
	Routing	Route a message to a statically specified service
	Dynamic Routing	Route to a service based on information in an XQuery resource
	Routing Table	Route to one of a set of services wrapped in a switch-style condition
	Publish	Publish a message to a statically specified service
	Dynamic Publish	Publish to a service based on information in an XQuery resource
	Publish Table	Publish to one of a set of services wrapped in a switch-style condition
	Service Callout	Configure a synchronous (blocking) callout to proxy or business service
	Routing Options	Override outbound request properties (such as URI or Retry Count)
	Transport Headers	Set the transport header values for outbound messages

Flow Control and Report

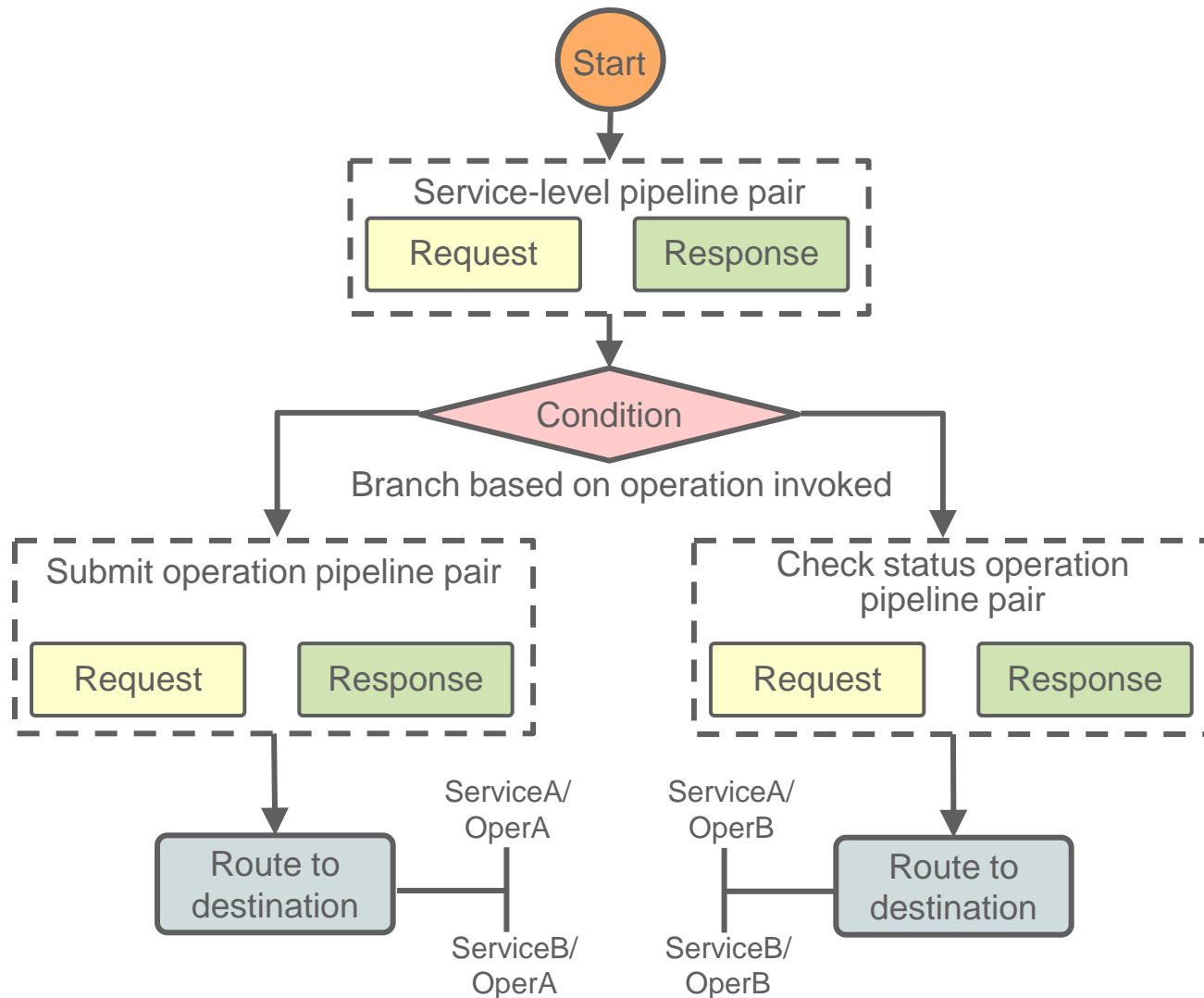
Flow Control Actions		
	For...Each	Iterate over a sequence of values and execute a block of actions
	If...Then	Perform an action or set of actions conditionally, based on the Boolean result of an XQuery expression
	Skip	Specify that at runtime, the execution of the current stage is skipped and the processing proceeds to the next stage in the message flow
	Reply	Force an immediate reply to be sent to the invoker of this proxy service
	Raise Error	Raise an exception with a specified error code (a string) and description
	Resume	Resume message flow after an error is handled by an error handler
Report Actions		
	Alert	Send an alert notification based on pipeline message context
	Log	Construct a message to be logged to the WebLogic server log file
	Report	Define data with an index and key value to be sent to the OSB Database

Only used in Error Handler stage

Message Flow Communication Actions



Pipeline: Example



Quiz



You can use more than one route node in a message flow.

- a. True
- b. False

Agenda

- Message flow
- Context variables
- Pipeline template
- Debugging

Message Context

A set of properties, known as *context variables*:

- Hold the message content as well as information about messages
- Shared across the request flow and response flow
 - Within one pipeline
 - Across multiple pipelines
- Used to:
 - Reassemble
 - Modify
 - Reroute
 - Make runtime decisions about message processing

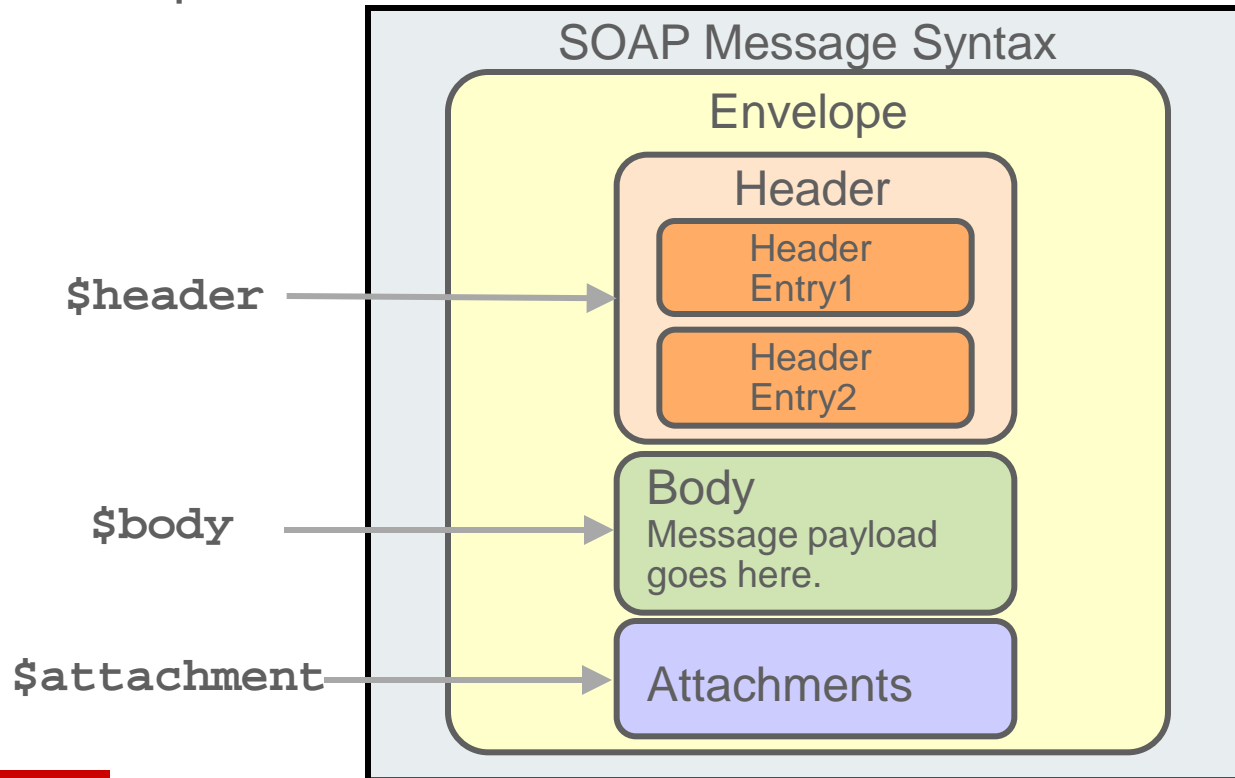
Predefined Context Variables

The predefined context variables can be grouped into the following types:

- Message-related variables: `$header`, `$body`, `$attachment`
- Inbound and outbound variables: `$inbound`, `$outbound`
- Operation variable: `$operation`
- Fault variable: `$fault`

Message-related Variables

- A message payload is contained in the `body` variable.
- The decision about which variable's content to include in an outgoing message is made at the point at which a message is dispatched, published, or routed.



`$inbound` and `$outbound` Variables

- `$inbound` variable contains:
 - Information about the proxy service that received a message
 - Inbound transport headers
- `$outbound` variable contains:
 - Information about the target service to which a message is sent
 - Outbound transport headers

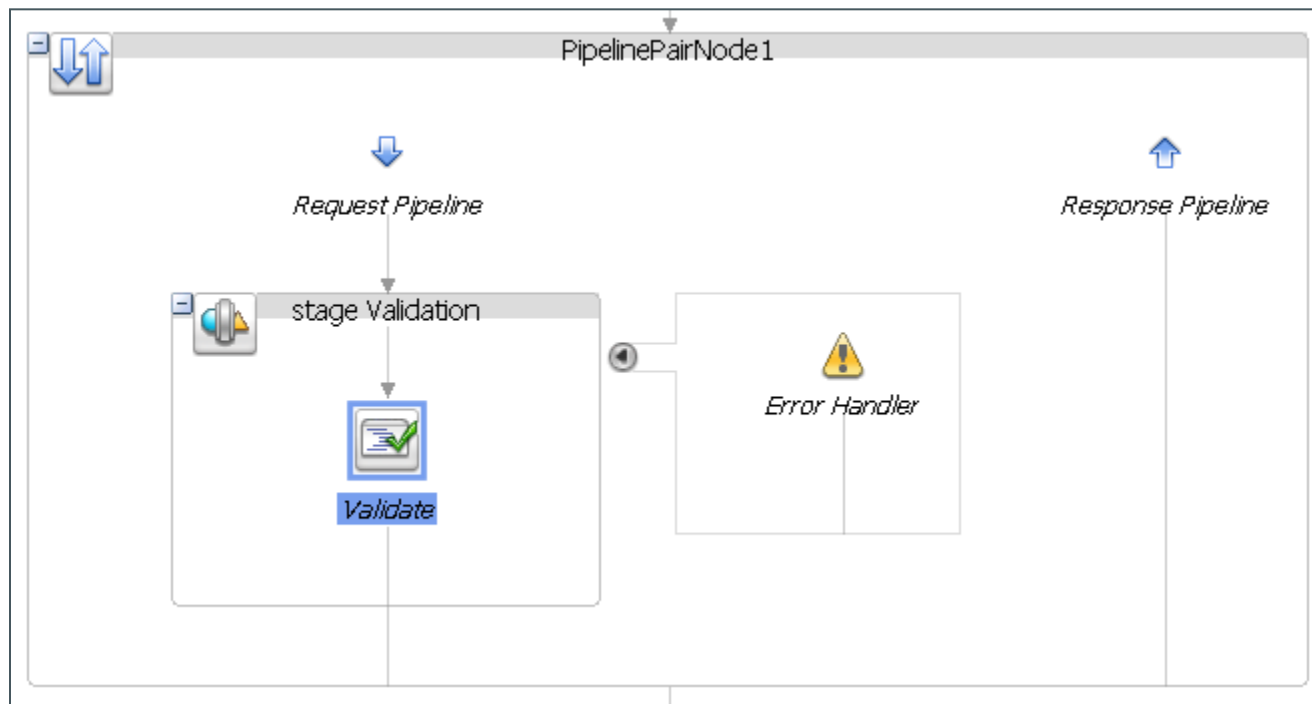
\$operation Variable

- The `operation` variable is a read-only variable.
- It contains a string that identifies the operation to be invoked on a proxy service.

\$fault Variable

\$fault variable is:

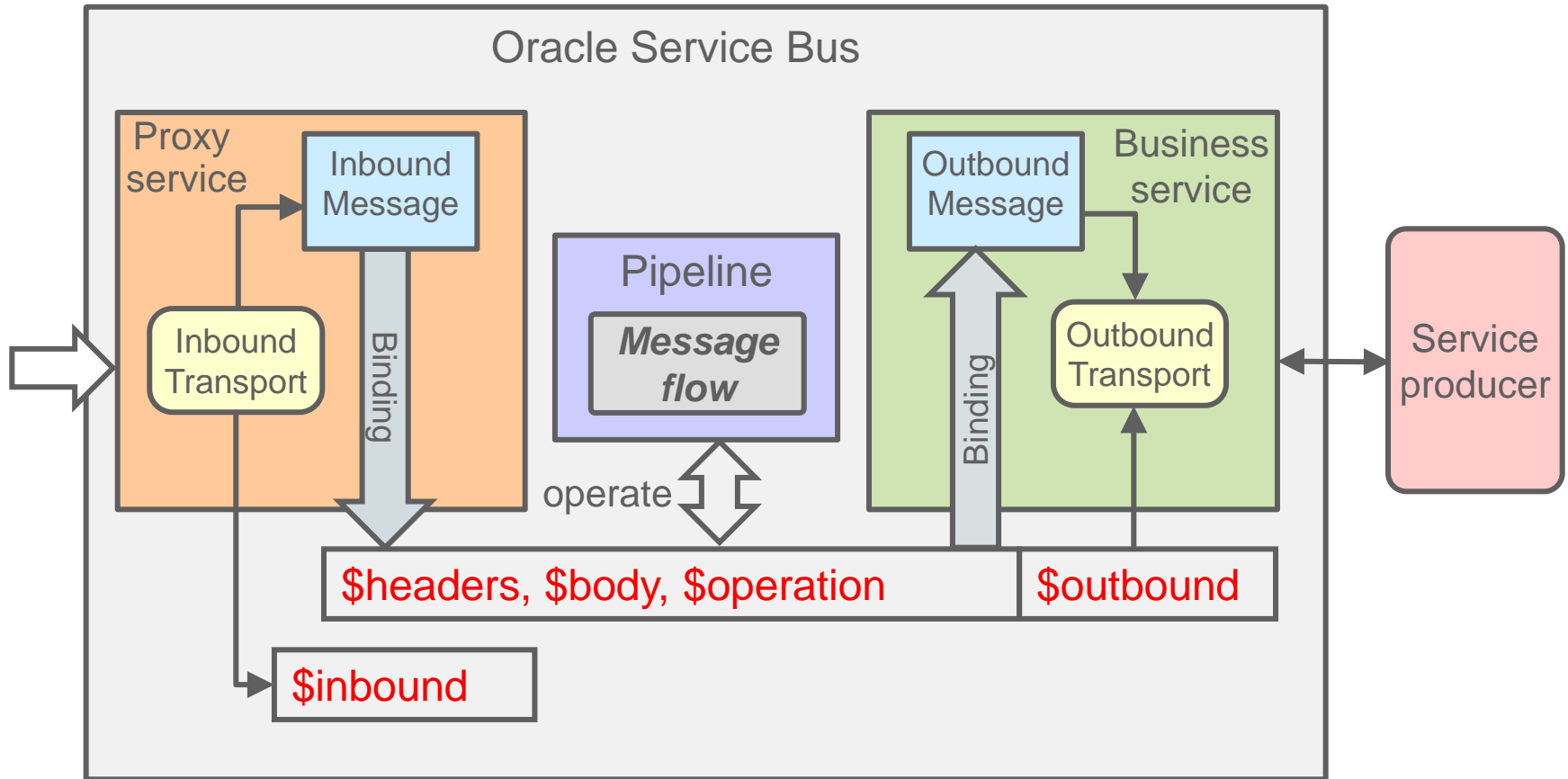
- Populated with error information before the appropriate error handler is invoked
- Defined only in error handler pipelines



Variables and Types

- At design time:
 - All variables (user and system defined) are scoped for the entire service
 - *\$outbound*, *\$inbound*, *\$fault*, and *\$operation* are typed. Others are not because their contents vary.
- At runtime:
 - Variables are automatically created when first assigned a value during execution
 - Variables are deleted when the service execution terminates

Context Variables in Message Processing



Pass Content by Reference

- Binary content (non-XML, non-text) (*always*)
 - Stored in an in-memory hash table
 - Referenced in the `$body` and `$attachments` variables:
`<binary-content ref=...>`
- File, email, and FTP (*optional*)
 - Passed as a reference in the message headers

EMAIL Transport Configuration
Use this page to configure the transport information for this service.

Transport Details

SSL Required ☐

Service Account* <Not Selected> 🔍

Managed Server

Polling Interval*

Email Protocol* pop3 ▼

Read Limit*

Pass By Reference ☒

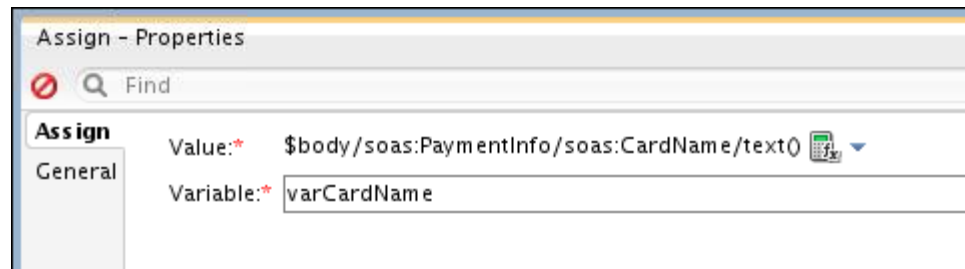
Pass Attachments By Reference ☒

Post Read Action* delete ▼

Attachments* ignore ▼

User-Defined Variables

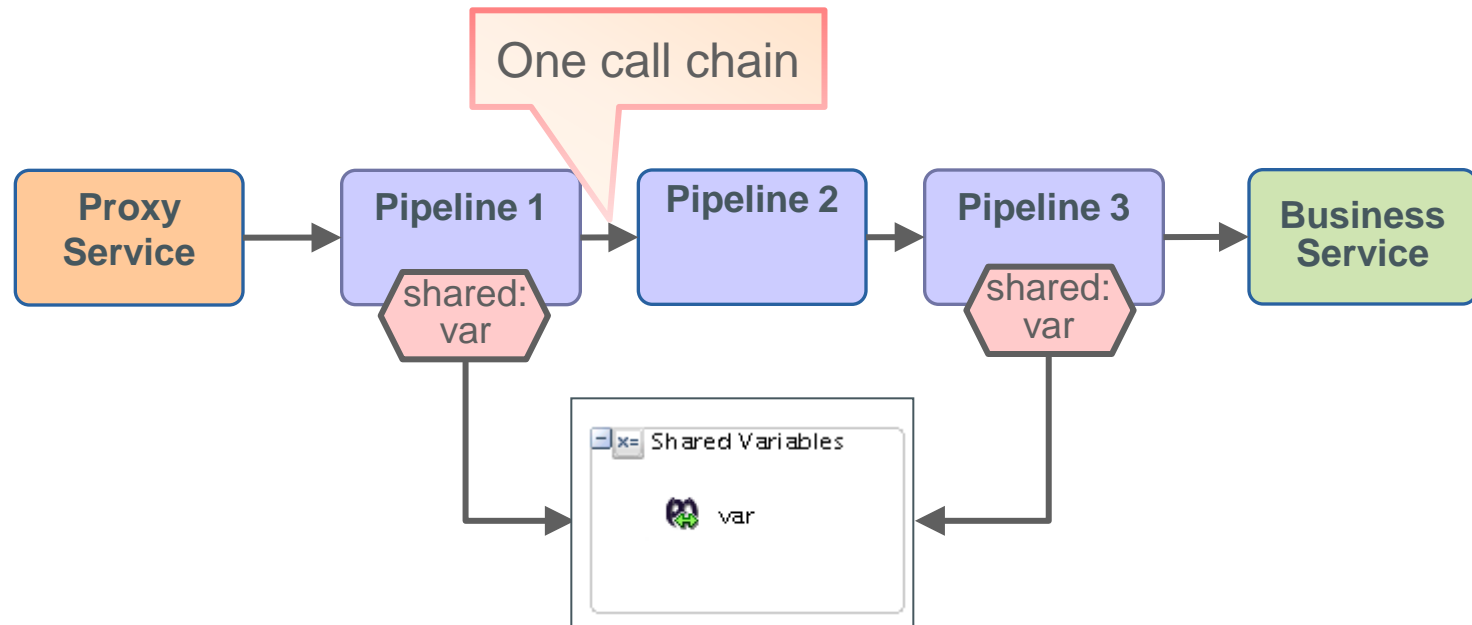
- Using the Assign action is one way of defining a variable in a pipeline.
- The scope of the variable is for the request pipeline, response pipeline, route node, and fault pipeline.



Note: Service Bus is stateless and therefore variables do not persist.

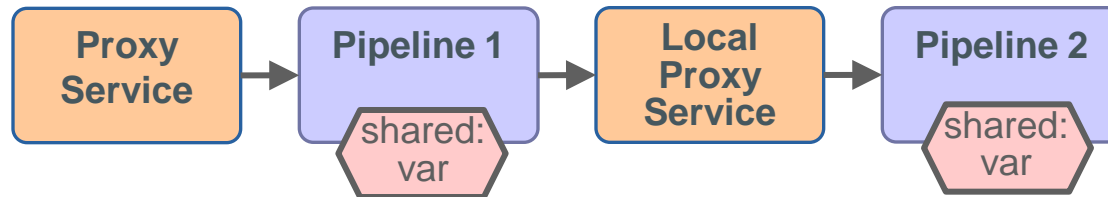
Shared Variables

When two pipelines in the same call chain declare a variable with the same name as shared, they will be reading and modifying the same variable.

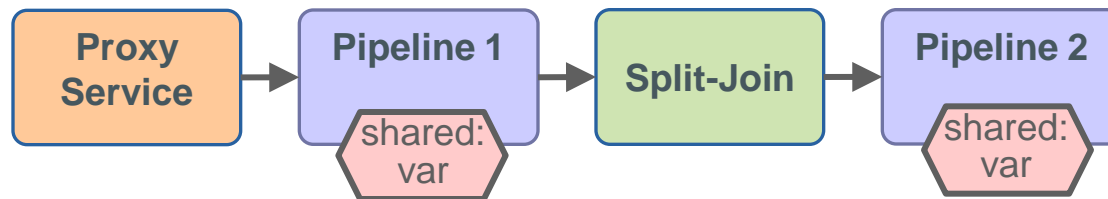


Shared Variable Rules

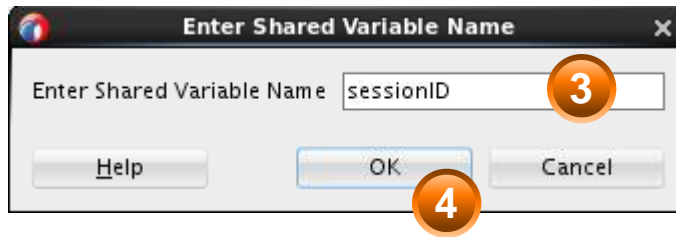
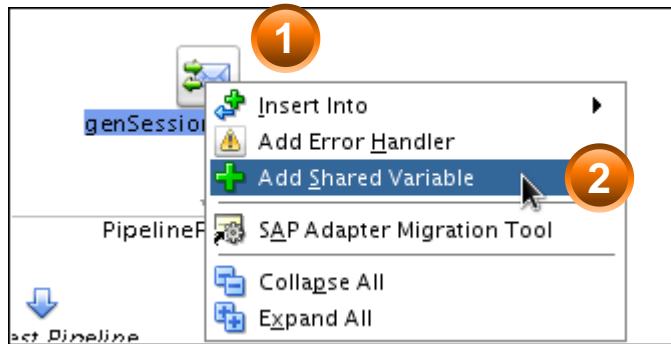
- The scope of sharing is confined to a single message being processed by pipelines within one call chain.
- Work across:
 - Local proxy invocations



- Split-join component invocations



Creating a Shared Variable



Quiz



Which one of the following is not a predefined context variable?

- a. Header
- b. body
- c. outbound
- d. fault
- e. transport

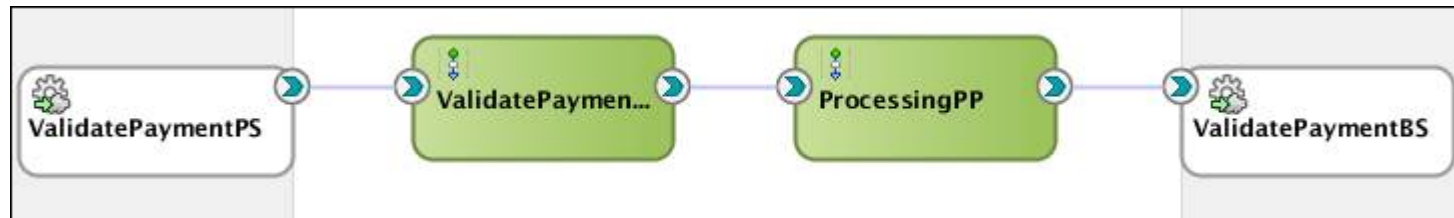
Agenda

- Message flow
- Context variables
- Pipeline template
- Debugging

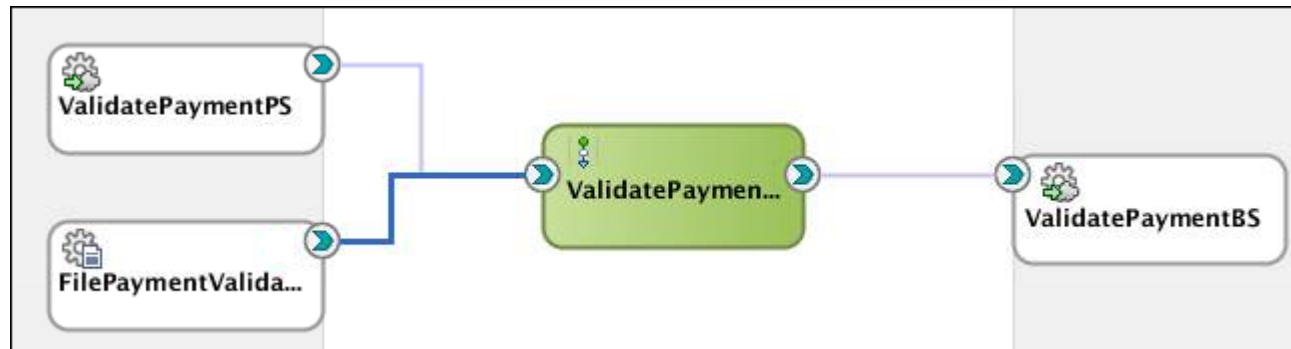
Pipeline Construction

Pipelines can be:

- Chained

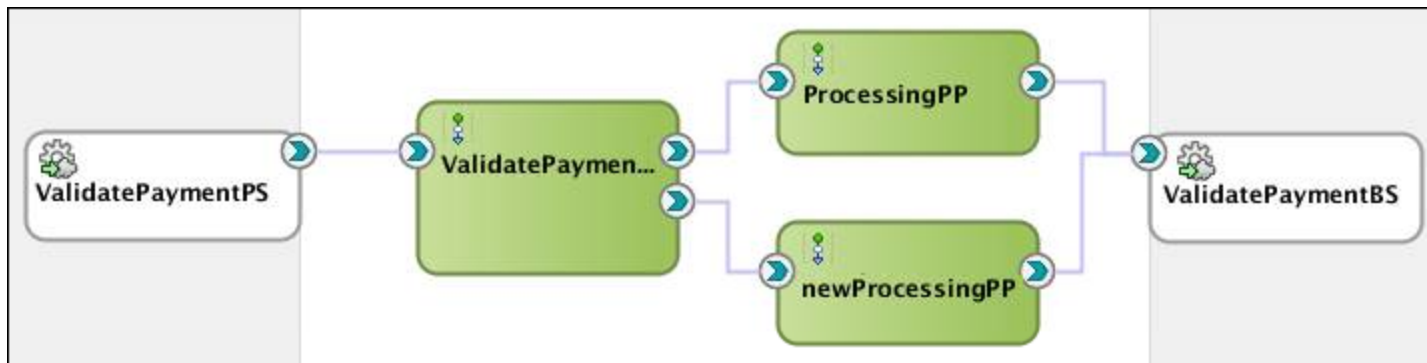


- Reused



Pipeline Construction

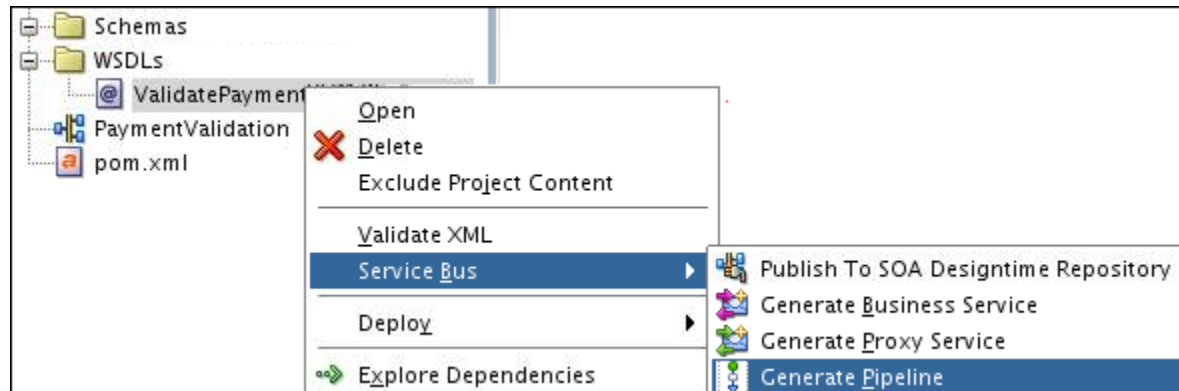
- One pipeline can invoke another pipeline by using a callout or publish action.



Creating a Pipeline

A Pipeline can be created by using:

- WSDL service definition document



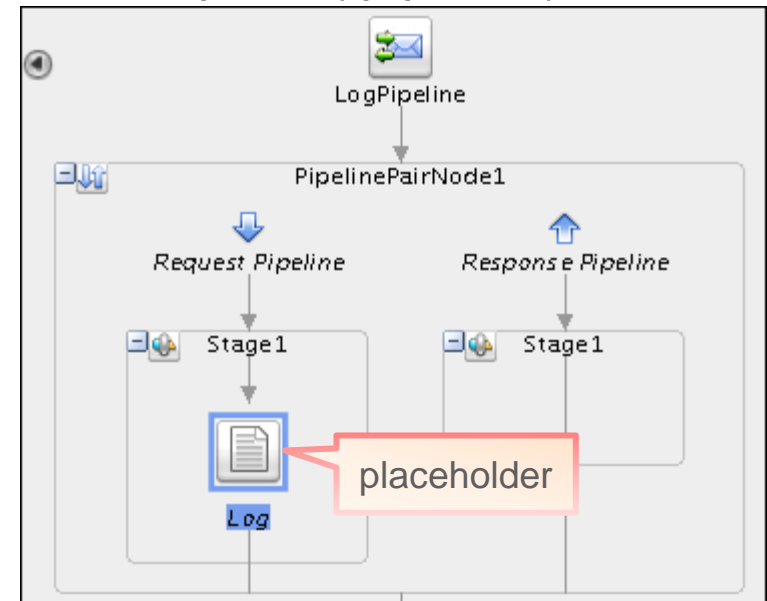
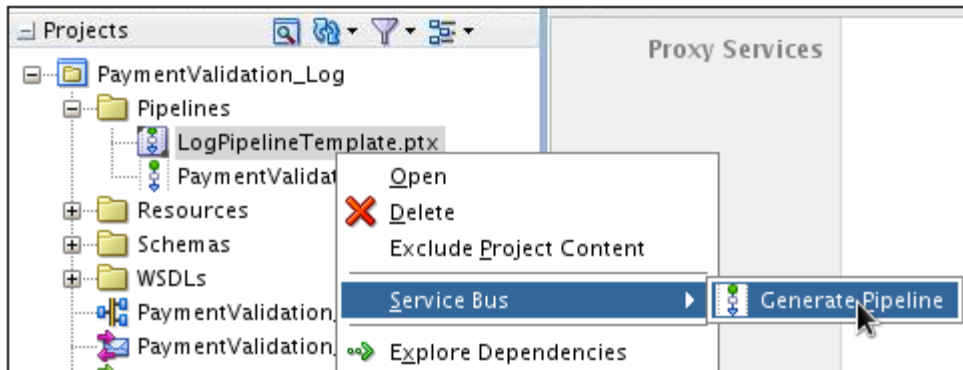
- Pipeline template
- Dragging the Pipeline component from the component palette
- Generating from the proxy service

Pipeline Templates

- Pipeline templates can be used to design prototype flows.
- Typical components included in a template are:
 - Validation
 - Error Handler
 - Logging
 - Alerts
 - Reports

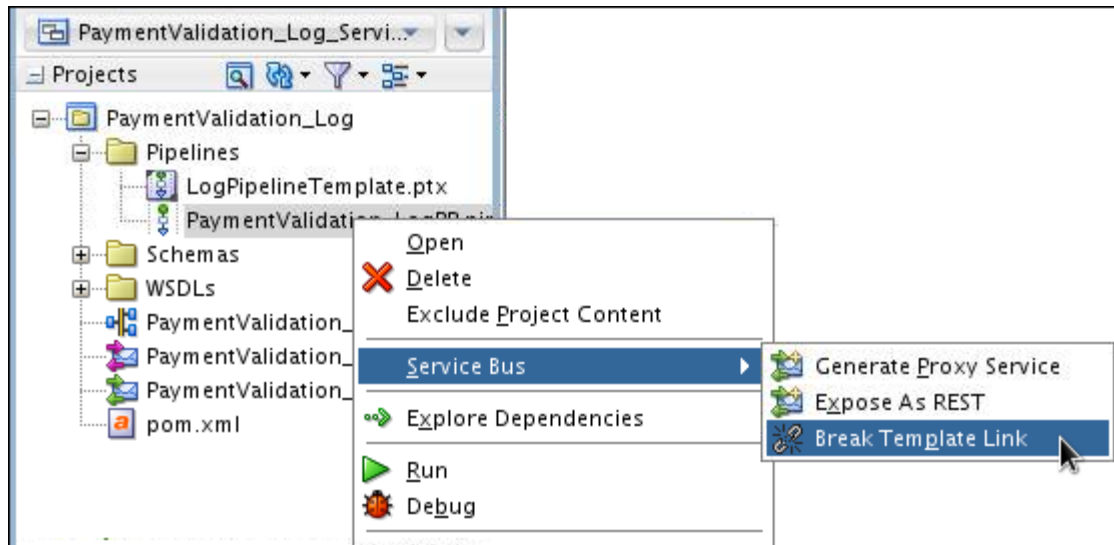
Creating and Using Pipeline Template

- Create template:
 - Create a pipeline template (.ptx).
 - Export the template file as a resource (.jar).
- Use template:
 - Import the template file.
 - Create a concrete pipeline using the template (.pipeline).
 - Fill the placeholders.



Breaking a Template Link

- A concrete pipeline is automatically associated with the pipeline template.
- To break the link:
 - Use the Configuration tab of the Pipeline Editor.
 - Right-click the pipeline in Application Navigator, select Service Bus > Break Template Link.



Agenda

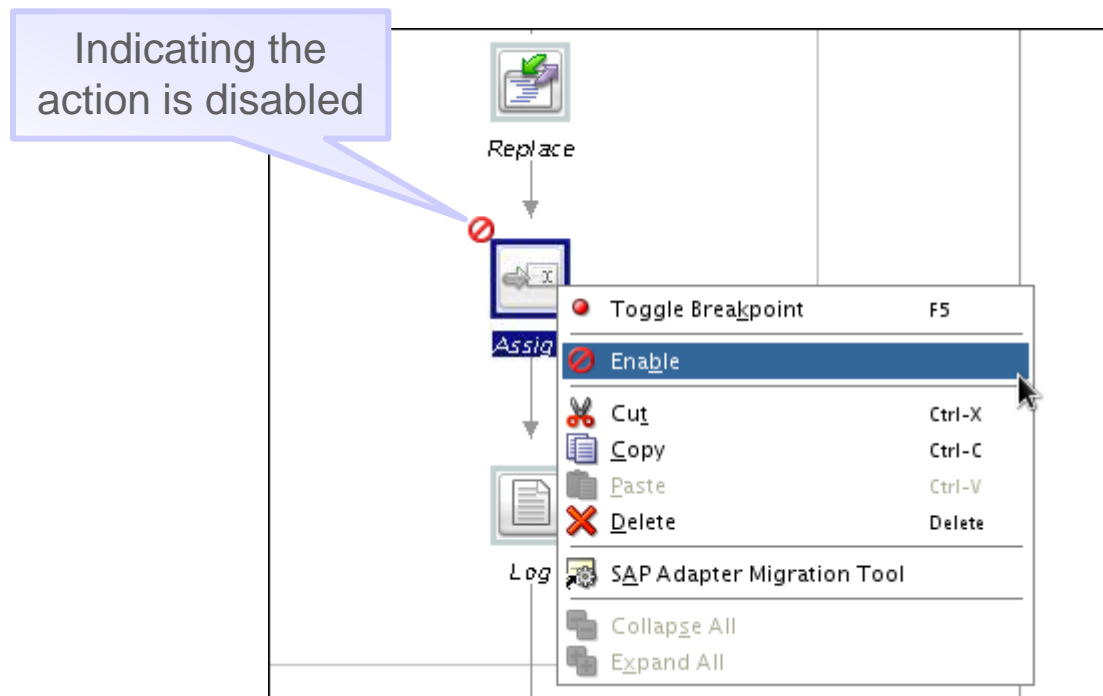
- Message flow
- Context variables
- Pipeline template
- Debugging

Debugging Service Bus Applications

- Enable/disable stages and actions (JDeveloper)
- Flow tracing (EM)
- Debugger (JDeveloper)

Enable/Disable Stages and Actions

- Allow the user to “skip” a single activity or a stage at runtime without having to delete the action/stage from the flow.
- An action or stage that was previously disabled may be re-enabled.



Enable Flow Tracing in Pipeline

EM Console

PaymentValidation_ValidateMsgPP (Pipeline) Test

Dashboard Properties

Monitoring

Monitoring ☐ Enabled

Monitoring Level Service level or above

Aggregation Interval 10 Mins

Tracing

Execution Tracing ☒ Enabled

Logging

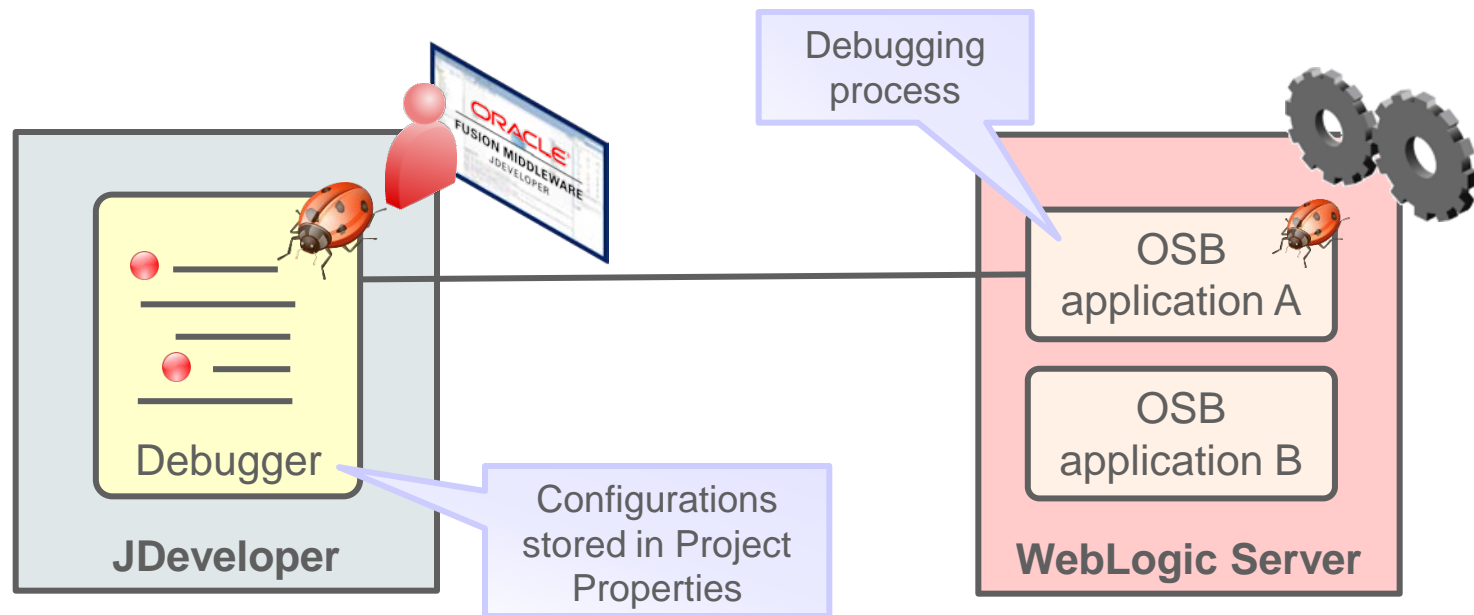
Test Console

Invocation Trace

- ⊕ (receiving request)
- ↕ Message Flow
 - ⊕ Data Validation
 - Message Context Changes**
 - ⚠ ⊕ changed \$body
 - ⚠ ⊕ changed \$inbound
 - ⊕ RoutetoBusinessService
 - Routed Service**
 - ⚠ ⊕ Route to: "PaymentValidation_ValidateMsgBS"
 - Message Context Changes**
 - ⊕ added \$outbound
 - ⚠ ⊕ changed \$body
 - ⚠ ⊕ changed \$attachments
 - ⚠ ⊕ changed \$inbound
 - ⚠ ⊕ changed \$header
 - ↕ Message Flow
 - ⊕ Auditing
 - Message Context Changes**
 - ⚠ ⊕ changed \$body
 - ⚠ ⊕ changed \$outbound
 - ⚠ ⊕ changed \$inbound

Debugging Service Bus Applications with Debugger

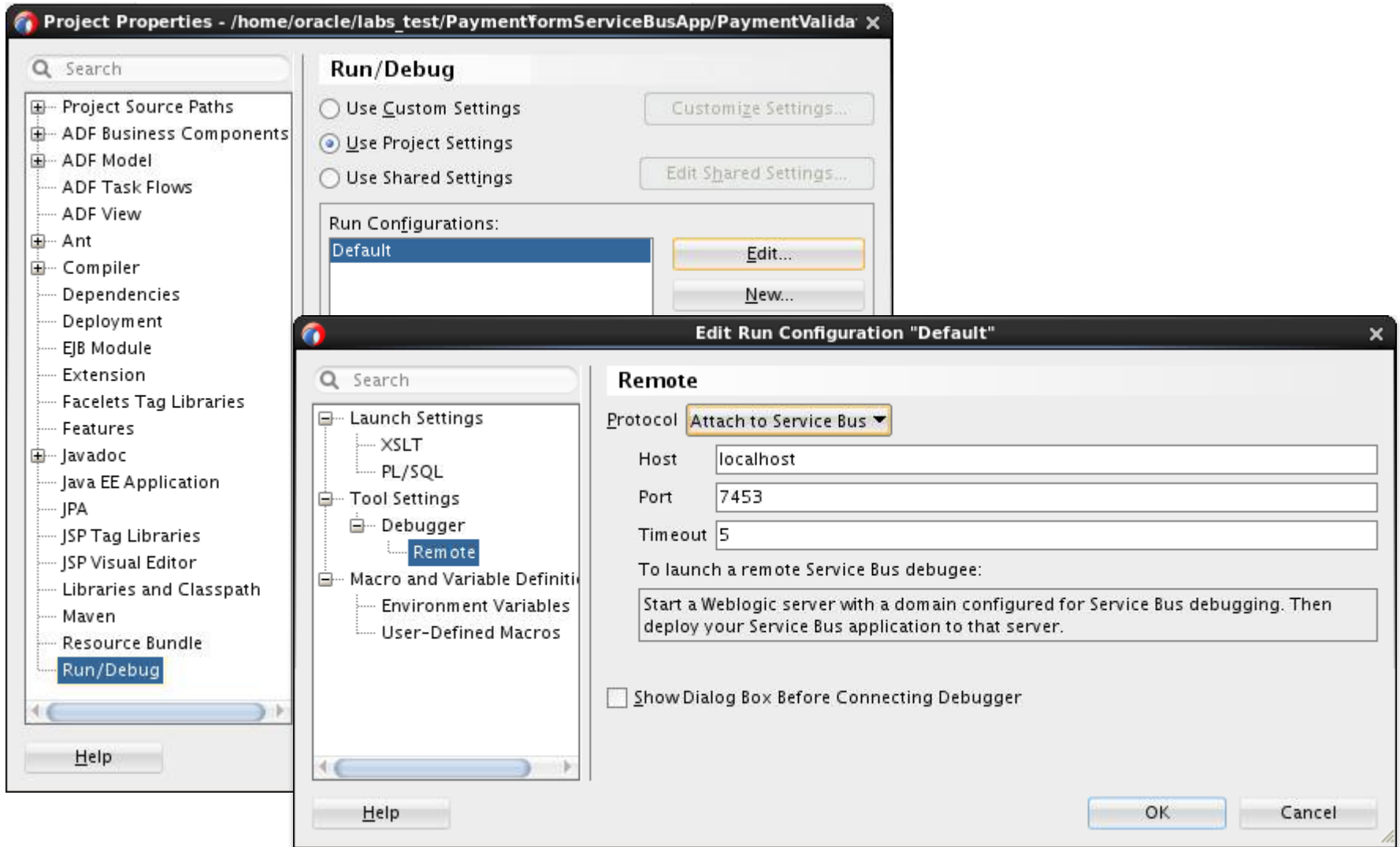
Service Bus debugger provides a troubleshooting environment within JDeveloper.



Features and Limitations

- Supports:
 - Multi-threaded debugging on split-joins
 - Debugging on:
 - Either integrated or stand-alone WebLogic server
 - Either a local or remote WebLogic server
- Limitations
 - Limited to design view in JDeveloper
 - No debugging cross-language features, such as a Java callout action, XSLT and XQuery transformations
 - Only one client at a time can connect to the debugger.

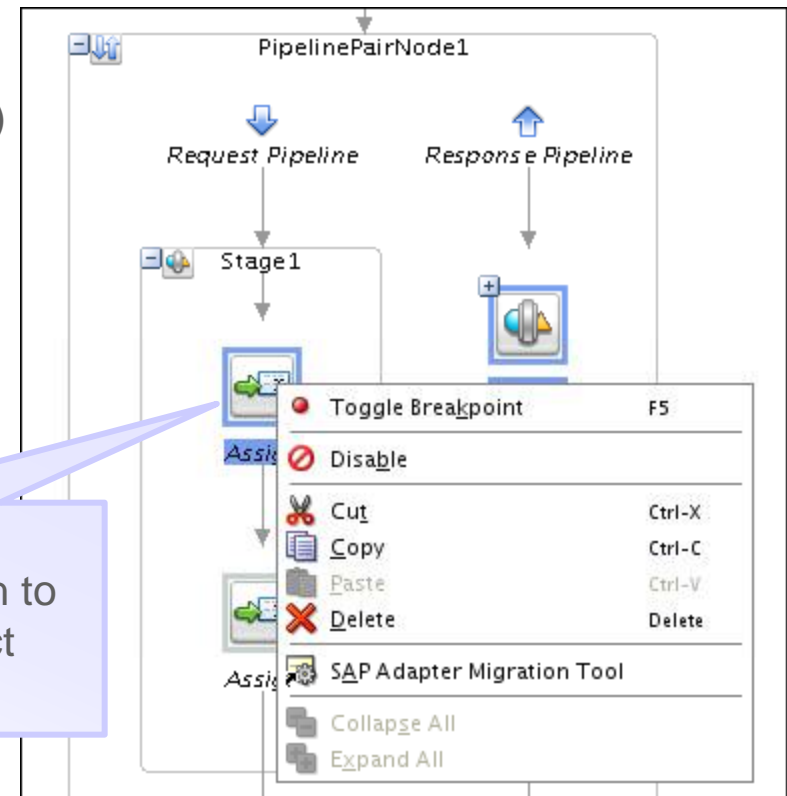
Configuring Debugger



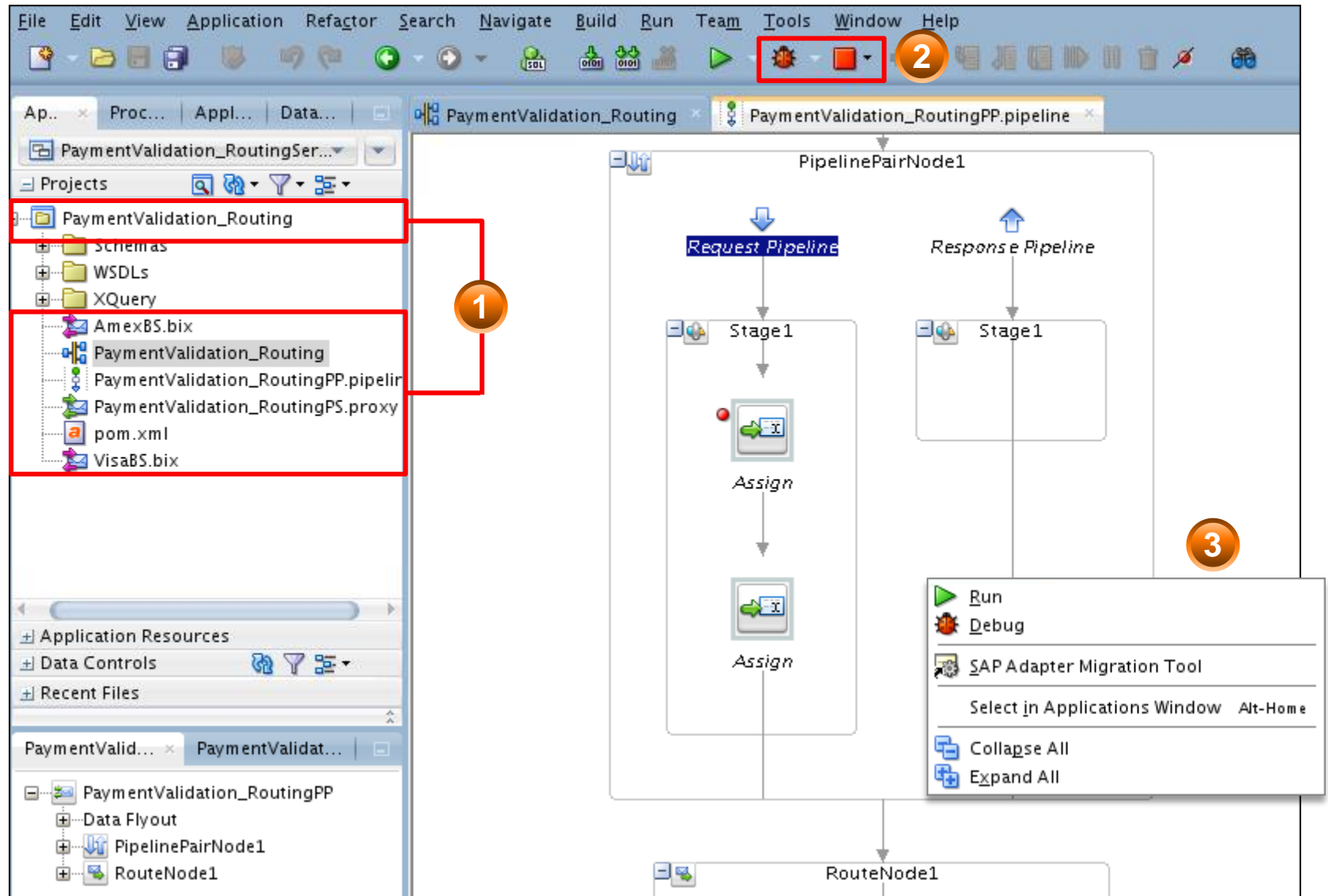
Setting Breakpoints

- Breakpoints can be added to the following nodes on a pipeline:
 - Route node
 - Route action
 - Branch node
 - Pipeline pair path (request or response)
 - Stage (pipeline pairs and error handlers)
 - Stage action
- Breakpoints can be added to all split-join actions.

Right-click the action/node/stage on which to set a breakpoint, and select Toggle Breakpoint.

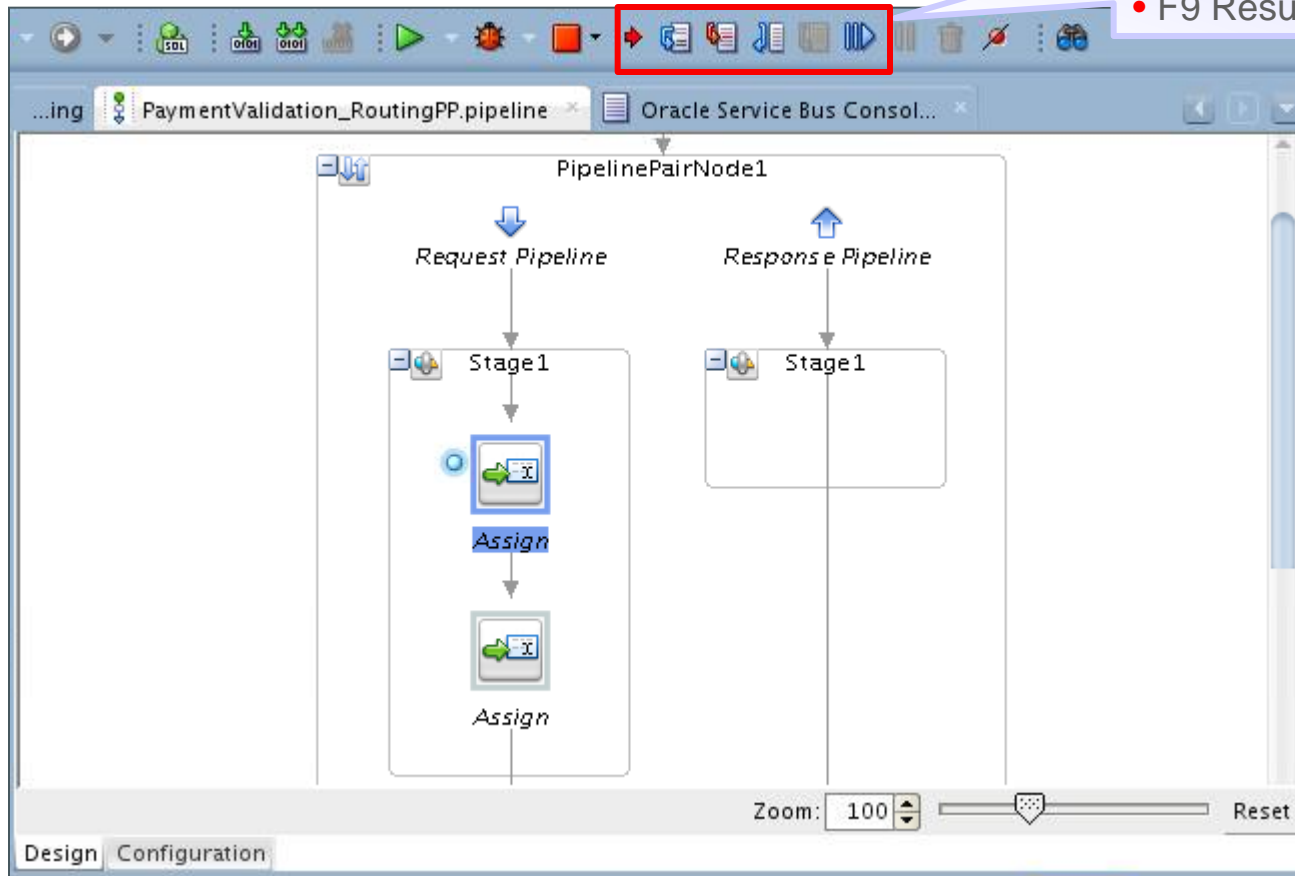


Starting the Debugger



Stepping Through a Debugging Session

- F7 Step Into
- Shift-F7 Step Out Of
- Shift-F8 Step Over
- F9 Resume Step



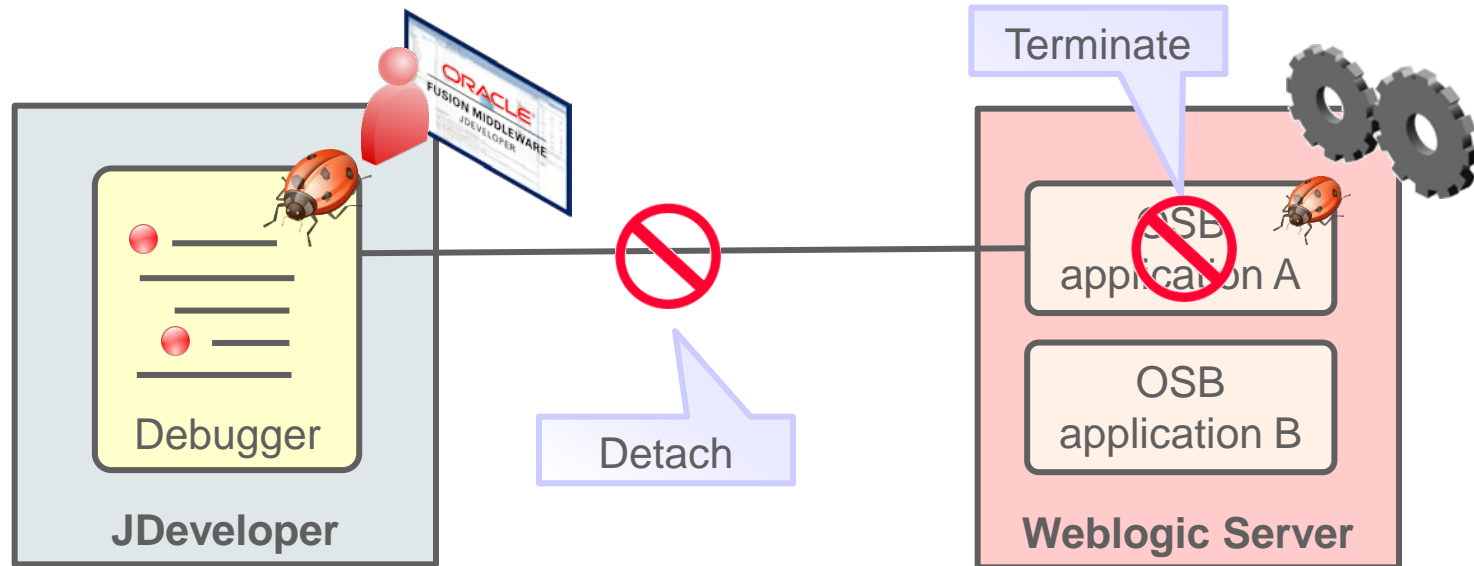
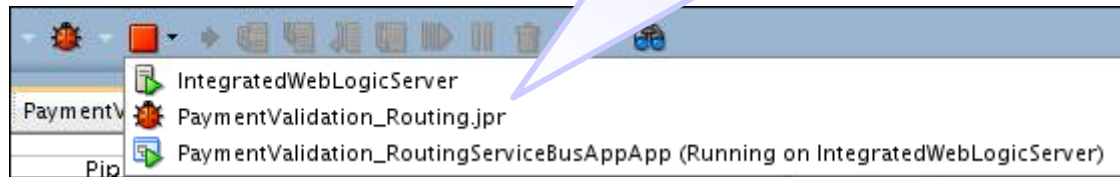
Viewing and Modifying Variable Values

The screenshot shows the Oracle ADF IDE interface. On the left, the 'Window' menu is open, displaying a list of tool windows. The 'Debugger' window is selected, and its sub-menu is also open, showing 'Data' as the active view. A blue callout box highlights the 'Data' window, which is shown in a larger, detailed view on the right. The 'Data' window displays a table of variables and their values.

Name	Value	Declared Type	Hex Value	Address
\$attachments	<attachm...			
\$body	<Body>...			
\$header	<Header/>			
\$inbound	<endpoin...			
\$messageID	a962529.321...			
\$operation	validate			
\$outbound	null			
\$varCardName	AMEX			
\$varConfig	<Configu...			
Java Repository				
Source Reposit...				

Ending a Debugging Session

As long as the debugger is running, the project cannot be edited.



Summary

In this lesson, you should have learned how to:

- Identify message exchange patterns
- Describe the message flow process
- Explain the functionality of Pipeline and Split-join
- Describe the usage patterns of different variables in a message flow
- Create a pipeline template and use it to create a pipeline
- Debug Service Bus pipelines



Practice 4: Overview

- 4-1: Creating a pipeline template for logging
- 4-2: Creating a new pipeline using a pipeline template
- 4-3: Debugging a Service Bus Application in JDeveloper