# Oracle® Application Integration Architecture Foundation 11*g* Release 1 (11.1.1.2.0): Infrastructure Components and Utilities Guide

Release 1 (11.1.1.2.0)

**Part No. E17366-01**

April 2010

ORACLE

Oracle Application Integration Architecture Foundation Pack 11*g* Release 1 (11.1.1.2.0): Infrastructure Components and Utilities Guide

Part No. E17366-01

# Contents

# Preface

Welcome to the *Oracle Application Integration Architecture Foundation Pack 11g Release 1 (11.1.1.2.0): Infrastructure Components and Utilities Guide.*

Oracle Application Integration Architecture (AIA) provides the following guides and resources for this release:

## Oracle AIA Guides

- *Oracle Application Integration Architecture Foundation Pack: Installation Guide*
- *Oracle Application Integration Architecture Foundation Pack: Getting Started with the Oracle AIA Foundation Pack and Demo*
- *Oracle Application Integration Architecture Foundation Pack: Concepts and Technologies Guide*
- *Oracle Application Integration Architecture Foundation Pack: Development Guide*
- *Oracle Application Integration Architecture Foundation Pack: Infrastructure Components and Utilities Guide*
- *Oracle Application Integration Architecture Foundation Pack: Reference Process Model Guide*
- *Oracle Application Integration Architecture Foundation Pack: Migration Guide for Foundation Pack 2.X to Foundation Pack 11gR1 (11.1.1.2.0)*

## Additional Resources

The following resources are also available:

| Resource | Location |
|---|---|
| Oracle Application Integration Architecture Foundation Pack: Product-to-Guide Index | My Oracle Support: https://support.oracle.com/ |
| Known Issues and Workarounds | My Oracle Support: https://support.oracle.com/ |
| Release Notes | Oracle Technology Network: http://www.oracle.com/technology/ |
| Documentation updates | My Oracle Support: https://support.oracle.com/ |

# Part: Working with the CAVS

# 1. Introduction to the CAVS

This chapter discusses the following topics:

- Describing the Purpose of the CAVS
- Describing Key Components of the CAVS Framework
- Describing the CAVS Design Assumptions and Knowledge Prerequisites

## 1.1. Describing the Purpose of the CAVS

The Composite Application Validation System (CAVS) is a framework that provides a structured approach to test integration of Oracle Application Integration Architecture (AIA) services. The CAVS includes test initiators that simulate web service invocations and simulators that simulate service endpoints.

In the context of AIA, where there is a sequence of service invocations; spanning Application Business Connector Services (ABCSs), Enterprise Business Services (EBSs), Enterprise Business Flows (EBFs), and participating applications; the CAVS test initiators and simulators enable a layered testing approach. Each component in an integration can be thoroughly tested without having to account for dependencies by using test initiators and simulators on either end.

Consequently, when you build an integration, you have the ability to add new components to an already tested subset, allowing any errors to be constrained to the new component or to the interface between the new component and the existing component. This ability to isolate and test individual web services within an integration provides the benefit of narrowing the test scope, thereby distancing the service test from possible faults in other components.

Test initiators and simulators can be used independent of each other, thereby allowing users to effectively substitute them for non-available AIA services or participating applications.

The CAVS provides a repository that stores these test initiator and simulator definitions created by the CAVS user, as well as an interactive user interface to create and manage the same. Tests can be configured to run individually or in a single-threaded batch.

The CAVS provides value as a testing tool throughout the integration development life cycle:

- Development

  Because integration developers working with AIA are dealing with integrating disparate systems, they typically belong to different teams. To this end, the CAVS provides an effective way to substitute dependencies, letting developers focus on the functionality of their own service rather than being preoccupied with integrations to other services.

- Quality assurance

  The CAVS allows quality assurance engineers to unit and flow test integrations, thereby providing a way to easily certify different pieces of an integration. The reusability of test definitions, simulators, and test groups helps in regression testing and provides a quick way to certify new versions of services.

# 1.2. Describing Key Components of the CAVS Framework

The CAVS framework operates using the following key components:

- Test definition
- Simulator definition

### Test Definition

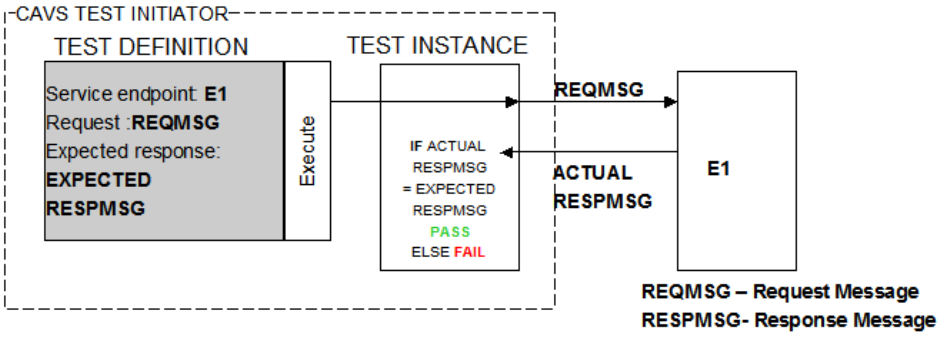
The CAVS test initiator reads test data and feeds it to the web service being tested. You create the test data as a part of a test definition. The test definition is a configuration of the test initiator and contains test execution instructions.

The CAVS user creates a definition using the CAVS user interface (UI) to define the service endpoint URL that needs to be invoked, as well as the request message that will be passed along with metadata about the test definition itself.

**For more information** about creating test definitions, see Creating and Modifying Test Definitions.

The test initiator is a logical unit that executes test definitions to call the endpoint URL defined and creates test instances. This call is no different from any other request initiated by other clients. If the test definition Service Type value is set to Synchronous or Asynchronous two-way, the actual response can be verified against predefined response data to validate the accuracy of the response.

This diagram illustrates the high-level concept of the test initiator:




CAVS Test Definition

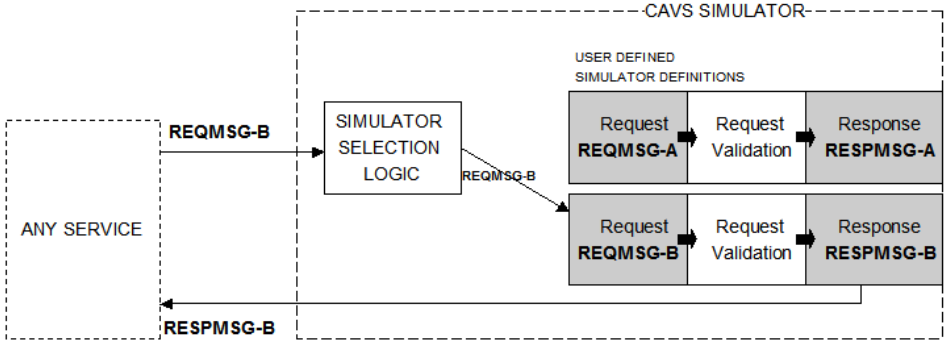
### Simulator Definition

The CAVS simulator is used to simulate a web service. Simulators typically contain predefined responses for a specific request. CAVS users create several simulator definitions, each for a specific set of input.

At runtime, the CAVS simulator framework receives data from the service being tested. Upon receiving the request, CAVS locates the appropriate simulator definition, validates the input against predefined request values, and then returns predefined response data so that the web service being tested can continue processing.

**For more information** about creating simulator definitions, see Creating and Modifying Simulator Definitions.

This diagram illustrates the high-level concept of the CAVS simulator:




CAVS Simulator

# 1.3. Describing the CAVS Design Assumptions and Knowledge Prerequisites

The CAVS operates with the following design assumptions:

- The CAVS assumes that the requester and provider ABCSs it is testing are implemented using BPEL.
- The CAVS is designed to initiate requests and simulate responses as SOAP messages using SOAP over HTTP. The request and response messages that you define in test and simulator definitions must contain the entire XML SOAP document, including the SOAP envelope, message header, and body (payload).
- The correlation logic between the test initiator and the response simulator is based on timestamps only. For this reason, test and simulator instances generated in the database schema will not always be reconcilable, especially when the same web service is invoked multiple times during a very short time period, as in during performance testing.
- The CAVS does not provide or authenticate security information for web services that are initiated by a test initiator or received by a response simulator. However, security information passed through the system by the web service can be used as a part of verification and validation logic.
- When a participating application is involved in a CAVS testing flow, execution of tests can potentially modify data in a participating application. Therefore, consecutive running of the same test may not generate the same results. The CAVS is not designed to prevent this kind of data tampering because it supports the user's intention to include a real participating application in the flow. The CAVS has no control over modifications that are performed in participating applications.

  This issue does not apply if your CAVS test scenario uses test definitions and simulator definitions to replace all participating applications and other dependencies. In this case, all cross-reference data is purged once the test scenario has been executed. This enables rerunning of the test scenario.

**Note.** CAVS cross reference data is purged at the end of a test execution when executing a test definition and at the end of a test group execution when executing a test group definition. Therefore, if you want to execute test definitions that are dependent on cross referencing data created by earlier test executions, ensure that you include all dependent test definitions in a test group and execute the test group.

**For more information** about how to make test scenarios rerunnable, see Purging CAVS-Related Cross Reference Entries to Enable Rerunning of Test Scenarios.

To work effectively with the CAVS, users must have working knowledge of the following concepts and technologies:

- AIA
- XML
- XPath
- SOAP

# 2. Preparing to Use the CAVS

Before you start creating and running tests in the Composite Application Validation System (CAVS), take the time to gather your test requirements and plan your approach to using CAVS.

This chapter provides a high-level discussion of the following questions you should answer to help gather requirements for the tests you want to create and run in the CAVS.

- What Can I Test Using CAVS?
- What Are the Oracle AIA Components That I Need to Test?
- Which Message Exchange Pattern Is Being Used by the Components Being Tested?
- Does the Scenario Need to be Unit or Flow Tested?
- Do I Have the Content I Need to Create the Definitions?

## 2.1. What Can I Test Using CAVS?

The CAVS supports the following testing scenarios:

- Create and execute test definitions against actual services in participating applications.
- Create and execute test definitions that call services that call simulators, which simulate actual services in participating applications.
- Use actual services in participating applications in cooperation with simulators to simulate any unavailable services.

## 2.2. What Are the Oracle AIA Components That I Need to Test?

Examine the components involved in the scenario that you need to test. Which of the following components does the scenario include?

- Requester Application Business Connector Services (ABCSs)
- Provider ABCSs
- Enterprise Business Flows (EBFs)
- Exposed services in participating applications

# 2.3. Which Message Exchange Pattern Is Being Used by the Components Being Tested?

Once you have assessed which components you need to test, identify the message exchange pattern (MEP) being used by the components to help you determine which types of CAVS test and simulator definitions you need to create. Based on the sequence of service calls and the MEPs employed, you can determine if you need to use synchronous, notify, or asynchronous two-way test definitions and/or simulator definitions.

This section discusses CAVS process flows for testing the following MEPs:

- Synchronous (request-and-response)
- Asynchronous (notify)
- Asynchronous two-way

**Note.** The information in this chapter provides CAVS processing details that can inform your creation of test and simulator definitions in the CAVS user interface (UI). As you prepare to define and run tests for a particular web service, refer to the section in this chapter that corresponds to the message exchange pattern of the service you want to test.

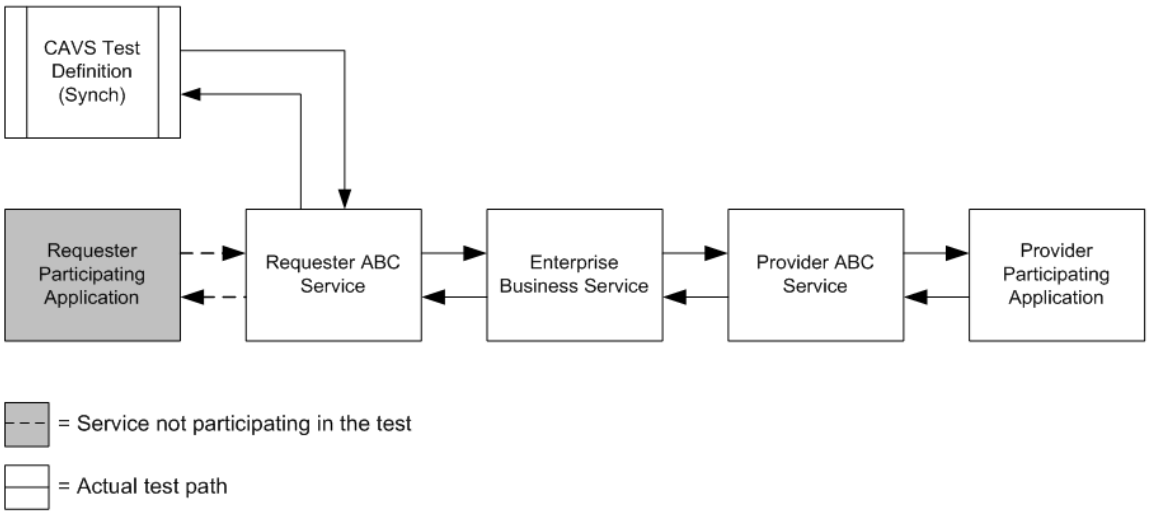
## 2.3.1. Describing CAVS Process Flows for Testing the Synchronous Message Exchange Pattern

The following diagrams describe CAVS process flows for testing a provider ABCS using a synchronous MEP.

These sample flows can be used as the basis for testing other artifacts as well, such as requester ABCSs, EBFs, or provider services.

### Synchronous MEP Testing Flow Using a Test Definition

The requester participating application is replaced by the CAVS test definition. The test definition points to the URL of the requester ABCS. It uses a composed request message to invoke the ABCS and expects a message in response.



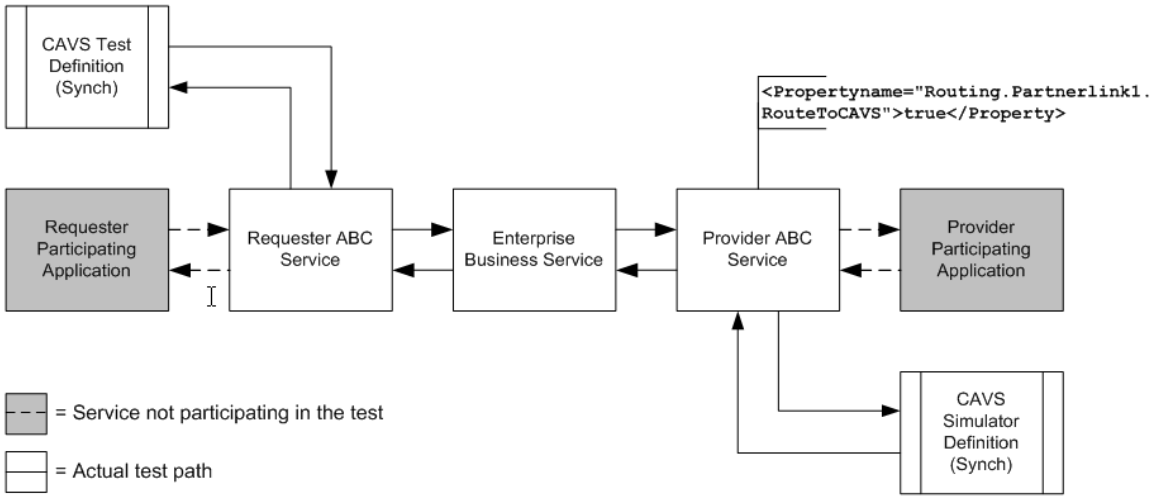

## Testing a synchronous MEP using a CAVS test definition

### Synchronous MEP Testing Flow Using a Test Definition and Simulator Definition

The requester participating application is replaced by the CAVS test definition. The test definition points to the URL of the requester ABCS. It uses a composed request message to invoke the ABCS and expects a message in response.

The provider participating application is replaced by the CAVS simulator definition. The provider ABCS is programmed to route to this simulator instead of the provider participating application. The simulator definition contains a predefined request and response message pair.

The simulator definition performs validations on message input from the provider ABCS and sends the message back to the provider ABCS. The provider ABCS sends the message back to the test definition, which validates this actual response against its predefined expected response.

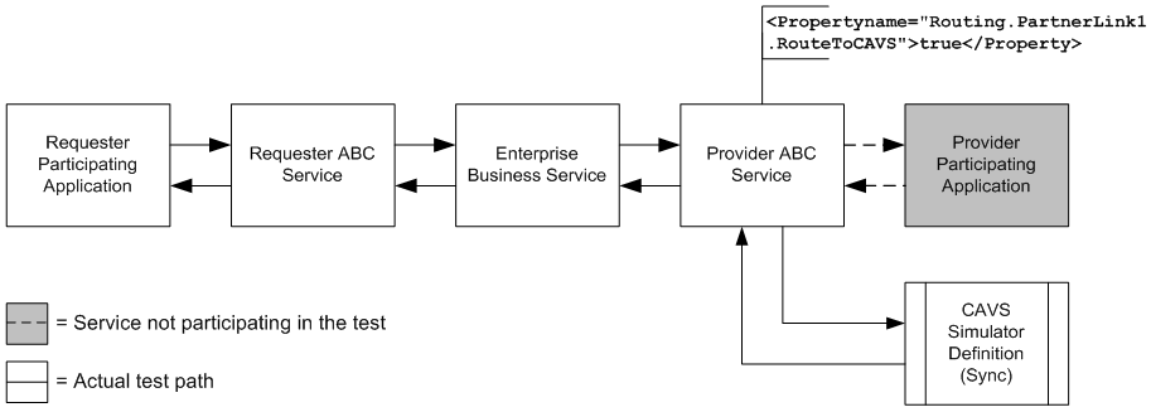



## Testing a synchronous MEP using a CAVS test definition and simulator definition

### Synchronous MEP Testing Flow Using a Simulator Definition

The provider participating application is replaced by the CAVS simulator definition. The provider ABCS is programmed to route to this simulator instead of the provider participating application. The simulator definition contains a predefined request and response message pair.

The simulator definition performs validations on message input from the provider ABCS and sends the message back to the provider ABCS. The provider ABCS sends the message back to requester participating application.




Testing a synchronous MEP using a CAVS simulator definition

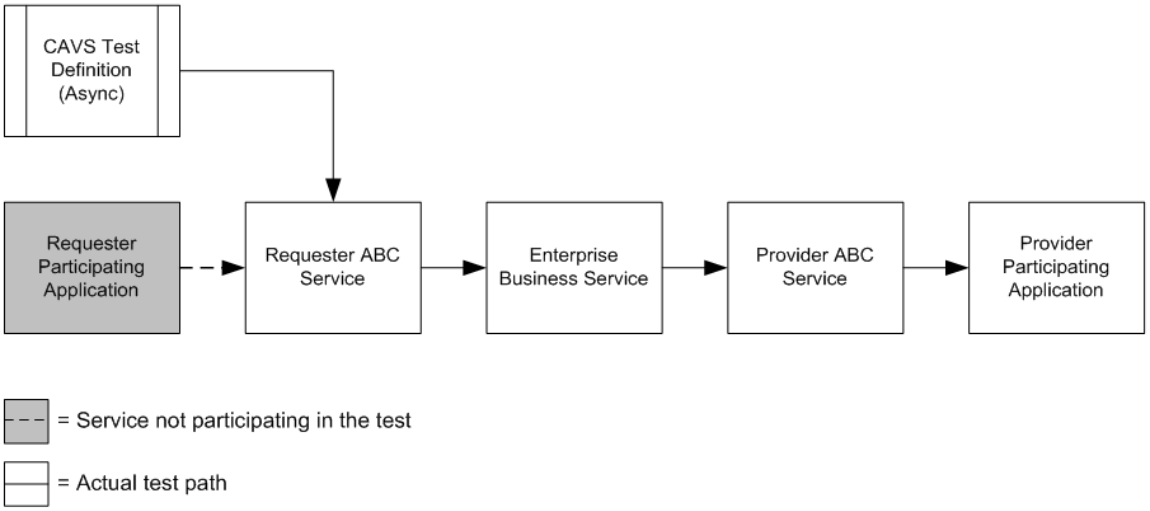
## 2.3.2. Describing CAVS Process Flows for Testing the Asynchronous (Notify) Message Exchange Pattern

The following diagrams describe CAVS process flows for testing a provider ABCS using an asynchronous (notify) MEP.

These sample flows can be used as the basis for testing other artifacts as well, such as the requester ABCS, EBF, or the provider service itself.

### Asynchronous (Notify) MEP Testing Flow Using a Test Definition

The requester participating application is replaced by the CAVS test definition. The test definition points to the URL of the requester ABCS. It uses a composed request message to invoke the ABCS and does not expect a message in response.



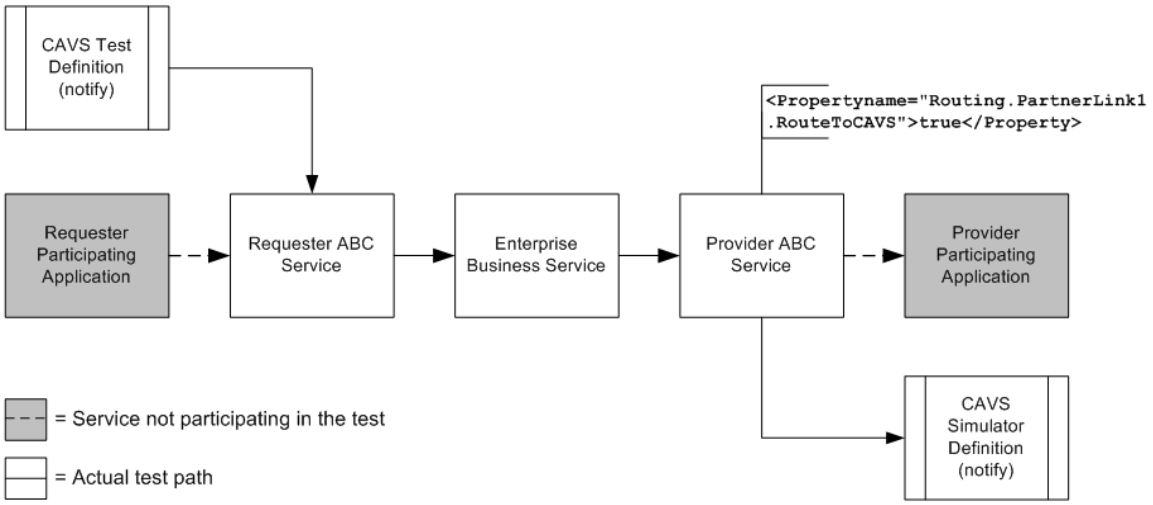

Testing an asynchronous (notify) MEP using a CAVS test definition

### Asynchronous (Notify) MEP Testing Flow Using a Test Definition and Simulator Definition

The requester participating application is replaced by the CAVS test definition. The test definition points to the URL of the requester ABCS. It uses a composed request message to invoke the ABCS and does not expect a message in response.

The provider participating application is replaced by the CAVS simulator definition. The provider ABCS is programmed to route to this simulator instead of the provider participating application. The simulator definition contains a predefined expected request message.

The simulator definition performs validations on message input from the provider ABCS.

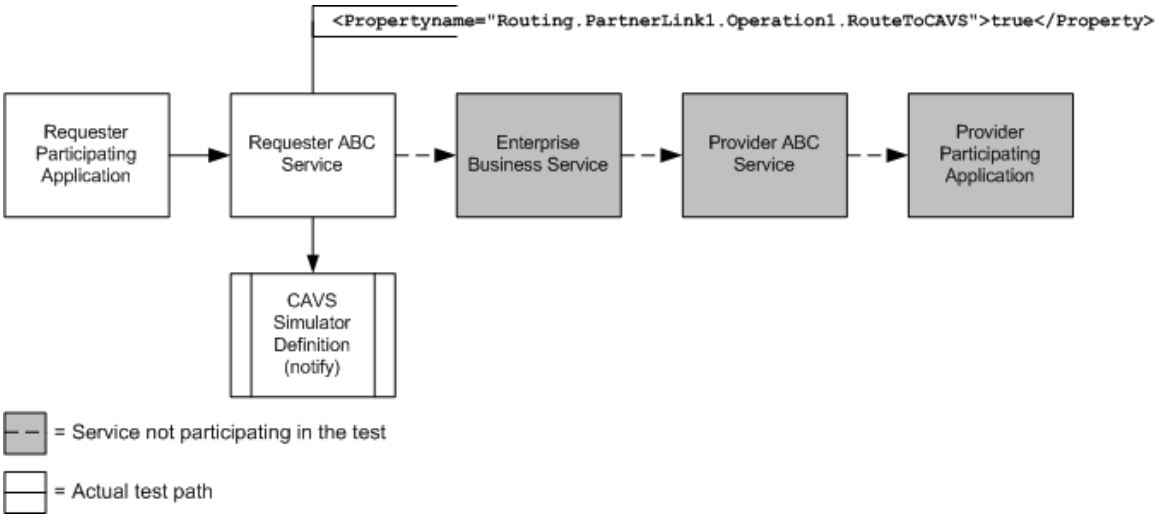



Testing an asynchronous (notify) MEP using a CAVS test definition and simulator definition

### Asynchronous (Notify) MEP Testing Flow Using a Simulator Definition

The provider participating application is replaced by the CAVS simulator definition. The requester ABCS is programmed to route to this simulator instead of the provider participating application. The simulator definition contains a predefined expected request message.

The simulator definition performs validations on message input from the provider ABCS




Testing an asynchronous (notify) MEP using a CAVS simulator definition

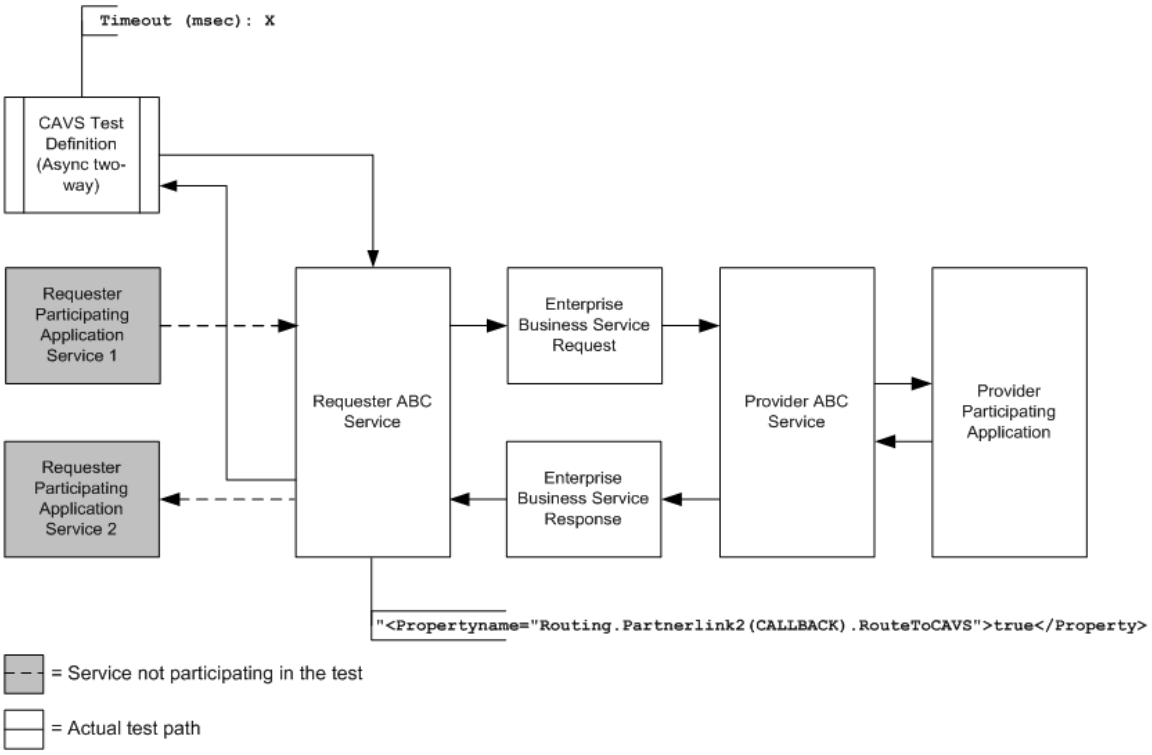
## 2.3.3. Describing Flows for Testing the Asynchronous Two-Way Message Exchange Pattern

The following diagrams describe CAVS process flows for testing a provider ABCS using an asynchronous two-way MEP.

These sample flows can be used as the basis for testing other artifacts as well, such as the requester ABCS, EBF, or the provider service itself.

### Asynchronous Two-Way MEP Testing Flow Using a Test Definition

The requester participating application is replaced by the CAVS test definition. The test definition points to the URL of the requester ABCS. It uses a composed request message to invoke the ABCS and expects an eventual message in response. The test definition includes a timeout value. If no response message is received within this timeout value, the test definition will experience a timeout failure.



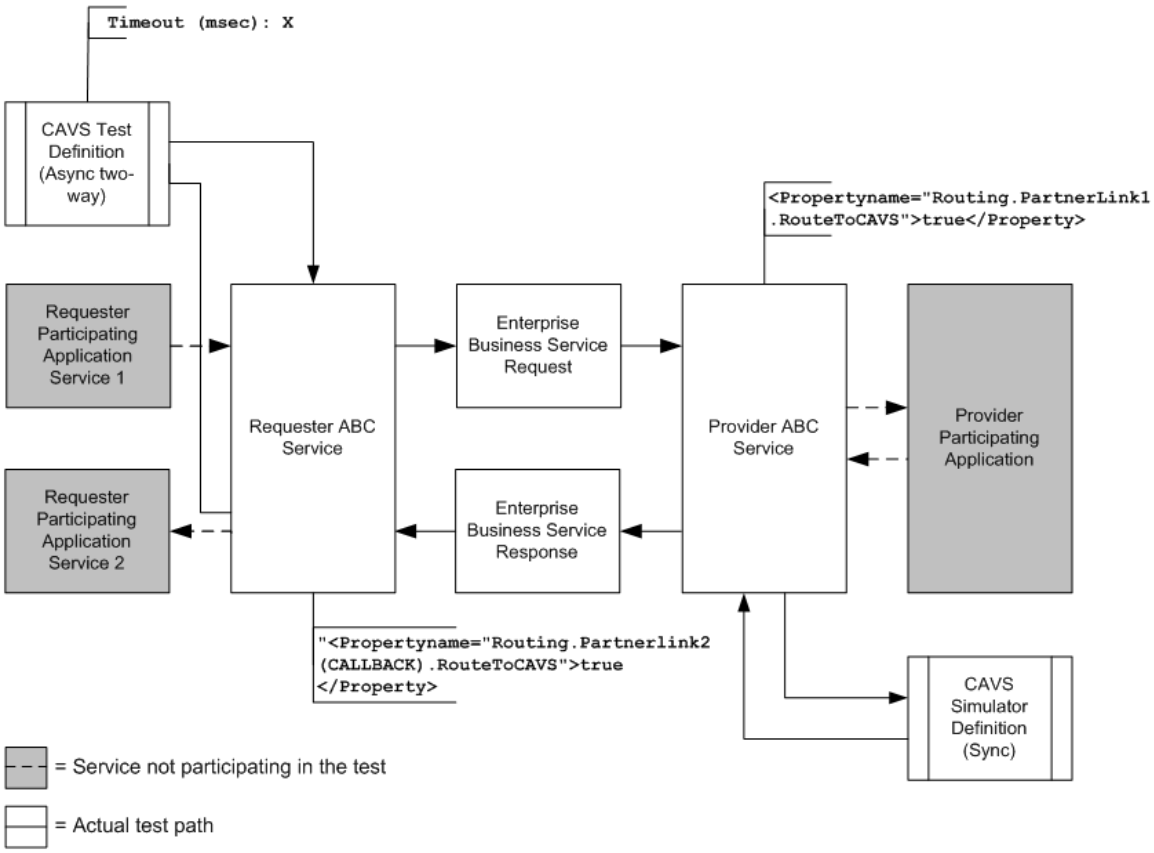

Testing an asynchronous two way MEP using a CAVS test definition

### Asynchronous Two-Way MEP Testing Flow Using a Test Definition and Simulator Definition

The requester participating application is replaced by the CAVS test definition. The test definition points to the URL of the requester ABCS. It uses a composed request message to invoke the ABCS and expects an eventual message in response. The test definition includes a timeout value. If no response message is received within this timeout value, the test definition will experience a timeout failure.

The provider participating application is replaced by the CAVS simulator definition. The provider ABCS is programmed to route to this simulator instead of the provider participating application. The simulator definition contains a predefined request and response message pair.

The simulator definition performs validations on message input from the provider ABCS and sends the message back to the provider ABCS. The provider ABCS sends the message back to the test definition, which validates this actual response against its predefined expected response.



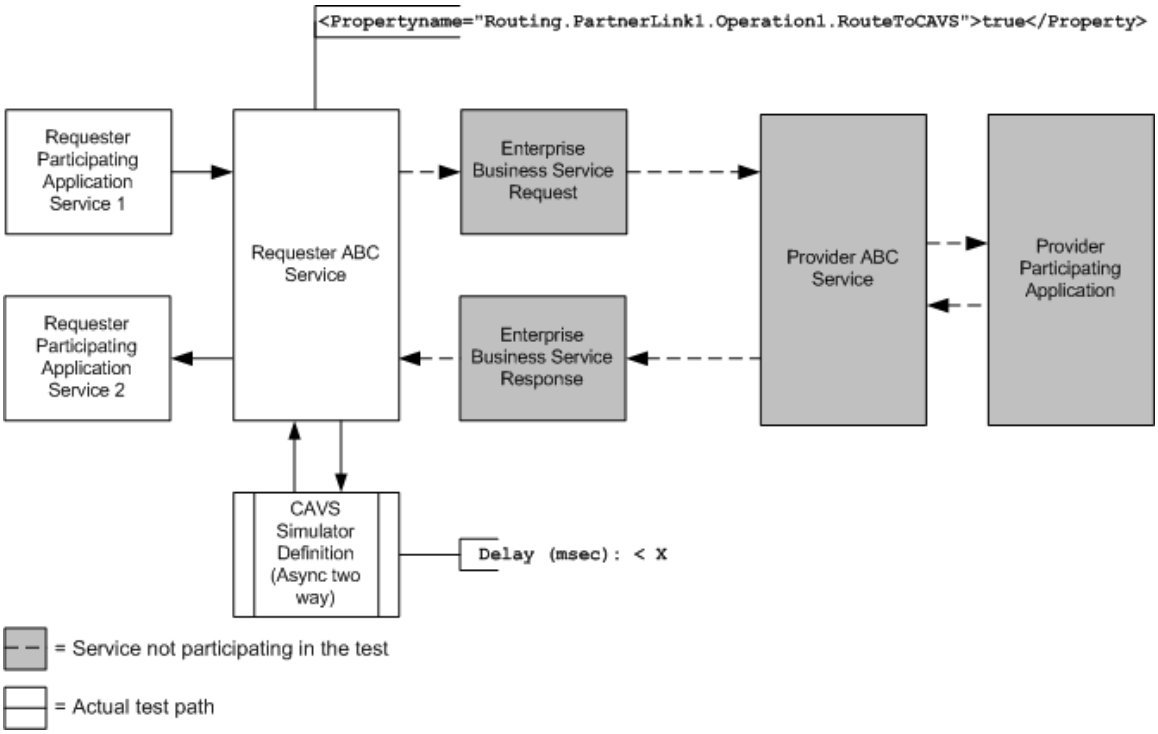

## Testing an asynchronous two way MEP using a CAVS test definition and simulator definition

### Asynchronous Two-Way MEP Testing Flow Using a Simulator Definition

The provider ABCS is replaced by the CAVS simulator definition. The requester ABCS is programmed to route to this simulator instead of having the flow reach the provider ABCS.

The simulator definition contains a predefined request and response message pair, as well as a user-defined delay value. The simulator definition will delay its response by this amount of time to simulate the asynchronous two-way nature of the provider participating application.

The simulator definition performs validations on message input from the requester ABCS and sends the message back to the requester ABCS.




Testing an asynchronous two way MEP using a CAVS simulator definition

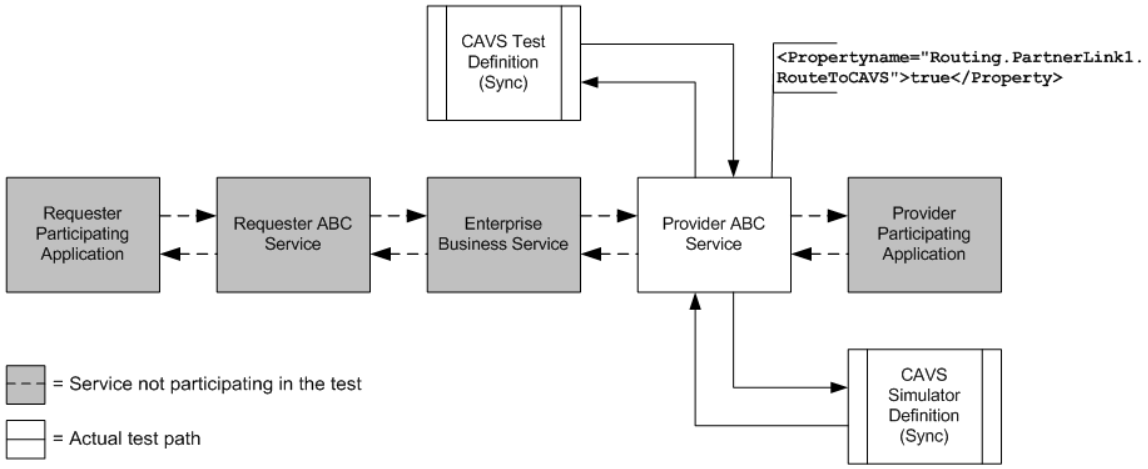
# 2.4. Does the Scenario Need to be Unit or Flow Tested?

This section discusses different configurations for test and simulator definitions to achieve unit and flow tests.

## 2.4.1. Describing a Unit Test Configuration

This section will use a synchronous provider ABCS as the focus of the test example. However, this test configuration is not specific to message exchange patterns, so it can be applied to asynchronous (notify) and asynchronous two-way components as well.

To unit test a component, place a test definition before the component and a simulator definition after it. This isolates the focus of the test to the single component.

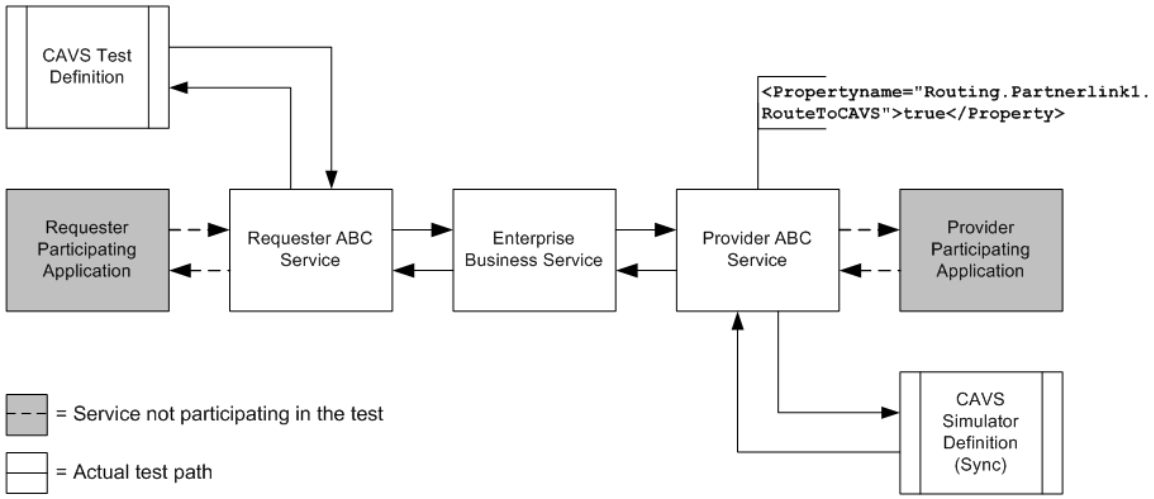



Unit testing a provider ABCS

## 2.4.2. Describing a Flow Test Configuration

This section will use a synchronous provider ABCS as the focus of the test example. However, this test configuration is not specific to message exchange patterns, so it can be applied to asynchronous (notify) and asynchronous two-way components as well.

Once you have unit tested the components in a scenario, you can flow test the scenario. To flow test a scenario, place a test definition before the requester ABCS at the front of the scenario and a simulator definition after the provider ABCS at the end of the scenario.

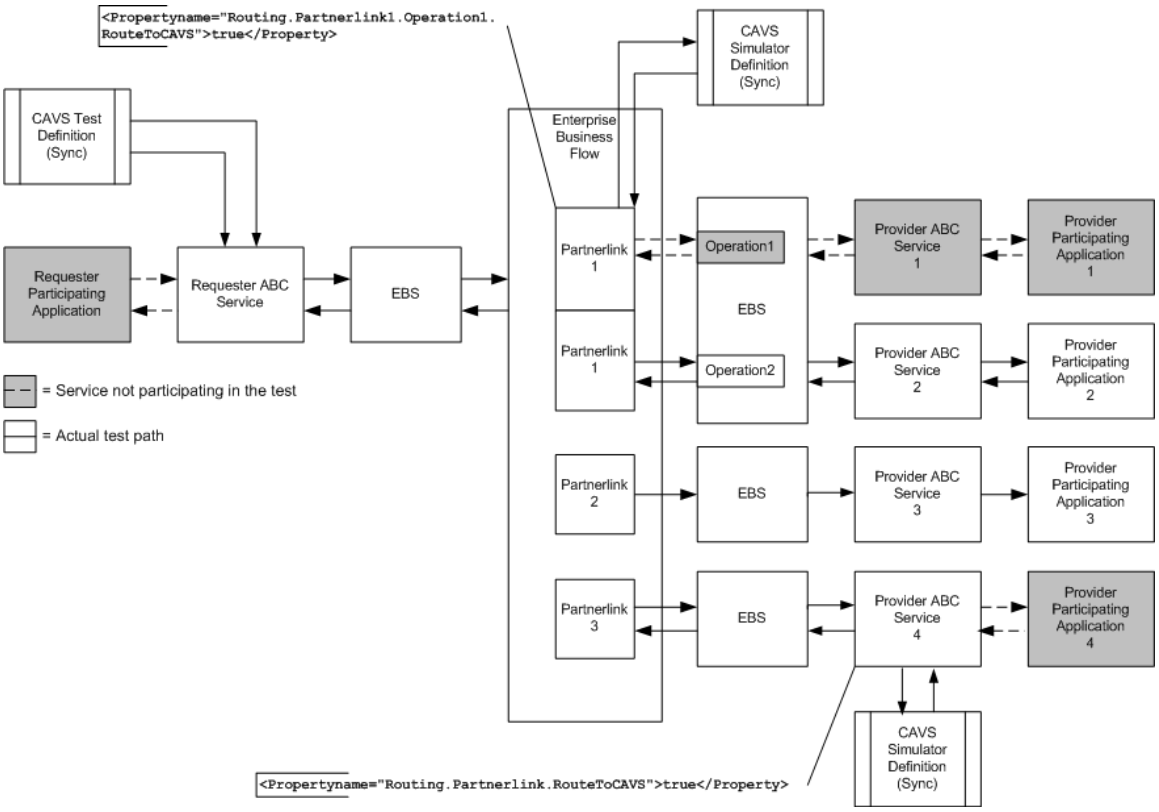



Flow testing a scenario

## 2.4.3. Describing a Complex Flow Test Configuration

This section will use an EBF as the focus of the test example. However, this test configuration is not specific to EBFs, so it can be applied to any service that conducts chatty conversations.

You can place a test definition before the requester ABCS at the front of a scenario and enable the EBF to make calls out to simulator definitions whenever required. You can then go on to replace some of the provider applications with simulators at the end of the scenario.




Complex flow testing an EBF

# 2.5. Do I Have the Content I Need to Create the Definitions?

Once you know what you need to test and which CAVS definitions you need to create, assess whether or not you have all of the content you need to create the definitions.

To create your test definitions and simulator definitions, you will primarily need request and response XML text.

If you are creating a Test Definition (or an asynchronous two-way simulator), you will need the endpoint URL of the web service you are testing.

The endpoint URL value can be found in the WSDL of the web service that you want to test.

When the endpoint URL is provided, CAVS will present you with available SOAP actions. Once you select the required SOAP action, CAVS will automatically generate the message stub for the service being called. You can then include data within the XML tags generated.

In either case, you can use the BPEL Console to obtain request and response XML text. Run the processes you are testing at least once with all participating applications and services in place and with the desired results. The XML messages generated by this successful run of the processes will provide your request and response XML for test and simulator definitions. The following section describes how to use the BPEL console to obtain these XML messages.

**Note.** Obtaining response message XML text is only applicable when testing synchronous and asynchronous two-way processes.

### 2.5.1. How to Obtain Message XML Text from a BPEL Process

To obtain request and response message XML text:

1. Access the BPEL Console for your AIA implementation.
2. Click the Instances tab.
3. In the BPEL Process drop-down list box, select the BPEL process for which you are creating a test definition and click Go. BPEL process instances for the selected BPEL process display in the List of BPEL Process Instances frame. Sort by ***Last Modified,*** if you want to access the most recent instance.
4. Click the link for the instance you want to use for your request and response XML message text.
5. Click the Flow link.

**Note.** If the instance you selected contains any faults, you may want to consider selecting a different instance. However, if you are trying to test fault messaging in the BPEL, you must select a BPEL process instance that contains the fault.

6. To obtain the Request Message XML text:
   a. Click the receiveInput element to get your test definition request message XML text.
   b. Click the Copy details to clipboard link at the bottom of the pop-up box displaying input Variable data.
   c. Open an XML editor and paste the XML text into a blank document.
   d. Remove the opening and closing ***inputVariable*** (***XYZ_InputVariable,*** in the case of a non-BPEL service, such as a participating application service) and ***part*** elements.
   e. Copy and paste the remaining XML text in the default SOAP envelope provided in the Request Message field on the Test Definition page or Simulator Definition page. Paste the XML text into the area indicated by the “Paste your SOAP Message Content here” placeholder text.
7. To obtain the Response Message XML text:

**Note.** Obtaining response message XML text is only applicable when testing a synchronous process.

   a. Click the **r**eply Output element to get your test definition response message XML text.
   b. Click the Copy details to clipboard link at the bottom of the pop-up box displaying output Variable data.
   c. Open an XML editor and paste the XML text into a blank document.
   d. Remove the opening and closing ***inputVariable*** (***XYZ_InputVariable,*** in the case of a non-BPEL service, such as a participating application service) and ***part*** elements.
   e. Copy and paste the remaining XML text in the default SOAP envelope provided in the Response Message field on the Test Definition page or Simulator Definition page.

# 3. Introduction to Defining and Running CAVS Tests Using the CAVS UI

This chapter discusses the following topics:



## 3.1. Describing the CAVS UI

The Composite Application Validation System (CAVS) enables you to configure test data, execute tests, review test results, and migrate tests using the following user interface (UI) components.

### Test Definitions

A test definition is a configuration of a single execution of the test initiator service. The test definition stores test data and test execution instructions. A test definition can be executed alone, or in a single-threaded batch as a part of a group definition.

You will find that the values you set for a test definition and simulator definition are similar. The test definition differs from the simulator definition in that it is an active participant in the CAVS framework, initiating tests. The test definition carries the following values that are not a part of the simulator definition. These values inform the active state of the test definition:

- Endpoint URL
- SOAP Action

**For more information** about test definitions, see Creating and Modifying Test Definitions.

If required by the business service pattern of the web service you are testing, you can assign a simulator definition to the test definition.

### Simulator Definitions

A simulator definition is a configuration of a single execution of response simulator service. The simulator definition simulates a web service and receives data from the tested web service and returns previously defined data so that the tested web service can continue processing.

You will find that the values you set for a simulator definition and test definition are similar. The simulator definition differs from the test definition in that it is a passive participant in the CAVS framework, awaiting initiation.

The simulator definition carries the following additional XPath attributes that are not a part of the test definition. These values participate in simulator definition request matching:

- Is Node Key

- Key Node Value

**For more information** about simulator definitions, see Creating and Modifying Simulator Definitions.

The success of the test is verified based on the simulator definition's previously defined data being accurately returned and matched to the expected response results defined in the test definition.

### Group Definitions

A group definition is a configuration of a single execution of one or more test definitions in a single-threaded batch.

### Test Instances

A test instance captures the details of the execution of a test definition.

### Simulator Instances

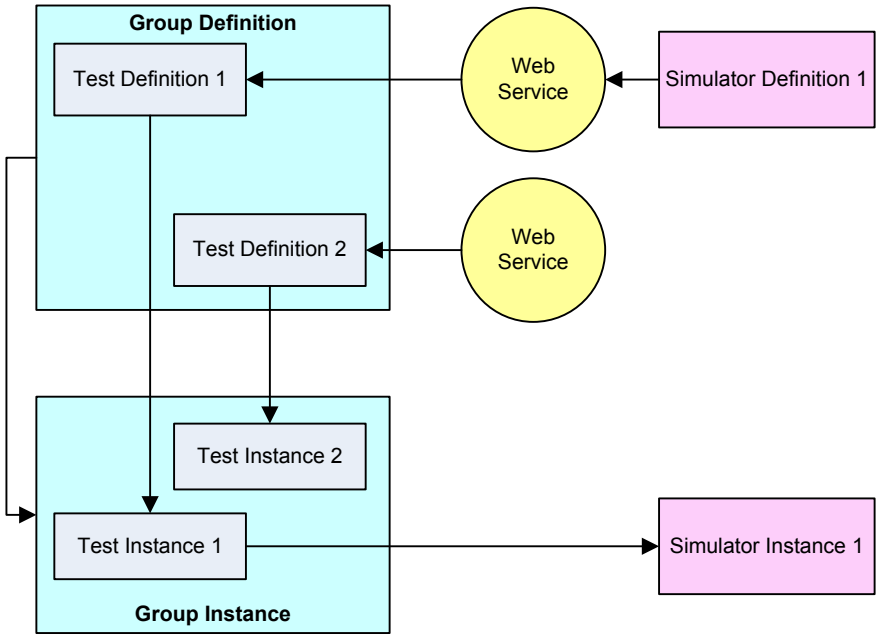
A simulator instance captures the details of a simulator definition's behavior during the execution of a test definition with which it is associated.

### Group Instances

A group instance captures the details of the execution of a group definition.

### Component Overview

This diagram provides a high-level overview of the relationships among the CAVS components discussed in this section.




Overview of CAVS component relationships

## 3.2. Overview of Defining and Running CAVS Tests

This high-level procedure provides the steps involved in defining and running tests using the CAVS UI.

1. Assess your test requirements and gather required content.

   **For more information,** see Preparing to Use the CAVS.

2. If your test requires a test definition, access the Create Test page to create your test definition.

   **For more information,** see Creating and Modifying Test Definitions.

3. If your test requires a simulator definition, access the Create Simulator page to create your simulator definition.

   **For more information,** see Creating and Modifying Simulator Definitions.

4. If your test requires a test definition and simulator definition, you can link their definitions on either Modify Test Definition page or Modify Simulator Definition page.
5. If your test requires multiple (single-threaded) executions of the same or different test definitions, create a group definition on the Create Group Definition page.

   **For more information,** see Working with Group Definitions.

6. If your test or group definition utilizes a simulator definition, you must set the Application Business Connector Service (ABCS) being tested to route to the response simulator. To do this:
   a. Access the Routing Setup page.
   b. Create a routing setup ID for the invoking service being tested.
   c. Associate this routing setup ID with the test definition being used to test your scenario.

      **For more information** about routing setup IDs, see Defining CAVS Routing Setup IDs.

   d. Alternatively, you can quickly set up a routing configuration on the Configurations page.

      **For more information** about routing configurations, see How to Set Up CAVS Routing Configurations Without Creating Routing Setup IDs.

7. To run a single test, access the test definition on the Definitions page or Modify Test Definitions page to run the test.

   **For more information,** see Searching for Test and Simulator Definitions.

   To run a group test, access the Group Definition page or Group Definition Detail page to run the group test.

   **For more information,** see Working with Group Definitions.

8. Once an individual test has been executed, view test results generated for the test instance on the Test Instances Detail page. If the test definition includes an associated simulator definition, view the simulator instance on the Simulator Instances Detail page.

**For more information,** see Working with Test and Simulator Instances.

Once a group of tests has been executed, view the test results generated for the group instance on the Group Instance Detail page.

**For more information,** see Working with Group Instances.

9. Once testing is complete for a test that involved a simulator definition for which you defined a routing configuration on the Configurations page, be sure to reset the ABCS tested to return to routing to its usual production destination and no longer route to the response simulator. To do this:

   a. Access the Configurations page.

   b. Clear the Route To CAVS option.

   c. Click Reload.

For **more information** about routing configurations, see How to Set Up CAVS Routing Configurations Without Creating Routing Setup IDs.

## 3.3. How to Execute CAVS Definitions as Web Services

The CAVS provides a web service that enables you to execute test definitions and test group definitions without the use of the CAVS user interface.

You can call this CAVS web service from any system: http://<hostname>:<port>/AIAValidationSystemAPIService/AIAValidationSystemAPIServiceSoapHttpPort.

This web service provides two operations:

- executeDefinition

  Executes a given test definition ID.

- executeGroupDefinition

  Executes a given test group definition ID.

Typically, these operations can be consumed by third-party testing tools or other systems to execute test definitions and test group definitions whenever desired, without the use of the CAVS UI.

The WSDL that defines the service contract is: http://<hostname>:<port>/AIAValidationSystemAPIService/AIAValidationSystemAPIServiceSoapHttpPort?wsdl.

## 3.4. How to Execute CAVS Definitions Using ANT

CAVS provides an ANT script that enables you to execute test definitions without the use of the CAVS user interface. This functionality is useful when trying to automate test execution as a part of automated deployment processes.

CAVS test definitions can also be executed as web services.

**For more information** about the CAVS web service, see How to Execute CAVS Definitions as Web Services.

To execute a CAVS definition using ANT:

1. Define your test definition using CAVS. In this example, the test definition ID is ***601***.
2. Navigate to AIAHOME/Infrastructure/CAVS.
3. Run `source AIAHOME/bin/aiaenv.sh`.
4. Run `ant -f AIACAVSInvoke.xml -Did=601`.

# 4. Creating and Modifying Test Definitions

A test definition is a configuration of a single execution of the test initiator service. The test definition stores test data and test execution instructions. A test definition can be executed alone, or in a single-threaded batch as a part of a group definition.

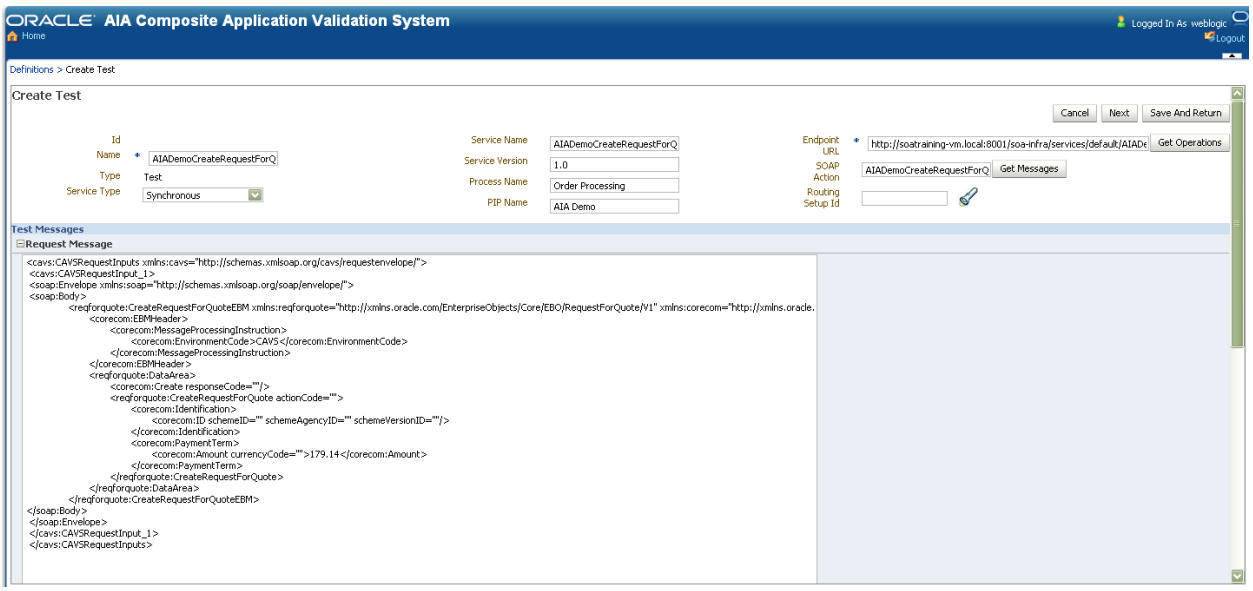
This chapter discusses

- How to Create a Test Definition
- How to Modify a Test Definition
- How to Provide Multiple Request and Response Message Sets in a Single Test Definition

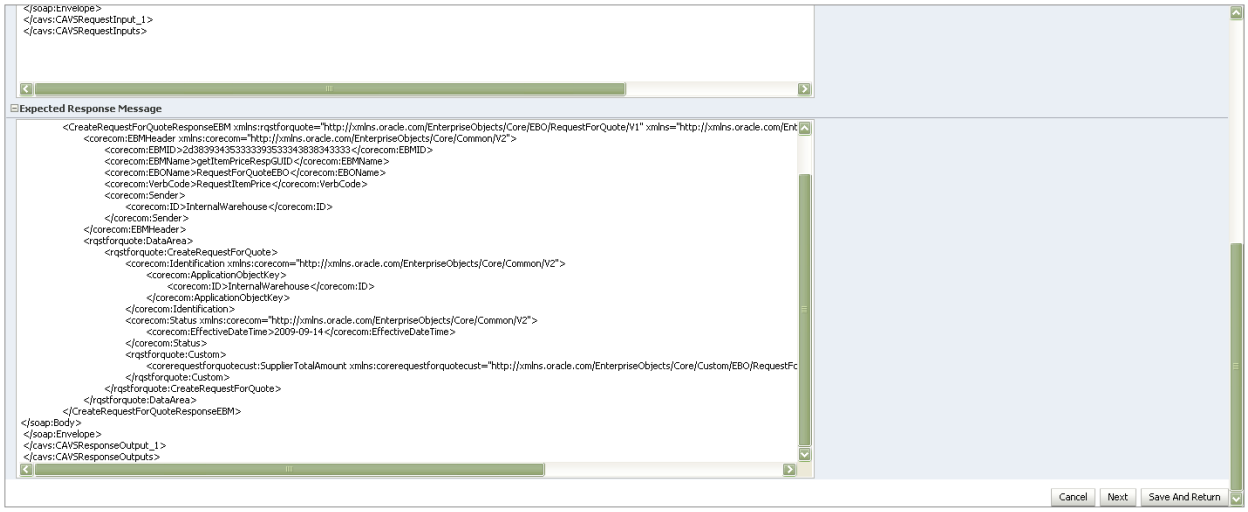
## 4.1. How to Create a Test Definition

To create a test definition:

1. Access the Create Test page. To access the page, access the Oracle Application Integration Architecture (AIA) Home Page. In the Composite Application Validation System area, click the Go button. Select the Definitions tab. Click the Create Test button.




Create Test page (1 of 2)




Create Test page (2 of 2)

**2.** On the Create Test page, use the following page elements to create test definitions.

| | |
|---|---|
| **Id** | Upon saving the test definition, displays a unique key identifier that is assigned to the test definition. |
| **Name** | Enter a descriptive name that you want to use for the test definition. |
| **Type** | Displays the type of definition you chose to create. On the Create Test page, this value will always be set to ***Test.*** |
| **Service Type** | Select the business service pattern of the web service that you want to test using the test definition.<br><br>***Synchronous*** (request-and-reply)<br><br>***Notify*** (asynchronous request-only)<br><br>***Asynchronous two way*** |
| **Service Name** | Enter the name of the web service that you want to test using the test definition. This is the name of the web service being called by the URL provided in the **Endpoint URL** field. |
| **Service Version** | Enter the version of the web service that you want to test using the test definition. This is the version of the web service being called by the URL provided in the **Endpoint URL** field. |
| **Process Name** | Enter the name of the process that includes the web service that you want to test using the test definition. |
| **PIP Name (Process Integration Pack name)** | Enter the name of the PIP that includes the web service that you want to test using the test definition. |
| **Endpoint URL** | Enter the URL of the web service that you want to test using the test definition. The endpoint URL value can be found in the WSDL of the web service that you want to test. |
| **Get Operations** | Click to display the list of operations supported by the WSDL associated with the **Endpoint URL** value you provided. Supported operations display in the Select WSDL Operations window. |

| | |
|---|---|
| | Select the operation that you want to test using the test definition. The selected operation displays in the **SOAP Action** field. |
| **SOAP Action** | If you clicked **Get Operations** to select an operation in the Select WSDL Operations window, selected operation displays here.<br><br>Alternatively, you can manually enter the operation called by the web service that you want to test using the test definition. The value you enter must match an action provided in the WSDL of the web service that you want to test. |
| **Get Messages** | Click to generate a request stub message for the operation specified in the **SOAP Action** field. For test definitions with the **Service Type** field set to ***Synchronous***, the response stub message will also be generated. |
| **Routing Setup Id** | Select a routing configuration that you want to use for the test. |

**For more information** about routing configurations, see Defining CAVS Routing Setup IDs.

## Test Messages

Use the **Test Messages** group box to enter request and response XML message text. By default, SOAP envelope XML text is provided in these fields. You can use the **Get Messages** button to generate request and response stub messages based on selected endpoint URL and operation values. Alternatively, you can paste XML text within this default SOAP envelope or paste your own XML text already enclosed in an envelope into these fields.

**For more information** about obtaining request and response XML message text, see How to Obtain Message XML Text from a BPEL Process.
**For more information** about how to create test request and response messages that hold multiple sets of test data in a single definition, see How to Provide Multiple Request and Response Message Sets in a Single Test Definition.

| | |
|---|---|
| **Request Message** | Entering request message XML text for a test definition is required, whether the **Service Type** field value is set to ***Synchronous, Notify,*** or ***Asynchronous two way.***<br><br>When you first access the Create Test page, the **Request Message** text box is populated with a SOAP stub message.<br><br>You can use the **Get Messages** button to generate a request stub message based on selected endpoint URL and operation values.<br><br>If you are manually entering your request message, the “Paste your SOAP Message Content here” text in the stub message indicates where you should paste your actual request message text. This request message should mimic the XML message text sent by the service that normally initiates the service. |
| **Expected Response Message** | The ability to enter response message XML text is available when the **Service Type** field value is set to ***Synchronous*** or ***Asynchronous two way.***<br><br>When you first access the Create Test page, the **Expected Response Message** text box is populated with a SOAP stub |

| | |
|---|---|
| | message.<br><br>For test definitions with the **Service Type** field set to ***Synchronous***, the response message stub will have been generated when you clicked **Get Messages** during request message generation.<br><br>If you are manually entering your request message, the "Paste your SOAP Message Content here" text in the stub message indicates where you should paste your actual response message text. Enter a response message that is the expected response message XML. This facilitates the generation of XPath values, which are used to validate the actual response message returned in the test. You may also choose to manually enter or modify the XPath values directly on the Modify Test Definition page. If you are manually entering XPath values, you do not need to enter response message XML text.<br><br>When you enter response message XML text on this page, you can click **Generate Xpath** on the Modify Test Definition page to generate the XPath values that will be used to validate the expected response message you entered on this page against the actual response returned by the test.<br><br>If the **Service Type** field value is set to ***Synchronous*** or ***Asynchronous two way,*** you may choose to not enter response message XML text in this field. You do not need to enter response message XML if you are manually entering XPath values directly on the Modify Test Definition or if the test you are running does not require validation of the response message. For example, your test may be focused on just populating data.<br><br>The **Expected Response Message** text box is unavailable when the **Service Type** field value is set to ***Notify.*** |
| **Cancel** | Click to exit the page and return to the Definitions page. |
| **Next** | Click to save entries on the Create Test page and go to the Modify Test Definition page, where you can further edit your test definition, generate XPaths, and execute the test. |
| **Save** | Click to save entries on the Create Test page and return to the Definitions page. |

# 4.2. How to Modify a Test Definition

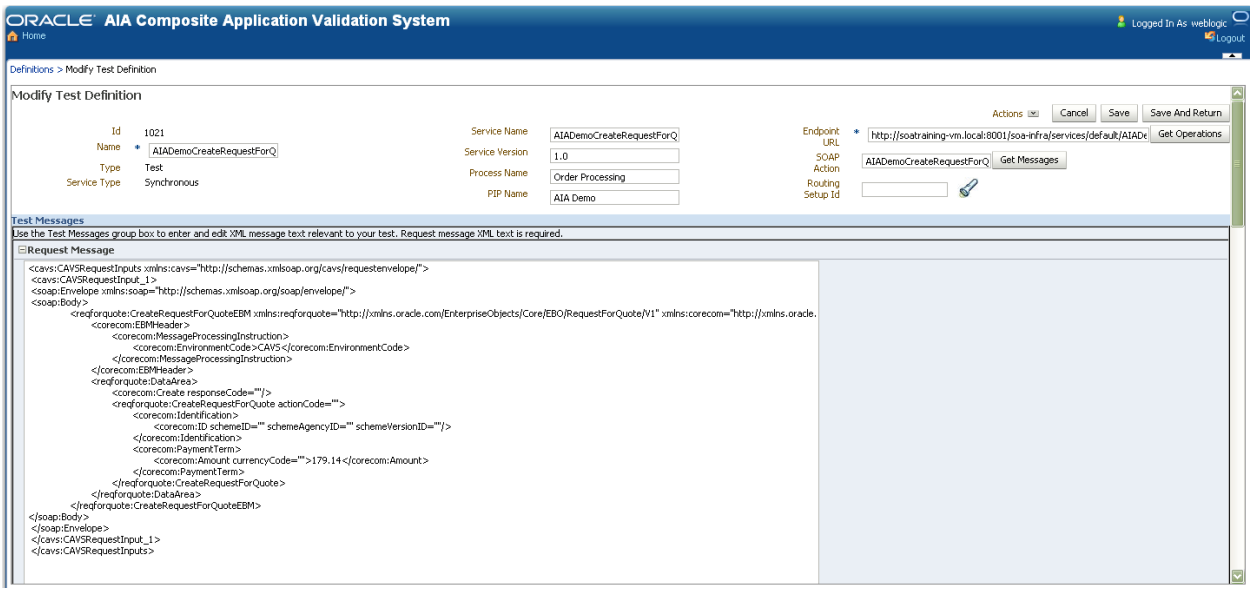
To modify a test definition:

1. Access the Modify Test Definition page. To access the page, use one of the following navigation paths:

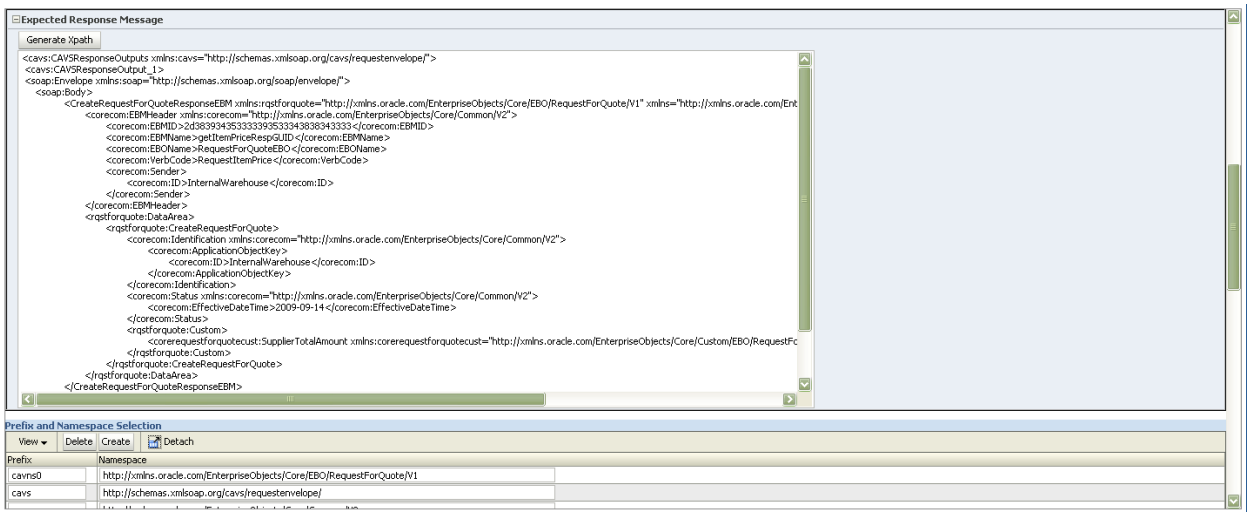
   - Enter required values on the Create Test page and click Next. To access the Create Test page, access the AIA Home Page. In the Composite Application Validation System area, click the Go button. Select the Definitions tab. Click the Create Test button.
   - Click an Id link for an unlocked test definition in the Search Result Selection grid on the Definitions page. To access the Definitions page, access the AIA Home Page. In the Composite

Application Validation System area, click the Go button. Select the Definitions tab.

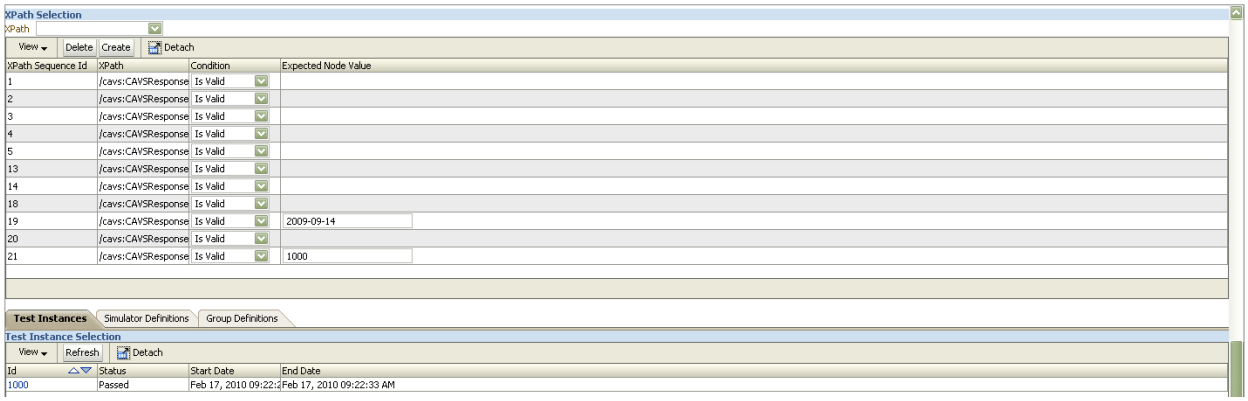
- Click a Definition Id link for an unlocked test definition on the Instances page. To access the Instances page, access the AIA Home Page. In the Composite Application Validation System area, click the Go button. Select the Instances tab.
- Click a Definition Id link for an unlocked test definition on the Test Instances Detail page. To access the Test Instances Detail page, access the AIA Home Page. In the Composite Application Validation System area, click the Go button. Select the Instances tab. Click an instance ID link.




Modify Test Definition page (1 of 3)




Modify Test Definition page (2 of 3)




Modify Test Definition page (3 of 3)

**2.** Use the following page elements on the Modify Test Definition page to modify an existing test definition and execute and manage existing test definitions.

The page displays values you have defined for the test definition. You can modify the values in editable fields.

**For more information,** see How to Create a Test Definition.

**Time-out (msec) (in milliseconds)**

This field displays only for a test definition with a **Service Type** value of ***Asynchronous two way.***

Enter the number of milliseconds that you want the test definition to remain available for the asynchronous reply before timing out. If this length of time passes before the asynchronous response is returned, a failure will be issued.

If your test includes a simulator definition, the Time-out (msec) value you provide here must be greater than the Delay (msec) value defined on the simulator definition.

**For more information** about the Delay (msec) field, see How to Create a Simulator Definition.

## Test Messages

Use the **Test Messages** group box to generate XPath values based on provided response XML message text. By default, SOAP envelope XML text is provided in these fields. You can use the **Get Messages** button to generate request and response stub messages based on selected endpoint URL and operation values. Alternatively, you can paste XML text within this default SOAP envelope, or paste your own XML text already enclosed in an envelope into these fields.

**For more information** about obtaining request and response XML message text, see How to Obtain Message XML Text from a BPEL Process.
**For more information** about how to create test request and response messages that hold multiple sets of test data in a single definition, see How to Provide Multiple Request and Response Message Sets in a Single Test Definition.

**Request Message**

If request message XML text was entered on the Create Test page,

it is accessible and editable on this page.

Entering request message XML text for a test definition is required, whether the **Service Type** field value is set to ***Synchronous, Notify,*** or ***Asynchronous two way***.

You can use the **Get Messages** button to generate a request stub message based on selected endpoint URL and operation values.

If you are manually entering your request message, the “Paste your SOAP Message Content here” text in the stub message indicates where you should paste your actual request message text. This request message should mimic the XML message text sent by the service that normally initiates the service.

**Request CorrelationId Message**

This field only displays for a test definition with the **Service Type** field value set to ***Asynchronous two way.*** For this service type, entering a correlation ID value ensures that when the asynchronous response is actually received, the Composite Application Validation System (CAVS) is able to correlate it to the correct request.

If your request message is an Enterprise Business Message (EBM), leave this field blank, as the EBM header ID is automatically used as the correlation ID. In this case, because the EBM header ID is used as the correlation ID, do not use it as a key column in the simulator definition, if applicable.

If your request message is not an EBM, you must enter a correlation ID value. This correlation must be based on a unique key of the message. For example, CreateOrder can use Order ID as the correlation ID.

Click **Lookup** to access the Choose Request Correlation Id page, where you can select a correlation ID from XPath variables available in the message.

**Expected Response Message**

The ability to enter response message XML text is available when the **Service Type** field value is set to ***Synchronous*** or ***Asynchronous two way.***

If expected response message XML test was entered on the Create Test page, it is accessible and editable on this page.

You can manually enter the response message text on this page, or for test definitions with the **Service Type** field set to ***Synchronous***, you can use the **Get Messages** button to generate a response stub message based on selected endpoint URL and operation values.

Entering the expected response message XML facilitates the generation of XPath values, which are used to validate the actual response message returned in the test. You may also choose to manually enter or modify the XPath values directly on the Modify Test Definition page. If you are manually entering XPath values, you do not need to enter response message XML text.

When you enter response message XML text on this page, you can click **Generate Xpath** on the Modify Test Definition page to generate the XPath values that will be used to validate the expected response message you entered on this page against the actual response returned by the test.

If the **Service Type** field value is set to ***Synchronous*** or ***Asynchronous two way,*** you may choose to not enter response message XML text in this field. You do not need to enter response message XML if you are manually entering XPath values directly on the Modify Test Definition or if the test you are running does not require validation of the response message. For example, your test may be focused on just populating data.

The **Expected Response Message** text box is unavailable when the **Service Type** field value is set to ***Notify.*** In this case, a response message is not a test requirement.

**Generate Xpath**

Click to generate namespace and XPath values based on available **Endpoint URL** and **Response Message** values.

**Note.** Once you have generated XPath values, consider deleting any rows that will not be used in the testing effort.

The **Generate Xpath** button is unavailable when the **Service Type** field value is set to ***Notify.*** In this case, a response message is not a test requirement.

**Response Message Correlation ID**

This field only displays for a test definition with the **Service Type** field value set to ***Asynchronous two way.*** For this service type, entering a correlation ID value ensures that when the asynchronous response is actually received, the CAVS is able to correlate it to the correct request.

If your response message is an EBM, leave this field blank, as the EBM header ID is automatically used as the correlation ID. In this case, because the EBM header ID is used as the correlation ID, do not use it as a key column in the simulator definition, if applicable. If your response message is not an EBM, you must enter a correlation ID value. This correlation must be based on a unique key of the message. For example, CreateOrder can use Order ID as the correlation ID.

Click **Lookup** to access the Choose Response Correlation Id page, where you can select a correlation ID from XPath variables available in the message.

### Prefix and Namespace Selection

Use the **Prefix and Namespace Selection** grid to define namespace data that will be used in the XPath values defined in the **XPath Selection** grid.

**Delete**

Select one or more namespace rows and click **Delete** to execute the deletion. This button only appears when namespace rows are present.

**Create**

Click to manually add and populate a namespace row.

**Prefix**

Prefix that should be used for the namespace.

**Namespace**

Namespace to be used in the XPath data for the test definition.

## XPath Selection

Use the **XPath Selection** grid to work with XPath values that are used to compare the actual response message returned in the test to the expected response message defined in the **Response Message** text box on this page. The values in this grid use the namespace values set in the **Prefix and Namespace Selection** grid.

A common adjustment you will likely need to make to XPath conditions and expected node values in this grid is to genericize certain specific values, such as EBM IDs. For example, an EBM ID is unique for each transaction, so your test definition will likely not want to specify a particular EBM ID as response criteria. Instead, you may want to genericize the criteria to just verify that the EBM ID is a number greater than zero or use the ***Is Valid*** condition value.

**Note.** If you are entering XPath values manually, it is important to maintain correlations with the values entered in the **Prefix and Namespace Selection** grid. Each XPath node must have a prefix (namespace alias) that has been defined in the **Prefix and Namespace Selection** grid, unless it is an XPath expression.

The **XPath Selection** grid is unavailable when the **Service Type** field value is set to ***Notify.*** In this case, a response message is not a test requirement.

**Xpath**

When working with a test definition that contains multiple request and response data sets, use the **Xpath** drop-down list box to select the data set you want to use to run the test.

**For more information** about providing multiple data sets in a test definition, see How to Provide Multiple Request and Response Message Sets in a Single Test Definition.

**Delete**

Select one or more XPath rows and click **Delete** to execute the deletion. This button only appears when XPath rows are present.

**Create**

Click to manually add and populate an XPath row.

**XPath Sequence Id**

Indicates the sequence of the XPath expressions. This value is required. This value is read-only when it has been generated using **Generate Xpath**.

**XPath**

XPath data to be used in the test definition. These values can include XPath nodes and expressions. This value is read-only when it has been generated using the **Generate Xpath** button.

**Condition**

***Is Valid:*** The value provided in the XPath field is valid and no Expected Node Value is supplied.

***Equals To:*** The value provided in the XPath field is valid and an Expected Node Value is supplied.

***Not Equal To***

***Less Than***

***Greater Than***

***Less Than Equal***

***Greater Than Equal***

| | |
|---|---|
| | ***Not Null*** |
| **Expected Node Value** | The value expected in the response XML message. When you use the **Generate Xpath** button to generate XPath data, this value may be populated, but can be modified as necessary. The **Condition** field value is used to qualify this value. |

### Test Instance Selection

Select the **Test Instances** tab to display the **Test Instance Selection** grid, which displays information about test instances generated using the test definition.

| | |
|---|---|
| **Id** | Click to access the test instance on the Test Instance Detail page. |

**For more information** about the Test Instance Detail page, see How to View Test Instance Details.

### Linked Simulator Definition Selection

Select the **Simulator Definitions** tab to display the **Linked Simulator Definition Selection** grid, which displays information about simulator definitions that are linked to the selected test definition.

| | |
|---|---|
| **Unassign** | Select one or more simulator definition rows that you want to disassociate with the test definition. Click **Unassign** to execute the disassociation. |
| **Assign** | Click to access the Search Definitions - Simulator page, where you can search for a simulator definition that you want to assign to the test definition. Making this assignment facilitates reporting. Once the test definition runs and generates a test instance, all simulator instances generated by the simulator definition associated with the test definition will automatically be linked to the test instance.<br><br>Once you have assigned a simulator definition using the Search Definitions - Simulator page, the Modify Test Definition page appears, and displays the selected simulator definition. |
| **Refresh** | Click to refresh the Modify Test Definition page. |
| **Simulator Definition Id** | Click for an unlocked simulator definition to access the Modify Simulator Definition page.<br><br>Click for a locked simulator definition to access the View Simulator Definition page, where you can access a read-only view of the simulator definition. |

### Group Definition Selection

Select the **Group Definitions** tab to display the **Group Definition Selection** grid, which displays information about group definitions that include the test definition.

| | |
|---|---|
| **Group Definition Id** | Click to access the group definition on the Group Definition Detail page. |

**For more information** about the Group Definition Detail page, see

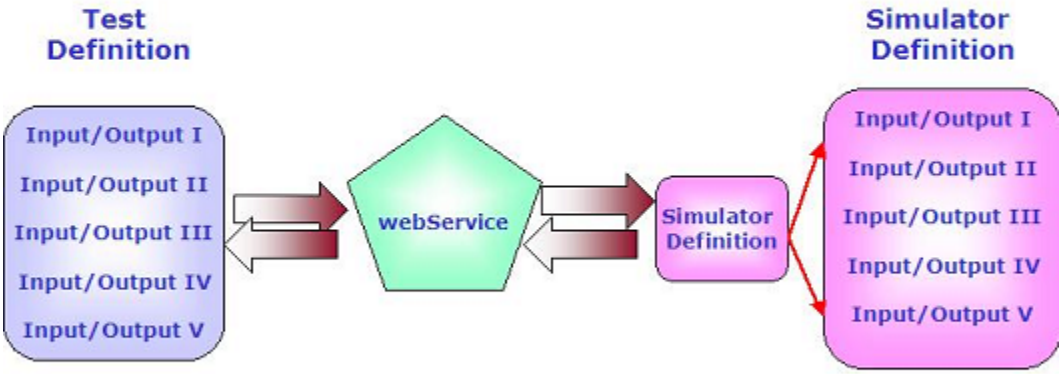
Working with Group Definitions.

| | |
|---|---|
| **Group Name** | Displays the descriptive name assigned to the group definition. |
| **Sequence Id within Group** | Displays the sequence in which the test definition is initiated by the group definition. |
| **Cancel** | Click to discard any updates you have made and return to the Definitions page. |
| **Actions** | Select the action you want to take with the test definition.<br><br>Execute: Select to execute the test definition. The status of the test execution appears at the top of the page. When a test definition has successfully executed, you can view details of the test instance on the Test Instance Details page.<br><br>**For more information** about the Test Instance Details page, see Working with Test and Simulator Instances.<br><br>Lock: Select to lock the test definition and view the test definition on the View Test Definition page. A locked definition cannot be edited.<br><br>Duplicate: Select to duplicate the test definition. The duplicate definition is created using the exact values of the original, with the exception of being given a unique Id value. |
| **Apply** | Click to apply and save any changes you have made to values on the page. |
| **Save** | Click to save entries on the page and go to the Definitions page.<br><br>**For more information** about the Definitions page, see Searching for Test and Simulator Definitions. |

## 4.3. How to Provide Multiple Request and Response Message Sets in a Single Test Definition

You can create a test definition that contains multiple pairs of request and response message data. This means that test definitions only need to be created per usage requirements, not per test data requirements.

For example, if you want to test a process against five sets of test data, you can create a single test definition to test the process and include in it all five sets of test data against which you want the process to operate. This is as opposed to creating five separate test definitions, one per combination of process and set of test data.




## Providing multiple request and response message sets in a single test definition

When multiple sets of test data are included in a test definition, each set will be executed in sequence. Separate test instances will be generated for each set of data. Test instances will reflect the success or failure of each segment of the test run using each set of test data.

### Request Message Format

Use the following format to include multiple sets of request data in the test definition.

The `CAVSRequestInputs` and `CAVSRequestInput_1` envelope are autogenerated. Use copy and paste commands to create more sets; `CAVSRequestInput_2` and `CAVSRequestInput_3`, for example.

```
<cavs:CAVSRequestInputs
xmlns:cavs="http://schemas.xmlsoap.org/cavs/requestenvelope/">
<cavs:CAVSRequestInput_1>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Body xmlns:ns1="http://xmlns.oracle.com/SimpleProcess">
   <ns1:SimpleProcessProcessRequest>
…
   </ns1:SimpleProcessProcessRequest>
   </soap:Body>
</soap:Envelope>
</cavs:CAVSRequestInput_1>

<cavs:CAVSRequestInput_2>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Body xmlns:ns1="http://xmlns.oracle.com/SimpleProcess">
   <ns1:SimpleProcessProcessRequest>
   …
   </ns1:SimpleProcessProcessRequest>
```

```
   </soap:Body>
</soap:Envelope>
</cavs:CAVSRequestInput_2>

</cavs:CAVSRequestInputs>
```

### Response Message Format

Use the following format to include multiple sets of response data in the test definition.

```
<cavs:CAVSResponseOutput_1>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Body xmlns:ns1="http://xmlns.oracle.com/SimpleProcess">
   <ns1:SimpleProcessProcessResponse>
       …
   </ns1:SimpleProcessProcessResponse>
   </soap:Body>
</soap:Envelope>
</cavs:CAVSResponseOutput_1>

<cavs:CAVSResponseOutput_2>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Body xmlns:ns1="http://xmlns.oracle.com/SimpleProcess">
   <ns1:SimpleProcessProcessResponse>
       …
   </ns1:SimpleProcessProcessResponse>
   </soap:Body>
</soap:Envelope>
</cavs:CAVSResponseOutput_2>
</cavs:CAVSResponseOutputs>
```

After entering request and response data sets and clicking the Generate Xpath button on the Modify Test Definition page, the XPath Selection grid provides access to the Please select an Xpath drop-down list box, where you can select the set of test data you want to use to run the test.

**For more information** about the Modify Test Definition page, see How to Modify a Test Definition.

If your testing scenario includes simulator definitions, you can likewise create simulator definitions that contain multiple request and response message sets that work with the sets defined in your test definition.

**For more information**, see How to Provide Multiple Request and Response Message Sets in a Single Simulator Definition.

# 5. Creating and Modifying Simulator Definitions

A simulator definition is created by the Composite Application Validation System (CAVS) user to simulate a particular service in an Oracle Application Integration Architecture (AIA) integration or a participating application. A simulator receives data from the tested web service and returns predefined data so that the tested web service can continue processing.

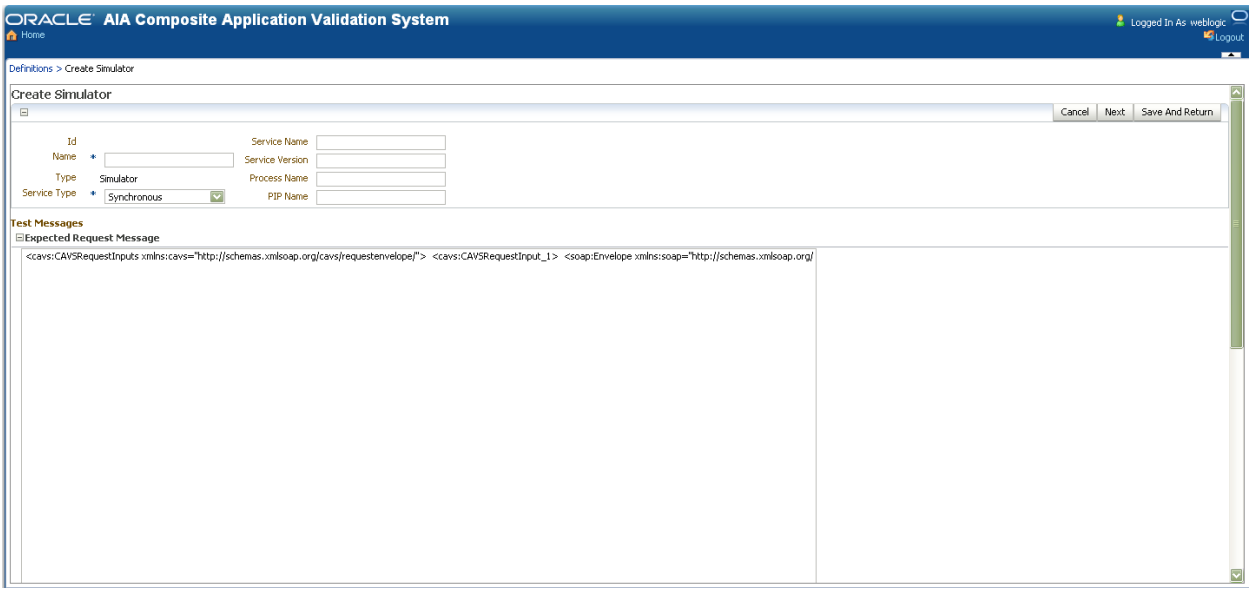
This chapter discusses:

- How to Create a Simulator Definition
- How to Modify a Simulator Definition
- How to Provide Multiple Request and Response Message Sets in a Single Simulator Definition
- How to Create a Simulator Definition that Supports Chatty Services
- How to Send Dynamic Responses in a Simulator Response

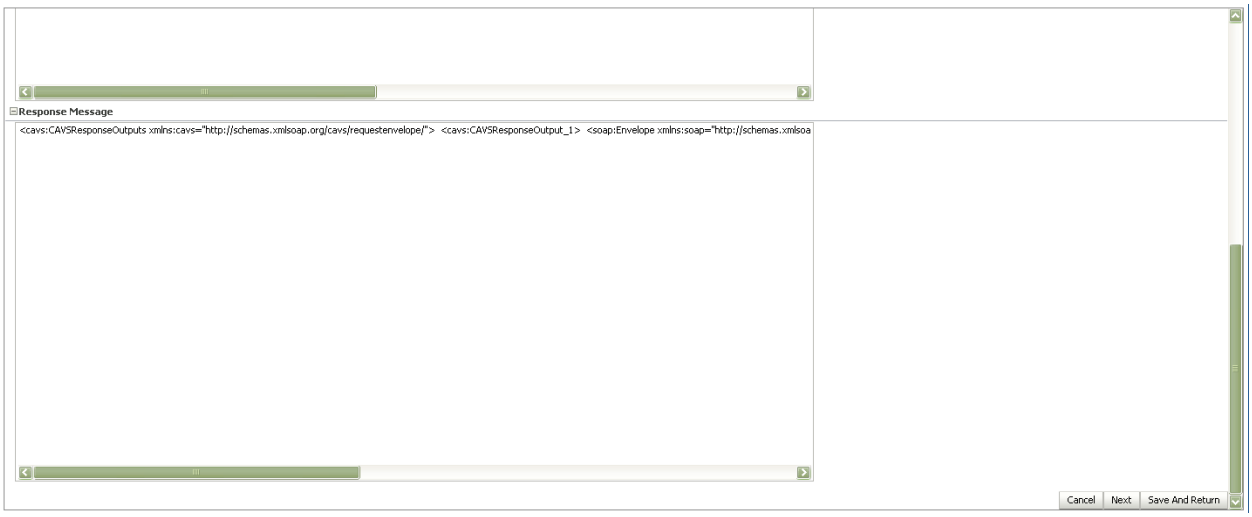
## 5.1. How to Create a Simulator Definition

To create a simulator definition:

1. Access the Create Simulator page. To access the page, access the AIA Home Page. In the Composite Application Validation System area, click the Go button. Select the Definitions tab. Click the Create Simulator button.




Create Simulator page (1 of 2)




Create Simulator page (2 of 2)

2. Use the following page elements on the Create Simulator page to create simulator definitions.

| | |
|---|---|
| **Id** | Upon saving the simulator definition, a unique key identifier is assigned to the simulator definition. |
| **Name** | Enter the descriptive name you want to use for the simulator definition. |
| **Type** | Displays the type of definition you have chosen to create. On the Create Simulator page, this value will always be set to ***Simulator.*** |
| **Service Type** | Select the business service pattern of the web service the simulator definition is simulating.<br>***Synchronous*** (request-and-reply)<br>***Notify*** (asynchronous request-only)<br>***Asynchronous two way*** |
| **Service Name** | Enter the name of the web service that you want to simulate using the simulator definition. |
| **Service Version** | Enter the version of the web service you want simulate using the simulator definition. |
| **Process Name** | Enter the name of the process that includes the web service that you want to simulate using the simulator definition. |
| **PIP Name (Process Integration Pack name)** | Enter the name of the PIP that includes the web service that you want to simulate using the simulator definition. |

### Test Messages

Use the **Test Messages** group box to generate XPath values based on provided request XML message text. By default, SOAP envelope XML text is provided in these fields. You can paste XML text within this default SOAP envelope, or paste your own XML text already enclosed in an envelope into these fields.

**For more information** about how to create simulator request and response messages that hold multiple sets of test data in a single definition, see How to Provide Multiple Request and Response Message Sets in a Single Simulator Definition.
**For more information** about how to create simulator request and response messages that support chatty service conversations, see How to Create a Simulator Definition that Supports Chatty Services.

**Expected Request Message**

Entering request message XML text facilitates the generation of XPath values that are used to match a received request with this simulator's expected request, as well as to validate values in this received request message. That is, the XPath values you supply provide a signature for the simulator definition that the simulator service attempts to match with arriving request actions. In addition to enabling the simulator service to match a test request with a simulator definition, the XPath criteria you provide can also serve to validate data sent in the test request.

If a simulator has been directly designated for use in the AIAConfigurationProperties.xml via the Routing Configurations page, the simulator definition will be identified directly. However, once the simulator has been identified, there may be multiple requests within it. If so, the XPath key field values provide an efficient search method for request matching.

**For more information** about the Routing Configurations page, see Defining CAVS Routing Setup IDs.

You can enter expected request message XML text on this page and click **Generate Xpath** on the Modify Simulator Definition page to generate XPath values used to validate the actual request sent by the test definition. You may also choose to manually enter or modify the XPath values directly on the Modify Simulator Definition page. You do not need to enter request message XML if you are manually entering XPath values directly on the Modify Simulator Definition page.

You may choose to copy and paste messages from the BPEL Console, instead of manually entering them.

**For more information,** see How to Obtain Message XML Text from a BPEL Process.

**Response Message**

Entering response message XML text for a simulator definition is required when the **Service Type** field value is set to ***Synchronous*** or ***Asynchronous two way.*** Enter the XML text of the response message that you want to use for the simulator definition. This response message should mimic the actual response message that would be sent by the service that the simulator definition is simulating.

This text box is hidden when the **Service Type field** value is set to ***Notify.*** In this case, a response message is not a simulator requirement.

You may choose to copy and paste messages from the BPEL

Console, instead of manually entering them.

**For more information,** see How to Obtain Message XML Text from a BPEL Process.

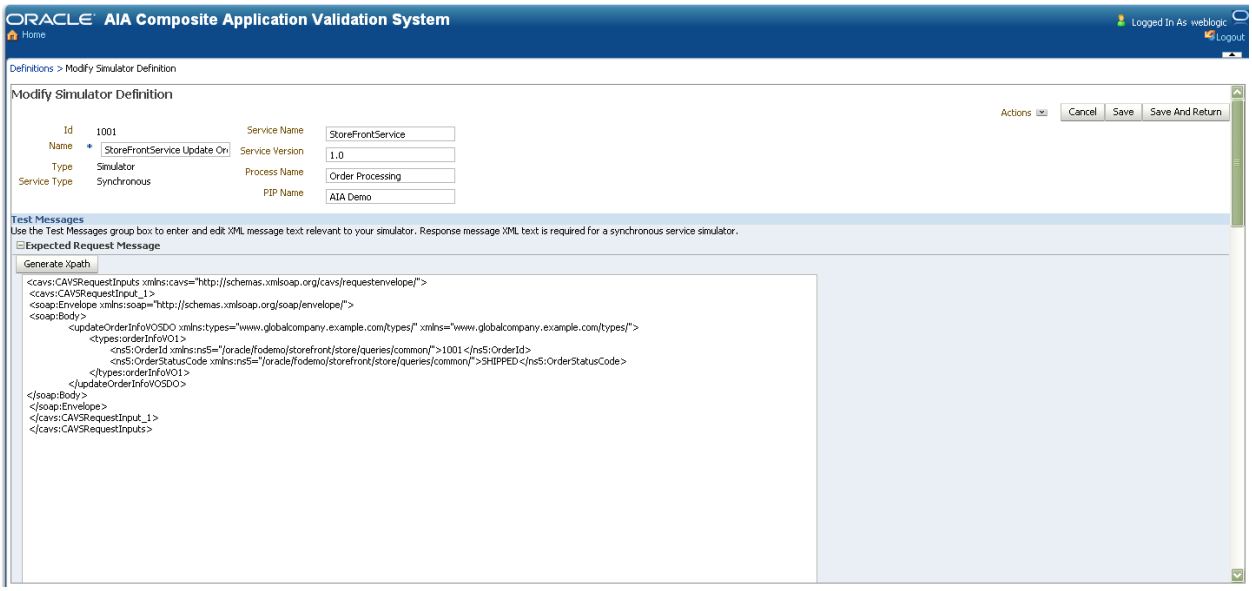
| | |
|---|---|
| **Cancel** | Click to discard any updates you have made and return to the Definitions page. |
| **Next** | Click to save entries on the Create Simulator page and go to the Modify Simulator Definition page, where you can generate XPaths and further edit and manage the simulator definition. |
| **Save** | Click to save entries on the Create Simulator page and return to the Definitions page. |

## 5.2. How to Modify a Simulator Definition

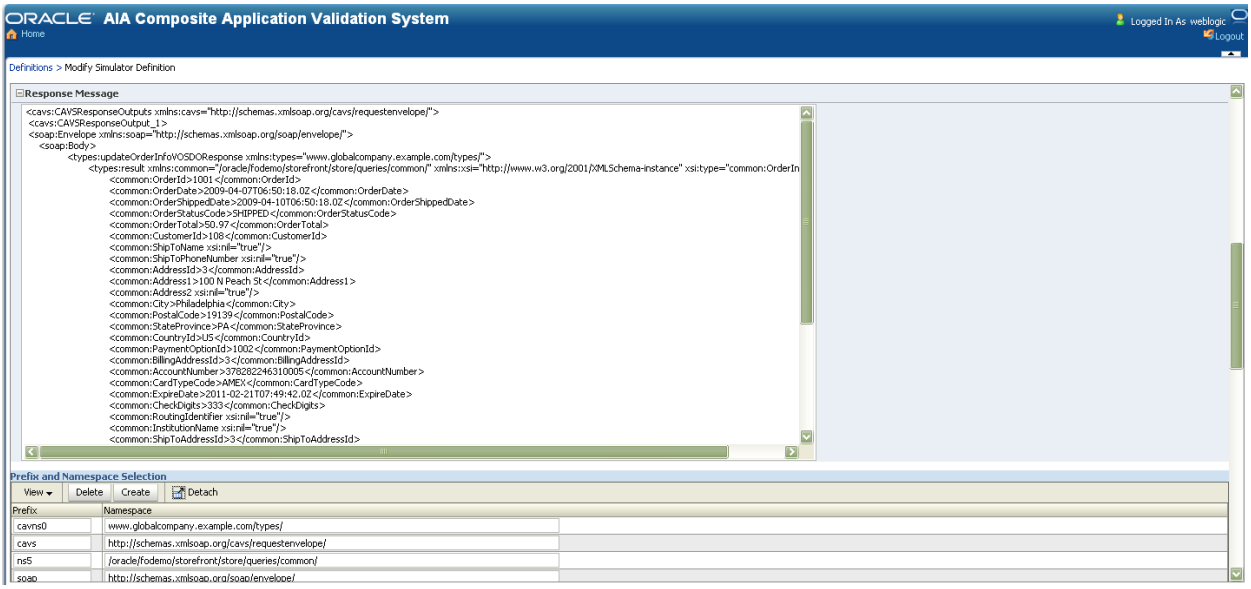
To modify a simulator definition:

1. Access the Modify Test Definition page. To access the page, use one of the following navigation paths:

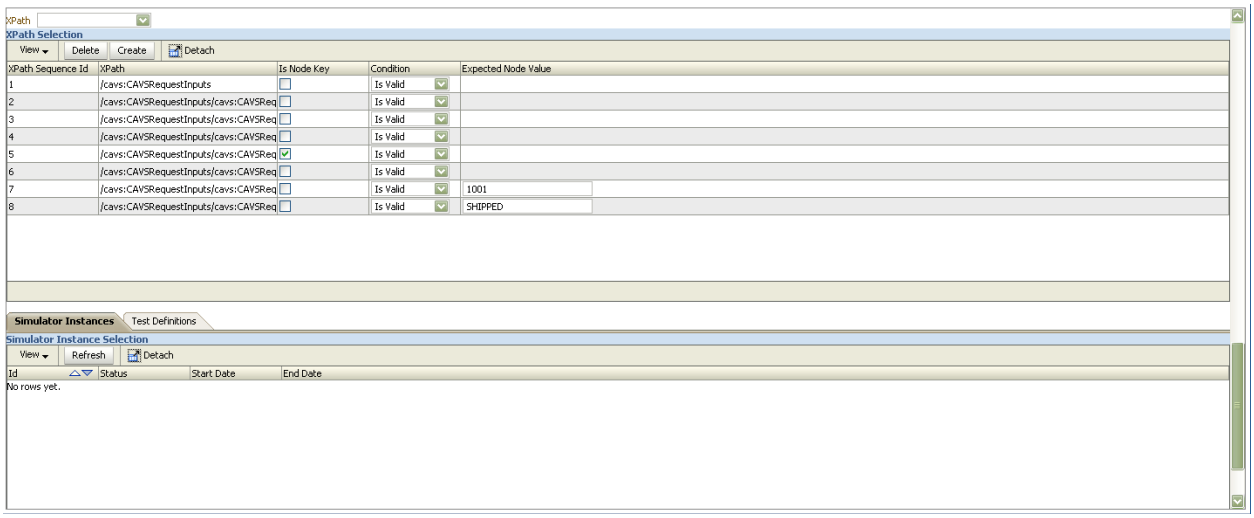
   - Enter required values on the Create Simulator page and click Next. To access the Create Simulator page, access the AIA Home Page. In the Composite Application Validation System area, click the Go button. Select the Definitions tab. Click the Create Simulator button.
   - Click an Id link for an unlocked simulator definition in the Search Result Selection grid on the Definitions page. To access the Definitions page, access the AIA Home Page. In the Composite Application Validation System area, click the Go button. Select the Definitions tab.
   - Click a Definition Id link for an unlocked simulator definition on the Instances page. To access the Instances page, access the AIA Home Page. In the Composite Application Validation System area, click the Go button. Select the Instances tab.




Modify Simulator Definition page (1 of 3)




Modify Simulator Definition page (2 of 3)




Modify Simulator Definition page (3 of 3)

2. Use the following page elements on the Modify Simulator Definition page to modify a simulator definition. The page displays values you defined for the simulator definition. You can modify the values in editable fields.

**Actions** — Select the action you want to take with the simulator definition.

***Lock:*** Select to lock the simulator definition and view the simulator definition on the View Simulator Definition page. A locked definition cannot be edited.

***Duplicate:*** Select to duplicate the simulator definition. The duplicate definition is created using the exact values of the original, with the exception of being given a unique Id value.

**Cancel** — Click to discard any updates you have made and return to the Definitions page.

**Apply** — Click to apply and save any changes you have made to values on

| | |
|---|---|
| | the page. |
| **Save** | Click to save entries on the page and go to the Definitions page. |

**For more information,** see Searching for Test and Simulator Definitions.

| | |
|---|---|
| **Callback URL** | If you are creating a simulator with a Service Type of Asynchronous two way, enter the URL of the web service that should be called back by the simulator. |
| **SOAP Action** | If you are creating a simulator with a Service Type of Asynchronous two way, enter the operation of the callback URL. |
| **Delay (msec)** | If you are creating a simulator with a Service Type of Asynchronous two way, enter the number of milliseconds that you want the simulator definition to wait before issuing the call back service invocation.<br><br>Note. If you are using this simulator along with an asynchronous two-way test definition, ensure that the Delay (msec) value you provide is less than the Timeout (msec) value defined for any test definition<br><br>For more information about the Timeout (msec) field, see How to Modify a Test Definition. |

### Test Messages

**For more information** about the elements in the **Test Messages** group box, see How to Create a Simulator Definition.

### Prefix and Namespace Selection

Use the **Prefix and Namespace Selection** grid to define namespace data that will be used in the XPath values defined in the XPath Selection grid.

| | |
|---|---|
| **Delete** | Select one or more namespace rows and click **Delete** to execute the deletion.<br><br>This button only appears when namespace rows are present. |
| **Create** | Click to manually add and populate a namespace row. |
| **Prefix** | Prefix that should be used for the namespace. |
| **Namespace** | Namespace to be used in the XPath data for the simulator definition. |

### XPath Selection

Use the **XPath Selection** grid to work with XPath values that are used to match the simulator definition with arriving requests. XPath values can also be used to validate data send in the test request. The values in this grid use the namespace values set in the **Prefix and Namespace Selection** grid.

**Note.** If you are entering XPath values manually, it is important to maintain correlations with the values entered in the **Prefix and Namespace Selection** grid. Each XPath node must have a prefix that has been defined in the **Prefix and Namespace Selection** grid, unless it is an XPath expression.

| | |
|---|---|
| **Delete** | Select one or more XPath rows and click **Delete** to execute the deletion.<br><br>This button only appears when XPath rows are present. |
| **Create** | Click to add and manually populate an XPath row. |
| **XPath Sequence Id** | Indicates the sequence of the XPath expressions. This value is required. This value is read-only when it has been generated using **Generate Xpath**. |
| **Xpath** | XPath value used to help match the simulator definition with arriving requests. These values can include XPath nodes and expressions. This value is read-only when it has been generated using the **Generate Xpath** button. |
| **Is Node Key** | Select if the XPath node is a key value to be used in matching arriving test requests with the simulator. |
| **Condition** | ***Is Valid:*** The value provided in the XPath field is valid and no **Expected Node Value** is supplied.<br><br>***Equals To:*** The value provided in the XPath field is valid and an **Expected Node Value** is supplied.<br><br>***Not Equal To***<br><br>***Less Than***<br><br>***Greater Than***<br><br>***Less Than Equal***<br><br>***Greater Than Equal***<br><br>***Not Null*** |
| **Expected Node Value** | The value that the simulator expects to receive from the service that invokes it. When the simulator is actually executed, this value is compared with the actual value based on the validation condition selected in the **Condition** field<br><br>When you use the **Generate Xpath** button to generate XPath data, this value may be populated, but can be modified as necessary. The **Condition** field value is used to qualify this value. |

## Simulator Instance Selection

Select the **Simulator Instances** tab to display the **Simulator Instance Selection** grid, which displays information about simulator instances generated using the simulator definition.

| | |
|---|---|
| **Refresh** | Click to refresh the Modify Simulator Definition page. |
| **Id** | Click to display the selected instance ID on the Simulator Instances Detail page. |

**For more information** about the Simulator Instances Detail

page, see How to View Simulator Instance Details.

| | |
|---|---|
| **Status** | Displays the status of the simulator instance generated by the simulator definition.<br>***Initiated:*** The simulator instance has been initiated.<br>***Ended:*** This status is only applicable to simulator instances that do not involve validations. Indicates that the instance has ended.<br>***Faulted:*** The simulator instance could not execute properly due to exceptions or faults.<br>***Failed:*** The simulator instance did not pass validation.<br>***Passed:*** The simulator instance passed validation. |
| **Start Date** | Displays the date and time at which the simulator instance was initiated. |
| **End Date** | Displays the date and time at which the simulator instance ended. |

### Test Definition Selection

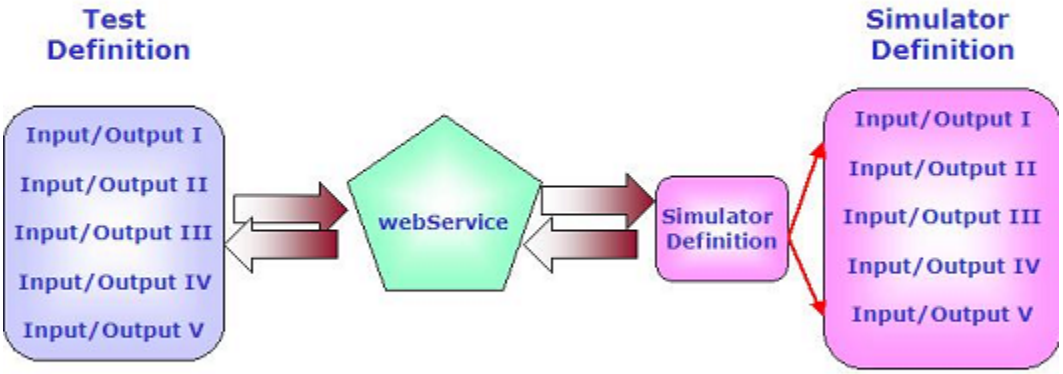
Select the **Test Definitions** tab to display the **Linked Test Definition Selection** grid, which displays information about test definitions associated with the simulator definition.

| | |
|---|---|
| **Delete** | Select one or more test definition rows that you want to delete and click **Delete** to execute the deletion. |
| **Assign** | Click to access the Search Definitions - Test page, where you can search for a test definition to which you want to assign the simulator definition. |
| **Refresh** | Click to refresh the Modify Simulator Definition page. |

## 5.3. How to Provide Multiple Request and Response Message Sets in a Single Simulator Definition

You can create a simulator definition that contains multiple pairs of request and response message data. This means that simulator definitions only need to be created per usage requirements, not per test data requirements.

For example, if you want to simulate a service against five sets of test data, you can create a single simulator definition to simulate the service and include in it all five sets of test data with which you can the service to operate. This is as opposed to creating five separate simulator definitions, one per combination of service and set of test data.




## Providing multiple request and response message sets in a single simulator definition

When a simulator definition that includes multiple test data sets is invoked, the appropriate data set is matched for use based on key attributes identified in the request. At this point, the request validation and response provision can occur. Since we would typically use such definitions to handle several sets of data, it is recommended that you choose the same key values for every set of data.

### Request Message Format

Use the following format to include multiple sets of request data in the simulator definition.

The `CAVSRequestInputs` and `CAVSRequestInput_1` envelope are autogenerated upon the input of the endpoint URL value on the test definition. Use copy and paste commands to create more sets; `CAVSRequestInput_2` and `CAVSRequestInput_3`, for example.

```
<cavs:CAVSRequestInputs
xmlns:cavs="http://schemas.xmlsoap.org/cavs/requestenvelope/">
<cavs:CAVSRequestInput_1>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Body xmlns:ns1="http://xmlns.oracle.com/SimpleProcess">
   <ns1:SimpleProcessProcessRequest>
…
   </ns1:SimpleProcessProcessRequest>
   </soap:Body>
</soap:Envelope>
</cavs:CAVSRequestInput_1>

<cavs:CAVSRequestInput_2>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Body xmlns:ns1="http://xmlns.oracle.com/SimpleProcess">
   <ns1:SimpleProcessProcessRequest>
   …
```

```
    </ns1:SimpleProcessProcessRequest>
    </soap:Body>
</soap:Envelope>
</cavs:CAVSRequestInput_2>

</cavs:CAVSRequestInputs>
```

### Response Message Format

Use the following format to include multiple sets of response data in the simulator definition.

```
<cavs:CAVSResponseOutput_1>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Body xmlns:ns1="http://xmlns.oracle.com/SimpleProcess">
    <ns1:SimpleProcessProcessResponse>
        …
    </ns1:SimpleProcessProcessResponse>
    </soap:Body>
</soap:Envelope>
</cavs:CAVSResponseOutput_1>

<cavs:CAVSResponseOutput_2>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Body xmlns:ns1="http://xmlns.oracle.com/SimpleProcess">
    <ns1:SimpleProcessProcessResponse>
        …
    </ns1:SimpleProcessProcessResponse>
    </soap:Body>
</soap:Envelope>
</cavs:CAVSResponseOutput_2>
</cavs:CAVSResponseOutputs>
```

Envelope text is prepopulated. Enter actual message content within appropriate tags provided within the envelopes.

After entering request and response data sets and clicking the Generate Xpath button on the Modify Simulator Definition page, the XPath Selection grid provides access to available XPath values and enables you to select the XPaths that must be treated as key nodes.

**For more information** about the Modify Simulator Definition page, see How to Modify a Simulator Definition.

If your testing scenario includes test definitions, you can likewise create test definitions that contain multiple request and response message sets that work with the sets defined in your simulator definition.

**For more information**, see How to Provide Multiple Request and Response Message Sets in a Single Test Definition.

## 5.4. How to Create a Simulator Definition that Supports Chatty Services

You can create a simulator definition that can simulate multiple services, each with a different schema.

In general, we recommend that you create simulators that simulate a single specific service. However, in the case of chatty conversations, for the ease of maintenance, you may choose to simulate all callouts of an Application Business Connector Service (ABCS) using a single simulator definition.

Using this method, you have the advantage of using one simulator for a particular ABCS, regardless of the number of callouts that need to be made. This method also provides ease of maintenance because linked callouts can all be viewed and modified in one place.

For example, in some integration scenarios, participating applications do not provide services at the same level of granularity as operations in Enterprise Business Services (EBSs). In these scenarios, a requester ABCS may need to adopt patterns such as message enrichment, splitting, and aggregation and disaggregation as required by an EBS. Likewise, a provider ABCS may need to adopt patterns as required by participating application services.

These ABCSs, which are typically implemented using BPEL process, call out to several services. To test this “chatty” ABCS using CAVS, there will likely be a need to replace the services that the ABCS calls out to with several simulators. It will also be required that these multiple request/response simulators be correlated, so that they accurately emulate the transaction of the same entity.

When this type of simulator is called, CAVS initiates the following general flow:

1. Selects simulator definition.
2. Validates the first request message based on the selected simulator.
3. Returns the appropriate response message, if the selected simulator is a two-way simulator.
4. Repeats steps 2 and 3 until the chatty service conversation is complete.

### Request Message Format

Use the following format to create a simulator definition that supports chatty service conversations. This format provides the ability to specify a set of request and response messages, along with success criteria for each of them. This format is the same as that used for multiple requests and responses in a simulator definition. However, in this case, the schemas for each set will be different.

```
<cavs:CAVSRequestInputs
xmlns:cavs="http://schemas.xmlsoap.org/cavs/requestenvelope/">
<cavs:CAVSRequestInput_1>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Body xmlns:ns1="http://xmlns.oracle.com/Service1">
   <ns1:Service1Request>
…
   </ns1: Service1Request>
   </soap:Body>
</soap:Envelope>
</cavs:CAVSRequestInput_1>

<cavs:CAVSRequestInput_2>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Body xmlns:ns2="http://xmlns.oracle.com/ Service2">
   <ns2: Service2Request>
…
   </ns2: Service2Request>
   </soap:Body>
</soap:Envelope>
</cavs:CAVSRequestInput_2>
```

Once you have provided request and response messages, click Generate Xpath on the Modify Simulator Definition page to generate XPath values. Modify the generated XPath values, if necessary.

**For more information** about the Modify Simulator Definition page, see How to Modify a Simulator Definition.

When this type of simulator is called, separate simulator instances are created for each request and response pair. The evaluation of actual response versus expected response is handled per instance created for the same simulator definition.

## 5.5. How to Send Dynamic Responses in a Simulator Response

CAVS simulator definitions are actually predefined request and response message sets. In some cases, you may not know the values for all the fields in the request message. Additionally, you may want to send these unknown dynamic values in a response to the service that called the simulator.

For example, consider the Enterprise Business Message (EBM) ID. This value is normally generated on the fly by AIA services. If you create a simulator that talks to this AIA service, you do not have a way to validate the value in the EBM ID field of the request message because the value is dynamically generated.

You may to choose to avoid validations of this value by setting the CAVS XPath validation for the EBM ID field to ***isValid.*** However, you may have a requirement in which you need to send this dynamic value back in a particular field of the simulator response. To meet this requirement, you can let the simulator pick the particular field (such as EBM ID) in the request and send it back as a field in the response.

To send a dynamic response in a simulator response:

1. Map a field from the request message and add it to the response message.

   These are two valid formats you can use:

   ```
   #@#XPATH.{copy the XPath from request msg Ex./soap:Envelop/soap:Body..}#@#

   #@#SYSTEM.{SYSDATE}#@#
   ```

2. Before sending the response, the simulator will pick up this ID from the generated XPath, substitute the actual value, and send it in the response.

   The strings referenced above will form a part of the response message. To know what the request message XPath values are, use the output that was generated by clicking the Generate Xpath button.

   For example, let's say that the request SOAP message has the following nodes:

   ```
   <corecom:PersonName>
     <corecom:FirstName>CAVS</corecom:FirstName>
     <corecom:MiddleName>FP</corecom:MiddleName>
     <corecom:FamilyName>AIA</corecom:FamilyName>
     <corecom:CreationDateTime></corecom:CreationDateTime>
   </corecom:PersonName>
   ```

   You would define your response SOAP message as follows:

   ```
   <corecom:PersonName>
   ```

```
<corecom:FirstName>#@#XPATH.{/soap:Envelope/soap:Body/corecom:CreateCustom
erPa
rtyListEBM/ebo:DataArea/ebo:CreateCustomerPartyList/corecomx:Contact/corec
omx:
PersonName/corecomx:FamilyName}#@#2dot1</corecom:FirstName>

<corecom:MiddleName>#@#XPATH.{/soap:Envelope/soap:Body/corecom:CreateCusto
merP
artyListEBM/ebo:DataArea/ebo:CreateCustomerPartyList/corecomx:Contact/core
comx
:PersonName/corecomx:MiddleName}#@#</corecom:MiddleName>

<corecom:FamilyName>#@#XPATH.{/soap:Envelope/soap:Body/corecom:CreateCusto
merP
artyListEBM/ebo:DataArea/ebo:CreateCustomerPartyList/corecomx:Contact/core
comx
:PersonName/corecomx:FirstName}#@#</corecom:FamilyName>
<corecom:CreationDateTime>#@#SYSTEM.{SYSDATE}#@#</corecom:CreationDateTime
>
</corecom:PersonName>
```

In this case, the response would be modified by the CAVS engine by copying values from the request as follows:

```
<corecom:PersonName>
  <corecom:FirstName>AIA2dot1</corecom:FirstName>
  <corecom:MiddleName>FP</corecom:MiddleName>
  <corecom:FamilyName>CAVS</corecom:FamilyName>
<corecom:CreationDateTime>2008-05-
12T15:26:43+05:30</corecom:CreationDateTime>
</corecom:PersonName>
```

Note that `2dot1` is a static string that is always appended to the FamilyName value.

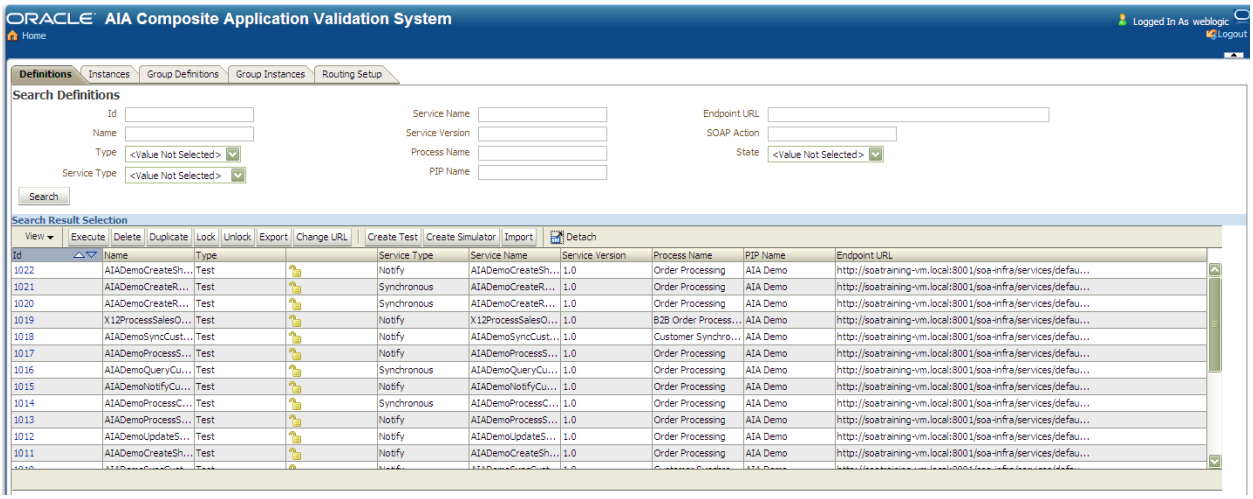
# 6. Searching for Test and Simulator Definitions

This chapter discusses How to Search for and Work with Test and Simulator Definitions.

## 6.1. How to Search for and Work with Test and Simulator Definitions

To search for and work with test and simulator definitions:

1. Access the Definitions page. To access the page, access the Oracle Application Integration Architecture (AIA) Home Page. In the Composite Application Validation System area, click the Go button. Select the Definitions tab.




Definitions page

2. Use the following page elements on the Definitions page to search for, execute, migrate, and manage existing test and simulator definitions. You can also access pages you can use to create and modify test and simulator definitions.

### Search Definitions

Use the **Search Definitions** group box to enter search criteria to find the test or simulator definition you are searching for.

| | |
|---|---|
| **Id** | Enter the unique key identifier assigned to the test or simulator definition. |
| **Name** | Enter the descriptive name assigned to the test or simulator definition. |
| **Type** | Select the type of definition.<br>***<Value Not Selected>:*** Select to display all definition types. |

| | |
|---|---|
| | ***Test*** |
| | ***Simulator*** |
| **Service Type** | Select the business service pattern of the web service for which the definition was created. |
| | ***<Value Not Selected>:*** Select to display definitions for all service types. |
| | ***Synchronous*** |
| | ***Notify*** |
| | ***Asynchronous two way*** |
| **Service Name** | Enter the name of the web service for which the definition was created. |
| **Service Version** | Enter the version of the service for which the definition was created. This is the web service whose URL is provided in the **Endpoint URL** field. |
| **Process Name** | Enter the name of the process that includes the web service for which the definition was created. |
| **PIP Name (Process Integration Pack name)** | Enter the name of the Process Integration Pack that includes the web service for which the definition was created. |
| **Endpoint URL** | Enter the URL of the web service for which the definition was created. |
| **SOAP Action** | Enter the operation called by the web service for which the definition was created. |
| **State** | Select the state of the definition. |
| | ***<Value Not Selected>:*** Select to display definitions in all states. |
| | ***Locked*** |
| | ***Unlocked*** |
| **Search** | Click to execute a search for definitions using the search criteria entered in the **Search Definitions** group box. |

## Search Result Selection

Use the **Search Result Selection** grid to work with definitions returned in your search results. Upon accessing this page, the grid displays all definitions.

**Execute**

Select one or more test definitions that you want to run and click **Execute** to execute the test definition. When a test definition has successfully executed, you can view details of the test instance generated by the test execution on the Test Instance Details page.

**For more information** about the Test Instance Details page, see How to View Test Instance Details.

Simulator definitions cannot be executed.

**Delete**

Select one or more definitions that you want to delete and click **Delete** to execute the deletion.

**Duplicate**

Select one or more definitions that you want to duplicate and click **Duplicate** to execute the duplication.

The duplicate definition is created using the exact values of the original, with the exception of being assigned a unique ID value.

**Lock**

Select one or more definitions that you want to lock and click **Lock** to lock the definitions. A definition with its **State** value set to ***Locked*** cannot be edited.

**Unlock**

Select one or more definitions that you want to unlock and click **Unlock** to unlock the definitions. An unlocked definition can be edited. A definition with its **State** value set to ***Unlocked*** is editable.

**Export**

**For more information** about exporting definitions and instances, see Exporting and Importing CAVS Definitions and Instances.

**Change URL**

Select one or more test definitions for which you want to change the endpoint URL value. Click **Change URL** to launch a pop-up window in which you can enter the new endpoint URL value that you want to use for the selected test definitions.

**Create Test**

Click to access the Create Test page, where you can create a test definition.

**For more information** about the Create Test page, see How to Create a Test Definition.

**Create Simulator**

Click to access the Create Simulator page, where you can create a simulator definition.

**For more information** about the Create Simulator page, see How to Create a Simulator Definition.

**Import**

**For more information** about importing test definitions, see Exporting and Importing CAVS Definitions and Instances.

**Id**

Click for an unlocked test definition to access the Modify Test Definition page.

Click for a locked test definition to access the View Test Definition page, where you can access a read-only view of the test definition.

Click for an unlocked simulator definition to access the Modify Simulator Definition page.

Click for a locked simulator definition to access the View Simulator Definition page, where you can access a read-only view of the simulator definition.

**For more information,** see How to Modify a Test Definition.

**For more information,** see How to Modify a Simulator Definition.

# 7. Working with Group Definitions

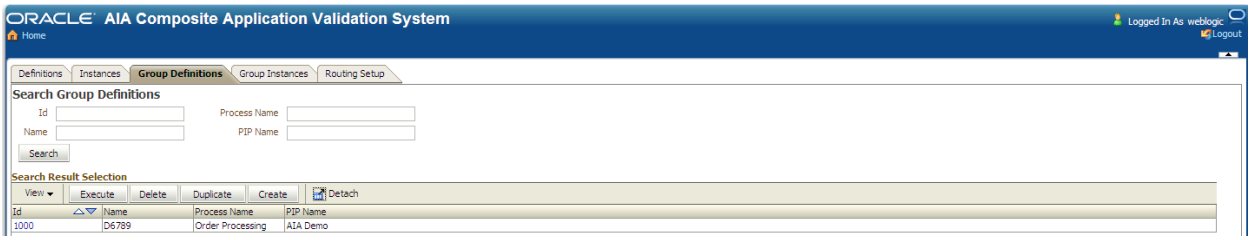
This chapter discusses:

- How to Work with Group Definitions
- How to Create and Modify a Group Definition

## 7.1. How to Work with Group Definitions

To work with group definitions:

1. Access the Group Definitions page. To access the page, access the Oracle Application Integration Architecture (AIA) Home Page. In the Composite Application Validation System area, click the Go button. Select the Group Definitions tab.




Group Definitions page

2. Use the following page elements on the Group Definitions page to search for, execute, and manage existing group definitions. You can also access a page you can use to create and modify group definitions.

### Search Group Definitions

Use the **Search Group Definitions** group box to enter search criteria to find the group definition you are searching for.

| | |
|---|---|
| **Id** | Enter the unique key identifier assigned to the group definition. |
| **Name** | Enter the descriptive name assigned to the group definition. |
| **Process Name** | Enter the name of the process associated with the group definition. |
| **PIP (Process Integration Pack) Name** | Enter the name of the Process Integration Pack (PIP) associated with the group definition. |
| **Search** | Click to execute a search for group definitions using the search criteria entered in the Search Group Definitions group box. |

### Search Result Selection

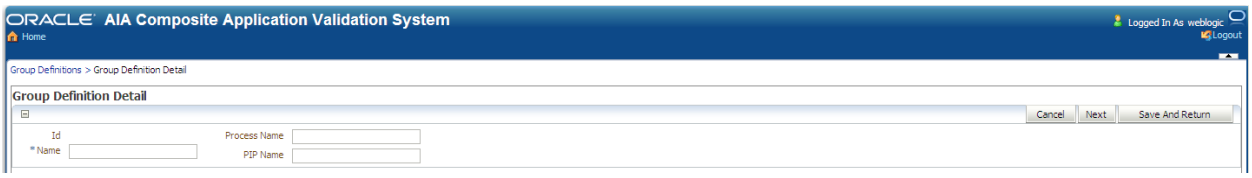
Use the **Search Result Selection** grid to work with group definitions returned in your search results. Upon accessing this page, the grid is populated by all group definitions.

| | |
|---|---|
| **Execute** | Select one or more group definitions that you want to run and click **Execute** to execute the group definition.<br><br>When a group definition has successfully executed, you can view details of the group instance on the Group Instances Detail page.<br><br>**For more information** about the Group Instances Detail page, see How to View Group Instance Details. |
| **Delete** | Select one or more group definitions that you want to delete and click **Delete** to execute the deletion. |
| **Duplicate** | Select one or more group definitions that you want to duplicate and click **Duplicate** to execute the duplication.<br><br>The duplicate group definition is created using the exact values of the original, with the exception of being given a unique Id value. |
| **Create** | Click to access the Group Definition Detail page, where you can create a group definition |
| **Id** | Click to access the Group Definition Detail page. |

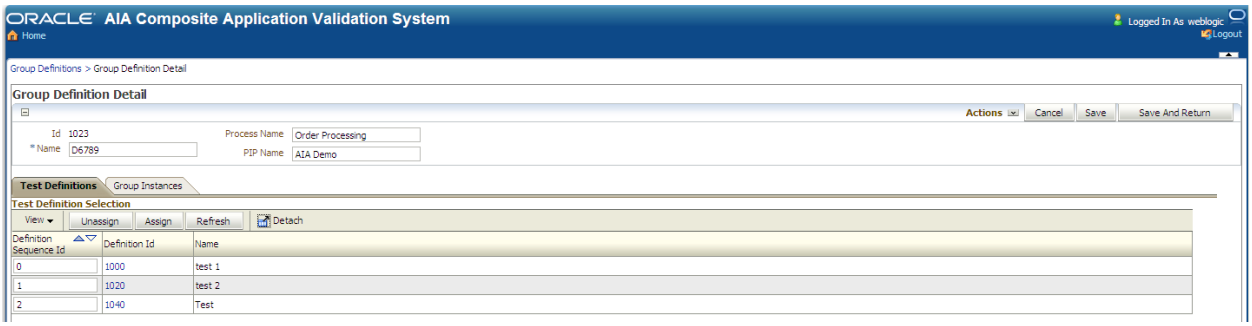
## 7.2. How to Create and Modify a Group Definition

To create and modify a group definition:

1. Access the Group Definition Detail page. To access the page, click the Create button on the Group Definitions page.




Group Definition Detail page (new group definition)




Group Definition Detail page (existing definition)

2. Use the following page elements on the Group Definition Detail page to create and modify a group definition that combines one or more tests and executes them in a single-threaded batch sequence.

| | |
|---|---|
| **Actions** | Select the action you want to take with the group definition.<br><br>***Execute:*** Select to execute the group definition.<br><br>When a group definition has successfully executed, you can view details of the test instance on the Group Instances page.<br><br>***Duplicate:*** Select to duplicate the group definition. The duplicate definition is created using the exact values of the original, with the exception of being given a unique Group Definition Id value. |
| **Cancel** | Click to discard updates to the page and return to the Group Definitions page. |
| **Next** | For a new group definition, click to save entries and display further group definition details on the Group Definition Detail page.<br><br>This button does not appear for existing group definitions. |
| **Apply** | Click to apply and save any changes you have made to values on the page. |
| **Save** | Click to save entries on the Group Definition Detail page and return to the Group Definitions page. |
| **Id** | Upon saving a new group definition, a unique key identifier is assigned to the group definition.<br><br>For an existing group definition, displays the unique key identifier assigned to the group definition. |
| **Name** | For a new group definition, enter a descriptive name for the group definition.<br><br>For an existing group definition, displays the descriptive name assigned to the group definition. |
| **Process Name** | For a new group definition, enter the name of the process you want to associate with the group definition.<br><br>For an existing group definition, displays the process name associated with the group definition. This value is editable. |
| **PIP (Process Integration Pack) Name** | For a new group definition, enter the name of the PIP you want to associate with the group definition.<br><br>For an existing group definition, displays the PIP associated with the group definition. This value is editable. |

## Test Definition Selection

Select the **Test Definitions** tab to access the **Test Definition Selection** grid, where you can associate test definitions with the group definition.

| | |
|---|---|
| **Unassign** | Select one or more test definition rows that you want to disassociate from the group definition. Click **Unassign** to execute the disassociation. |
| **Assign** | Click to access the Search Definitions - Test page, where you can search for a test definition that you want to assign to the simulator definition. |

| | |
|---|---|
| **Refresh** | Click to refresh the Group Definition Detail page. |
| **Definition Sequence Id** | Displays the sequence in which the test definition is initiated by the group definition. |
| **Definition Id** | Click for an unlocked test definition to access the Modify Test Definition page.<br><br>Click for a locked test definition to access the View Test Definition page, where you can access a read-only view of the test definition. |

**For more information,** see How to Modify a Test Definition.

## Group Instance Selection

Select the **Group Instances** tab to display the **Group Instance Selection** grid, which displays information about group instances generated by the group definition.

| | |
|---|---|
| **Refresh** | Click to refresh the Group Definition Detail page. |
| **Id** | Click to access the Group Instances Detail page. |
| **Start Date** | Displays the date and time at which the group instance was initiated. |

# 8. Defining CAVS Routing Setup IDs

This chapter discusses the following topics:



## 8.1. Introduction to CAVS Routing Setup IDs

Composite Application Validation System (CAVS) routing setups are used in the following two scenarios.

- CAVS test definitions call services that in turn, call CAVS simulators.
- Actual applications and services call CAVS simulators instead of calling subsequent actual services.

CAVS routing setup IDs are used to route the service calls to the CAVS simulators. Use the pages covered in this chapter to set up CAVS routing setup IDs before executing tests. These CAVS routing setup IDs are stored as RouteToCAVS properties in the AIAConfigurationProperties.xml file in <AIA_HOME>/aia_instances/$INSTANCE_NAME/AIAMetaData/config. This file is read during runtime to determine whether routing needs to be made to a CAVS simulator or to an actual system.

For example, you could create three routing setup IDs for the scenarios illustrated below.

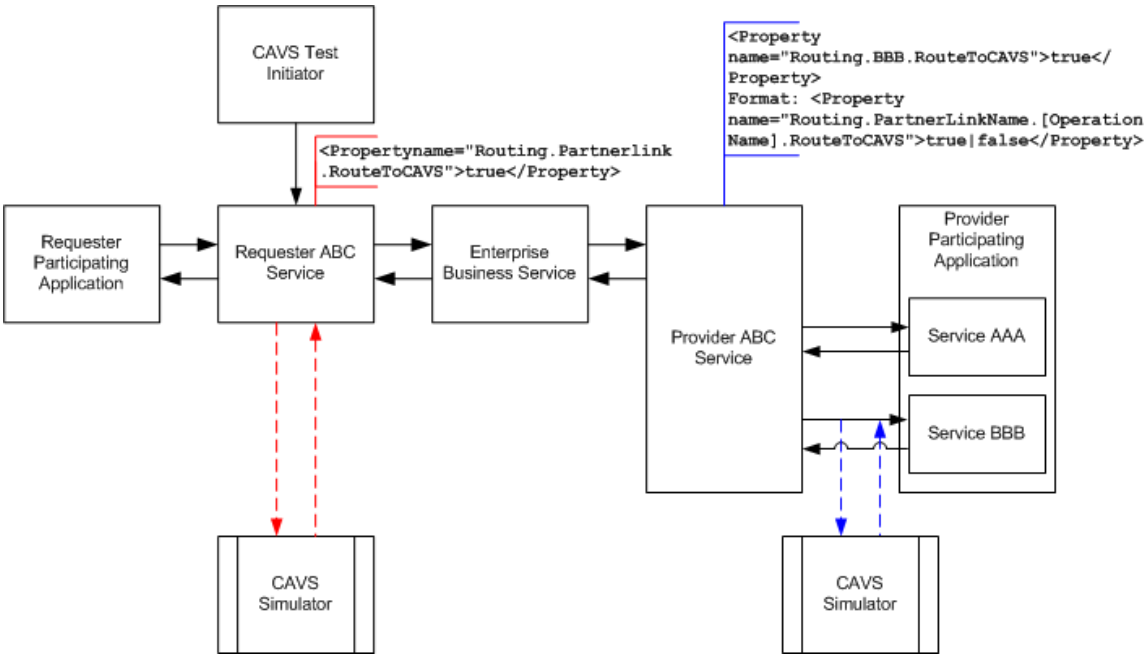
1. To test the requester Application Business Connector Service (ABCS) or when the provider ABCS is not available, you would want the requester ABCS to call a simulator instead of actual Oracle AIA services. For this scenario, create a routing setup ID to set the RouteToCAVS property to TRUE on the requester ABCS. This will ensure that the message is routed to the CAVS simulator, as indicated in red.

   **Note.** An actual participating application or test definition can be used to invoke the requester ABCS.

2. To test the provider ABCS or when the provider application is not available, you would want the provider ABCS to call a simulator instead of an actual provider application service. For this scenario, create a routing setup ID to set the RouteToCAVS property to TRUE on the provider ABCS. This will ensure that the message is routed to the CAVS simulator, as indicated in blue.

   **Note.** If there is more than one callout from the provider ABCS, the CAVS user can have fine-grained control over the routing by setting the routing at the PartnerLink level (and optionally at the operation level). This is indicated in the figure.

3. To test the requester ABCS and the provider ABCS together, you would create a routing setup ID to set the RouteToCAVS property to FALSE on the requester ABCS so that it can go on to call the provider ABCS and TRUE on the provider ABCS.




### Sample scenarios for using CAVS routing setup IDs

This figure helps to illustrate the need for different routing setup IDs to test each of these three scenarios. When creating test definitions that will be used to initiate these test scenarios, CAVS allows you to associate the test definition with a specific routing setup ID. This routing setup ID determines the configuration that is required and automatically applies it before executing the test.

For example, if these three test scenarios are grouped into a single test group for execution, each test requires a different routing setup. In this case, you would create three routing setup IDs, 1001, 1002, and 1003, for example. Each routing setup ID is required by one of the scenarios. You assign routing setup ID 1001 to the test definition for scenario 1, 1002 to the test definition for scenario 2, and so forth. When these three test definitions are executed as a part of the test group, the CAVS system automatically applies routing setup IDs 1001, 1002, and 1003 when executing the appropriate test definition. This eliminates the need to manually modify routing configurations between test scenario executions.

If, for example, you did not associate routing setup ID 1002 with the test definition for scenario 2, the test definition for scenario 2 would use routing setup ID 1001, because it was the last applied routing setup ID.

**For more information** about assigning a routing setup ID to a test definition, see How to Create a Test Definition.

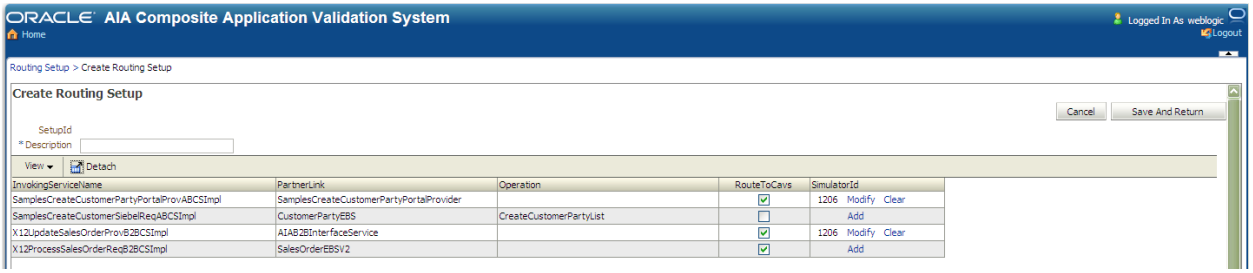
Another option for applying routings is to directly modify them on the Configuration page.

**For more information** about the Configuration page, see How to Set Up CAVS Routing Configurations Without Creating Routing Setup IDs.

## 8.2. How to Create CAVS Routing Setup IDs

To create CAVS routing setup IDs:

1. Access the Create Routing Setup page. To access the page, access the Oracle Application Integration Architecture (AIA) Home Page. In the Composite Application Validation System area, click the Go button. Select the Routing Setup tab. Click the Create button.




Create Routing Setup page

2. Upon access, the Create Routing Setup page displays routing information for all services with a RoutetoCAVS property defined in the AIAConfigurationProperties.xml file in <AIA_HOME>/aia_instances/$INSTANCE_NAME/AIAMetaData/config.

Use this page to perform a one-time setup of routing setup IDs that you can later associate with test definitions using the SetupId field on the Create Test page. By making this association, the required routing setup will be automatically applied during the execution of the test definition.

**For more information** about the SetupId field, see How to Create a Test Definition.

Data saved on this page is stored in a CAVS table, rather than in the AIAConfigurationProperties.xml.

**For more information** about how to quickly define a routing configuration that is stored in AIAConfigurationProperties.xml, see How to Set Up CAVS Routing Configurations Without Creating Routing Setup IDs.

Use the following page elements on the Create Routing Setup page to create a new CAVS routing.

| | |
|---|---|
| **SetupId** | Upon saving, a sequentially generated ID is assigned to the routing setup ID. |
| **Description** | Enter a description of the routing setup ID you are creating. |
| **InvokingServiceName** | Lists all services defined in the AIAConfigurationProperties.xml file in <AIA_HOME>/aia_instances/$INSTANCE_NAME/AIAMetaData/config. |
| **PartnerLink** | The PartnerLink that is invoked by the service that you want to route to the CAVS simulator. |
| **Operation** | The operation of the PartnerLink that you want to route to the CAVS simulator. Displays a value only when multiple operations on the service are invoked using the same PartnerLink, typically when calling an enterprise business service. |
| **RouteToCavs** | Select to indicate that the invoking service should route to the selected CAVS simulator. |
| **SimulatorId** | Click Add to access the Search Definitions page, where you can select the simulator definition that you want an invoking service to route to. Upon access, the page displays all available CAVS simulator definition IDs. Select the simulator definition to which you want to route an |

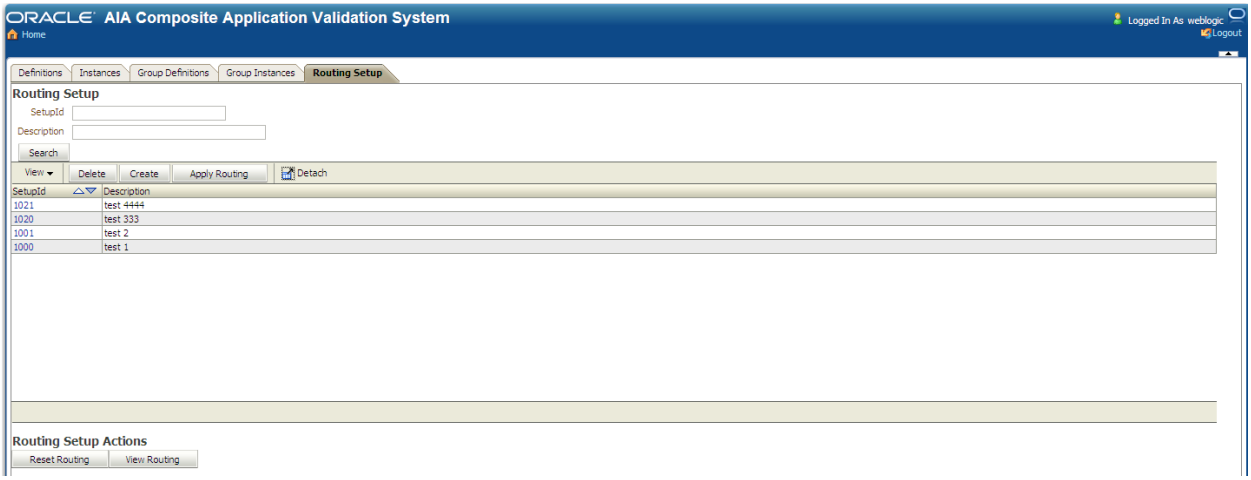
invoking service and click the Select button.

If a simulator definition has already been selected, the simulator ID displays. Click Modify to select a different simulator ID. Click Clear to clear the selection.

## 8.3. How to Search for CAVS Routing Setup IDs

To search for CAVS routing setup IDs:

1. Access the Routing Setup page. To access the page, access the AIA Home Page. In the Composite Application Validation System area, click the Go button. Select the Routing Setup tab.




Routing Setup page

2. Use the following page elements to search for an existing CAVS routing setup ID, or access functionality to create and delete routings.

**SetupId** — Enter the ID assigned to the routing setup ID you are searching for.

**Description** — Enter description text used for the routing setup ID you are searching for.

**Search** — Click to execute a search for routing setup IDs using the search criteria entered in the **Search Routing Setups** group box.

**Delete** — Select one or more routing setup IDs that you want to delete and click **Delete** to execute the deletion.

**Create** — Click to access the Create Routing Setup page, where you can create a routing setup ID.

**For more information**, see How to Create CAVS Routing Setup IDs.

**Apply Routing** — Once you have created a new routing setup ID, you may apply it to populate the AIAConfigurationProperties.xml file. To do this, select a single routing setup ID and click **Apply Routing**.

If you apply the routing setup ID to the AIAConfigurationProperties.xml file, it becomes a routing configuration that is applied in all executions of the associated invoking service, not just when the routing setup ID is referenced on a test definition.

**SetupId** — Click to access the Routing Setup page, where you can modify an existing routing setup ID.

**For more information** the Routing Setup page, see How to Modify Routing Setup IDs.

### Routing Setup Actions

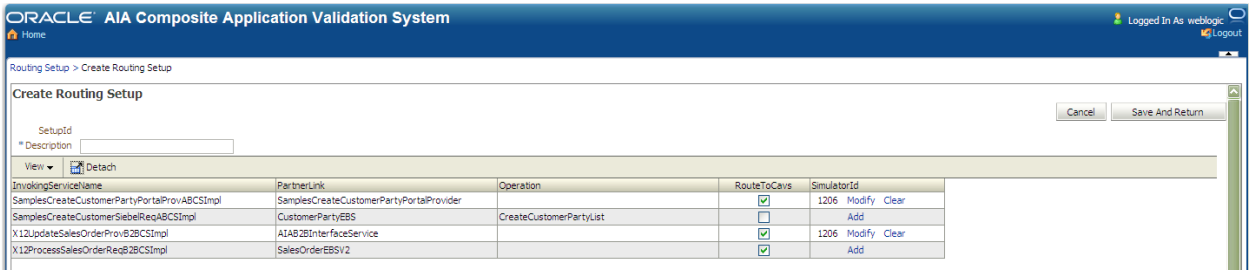
**Reset Routing** — Click to set all routing configurations to ***FALSE.*** This means that all routings to simulators (RoutetoCAVS property settings) in the AIAConfigurationProperties.xml file will be set to ***FALSE,*** whether you have defined them through the Routing Setup pages or directly in the file.

**View Routing** — Click to access the Configuration page, where you can access a read-only view of the last applied, or active, routing setup ID.

## 8.4. How to Modify Routing Setup IDs

To modify routing setup IDs:

1. Access the Routing Setup page. To access the page, access the AIA Home Page. In the Composite Application Validation System area, click the Go button. Select the Routing Setup tab. Click a SetupId link.




Routing Setup page

2. Use the following page elements on the Routing Setup page to modify existing routing setups. Data saved on this page is stored in a CAVS table, rather than in the AIAConfigurationProperties.xml.

   If you want to apply the data to the AIAConfigurationProperties.xml file, you must click Apply Routing for the routing setup ID on the Search Routing Setups page.

**For more information** about the Apply Routings button, see How to Search for CAVS Routing Setup IDs.

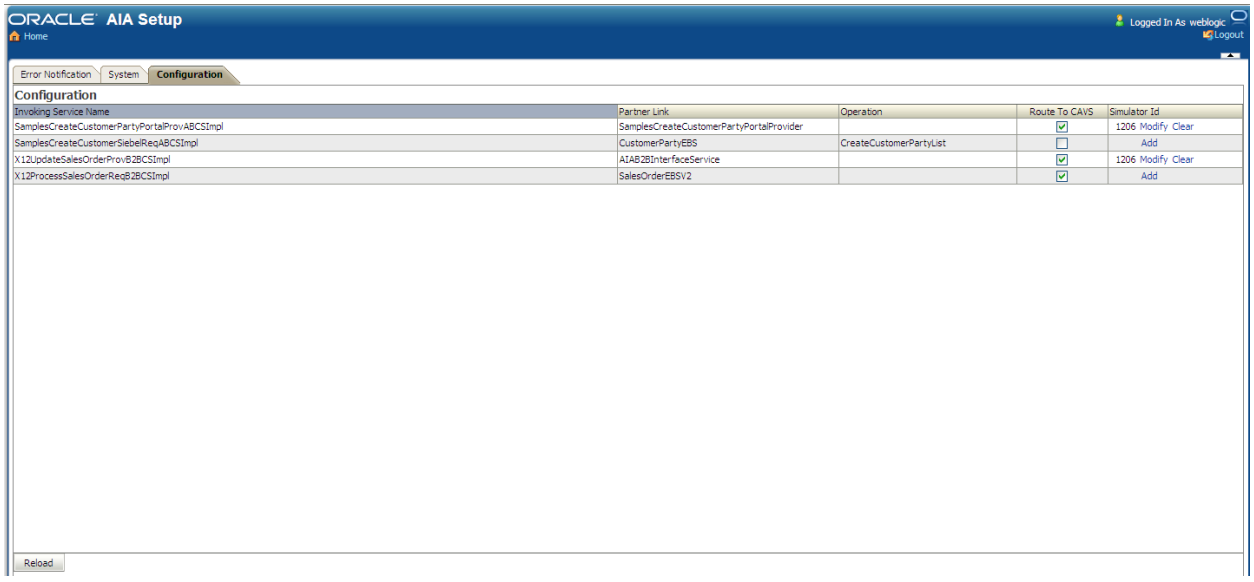
**SetupId** — Displays the ID you assigned to routing setup ID on the Create

| | |
|---|---|
| | Routing Setup page. |
| **Description** | If applicable, edit the routing setup ID description. |
| **Invoking Service Name** | This is the service after which the service routing to CAVS should happen. |
| **PartnerLink** | The PartnerLink that is invoked by the service that you want to route to the CAVS simulator. |
| **Operation** | The operation of the PartnerLink that you want to route to the CAVS simulator. Displays a value only when multiple operations on the service are invoked using the same PartnerLink, typically when calling an enterprise business service. |
| **RouteToCavs** | Select to indicate that the invoking service should route to the selected CAVS simulator. |
| **SimulatorId** | Click the icon to access the Search Definitions page, where you can select the simulator definition that you want an invoking service to route to.<br><br>If a simulator definition has already been selected, the simulator ID displays. Click Modify to select a different simulator ID. Click Clear to clear the selection. |

# 8.5. How to Set Up CAVS Routing Configurations Without Creating Routing Setup IDs

To set up CAVS routing configurations without creating routing setup IDs:

1. Access the Configuration page. To access the page, access the AIA Home Page. In the Setup area, click the Go button. Select the Configuration tab.




Configuration page

2. Use this page to quickly set up a CAVS routing configuration without having to create routing setup IDs. This is particularly useful when you are only interested in using CAVS simulators without CAVS test definitions.

   For example, you may only need to use the CAVS simulator feature for your development purposes and you may not need to uptake the complexity involved in setting up routing setup IDs. In this case, you can use this page to directly modify service routing configurations in the AIAConfigurationProperties.xml file.

   **Note.** If you use this page to modify these service routing configurations, there is no need to manually reload the configurations.

   However, if you are using CAVS for extensive testing purposes, we recommend that you use the Routing Setup pages to create your routing setup.

   **For more information** about the Routing Setup page, see How to Create CAVS Routing Setup IDs.

# 9. Working with Test and Simulator Instances

A test instance captures the details of the execution of a test definition. A simulator instance captures the details of a simulator definition's behavior during the execution of a test definition with which it is associated.

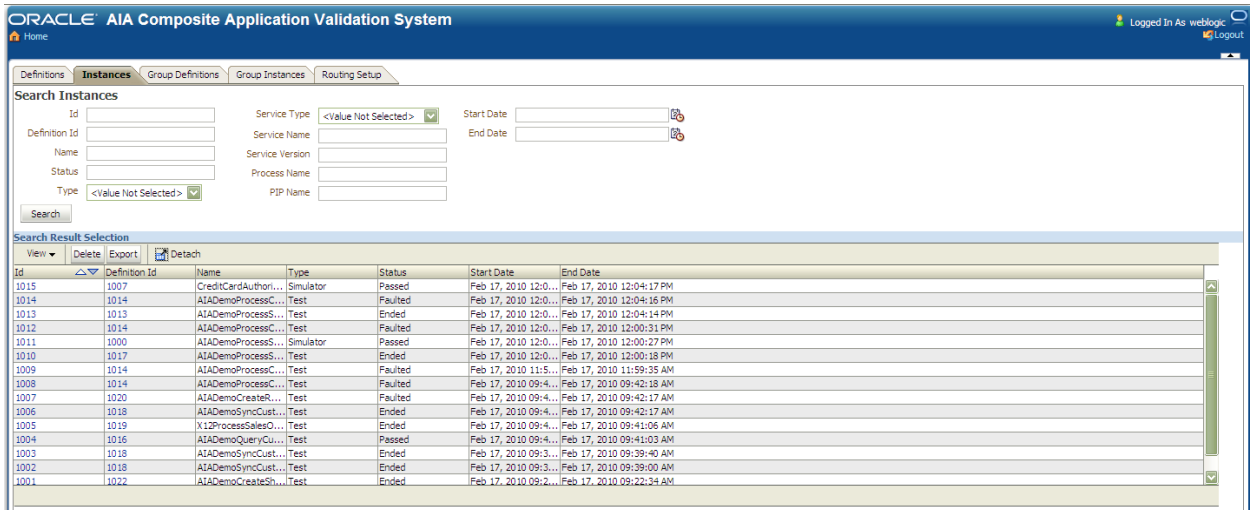
This chapter discusses:

- How to Work with Test and Simulator Instances
- How to View Test Instance Details
- How to View Simulator Instance Details

## 9.1. How to Work with Test and Simulator Instances

To work with test and simulator instances:

1. Access the Instances page. To access the page, access the Oracle Application Integration Architecture (AIA) Home Page. In the Composite Application Validation System area, click the Go button. Select the Instances tab.




Instances page

2. Use the following page elements on the Instance page to search for test and simulator instances. You can also access pages you can use to view test and simulator instance details.

### Search Instances

Use the **Search Instances** group box to enter search criteria to locate the instance you are searching for.

| | |
|---|---|
| **Id** | Enter the unique key assigned to the instance. |
| **Definition Id** | Enter a definition ID associated with the definition that generated the instance. |

| | |
|---|---|
| **Name** | Enter the descriptive name given to the definition that generated the instance. |
| **Status** | Enter the status of the instance.<br><br>***Ended:*** This status is only applicable to instances that do not involve validations. Indicates that the instance has ended.<br><br>***Faulted:*** The instance could not execute properly due to exceptions or faults.<br><br>***Failed:*** The instance did not pass validation.<br><br>***Passed:*** The instance passed validation.<br><br>***Delayed:*** For an asynchronous two-way test instance, indicates that the test instance is still active and waiting for an asynchronous reply. |
| **Type** | Select the type of instance.<br><br>***<Value Not Selected>***<br><br>***Test***<br><br>***Simulator*** |
| **Service Type** | Select the business service pattern of the web service associated with the instance.<br><br>For example, if you are searching for a test instance, this is the business service pattern of the web service tested by the test definition that generated the test instance. If you are searching for a simulator instance, this is the business service pattern of the web service simulated by the simulator definition that generated the simulator instance.<br><br>***<Value Not Selected>***<br><br>***Synchronous***<br><br>***Notify***<br><br>***Asynchronous two way*** |
| **Service Name** | Enter the name of the web service associated with the definition that created the instance. |
| **Service Version** | Enter the version of the web service associated with the definition that created the instance. |
| **Process Name** | Enter the name of the process associated with the definition that created the instance. |
| **PIP Name (Process Integration Pack name)** | Enter the name of the Process Integration Pack (PIP) associated with the definition that created the instance. |
| **Start Date** | Enter a start date and time that you want to use as search criteria. The search will look for all instances that were created on and after the given date and time. |
| **End Date** | Enter an end date and time that you want to use as search criteria. The search will look for all instances that were created before and |

on the given date and time.

**Search** — Click to execute a search for instances using the search criteria entered in the Search Instances group box.

### Search Result Selection

Use the **Search Result Selection** grid to work with instances returned in your search results. Upon accessing this page, the grid is populated by all instances.

**Delete** — Select one or more instances that you want to delete and click the **Delete** button to execute the deletion.

**Export**

**For more information** about exporting instances, see Exporting and Importing CAVS Definitions and Instances.

**Id** — Click for a test instance to access the Test Instance Detail page.

**Definition Id** — Click for a simulator instance to access the Simulator Instance Detail page.

For a test instance, click to access details about the test definition that generated the test instance. An unlocked test definition displays on the Modify Test Definition page. A locked test definition displays on the View Test Definition page.

**For more information,** see How to Modify a Test Definition.

For a simulator instance, click to access details about the simulator definition that generated the simulator instance. An unlocked simulator definition displays on the Modify Simulator Definition page. A locked test definition displays on the View Simulator Definition page.

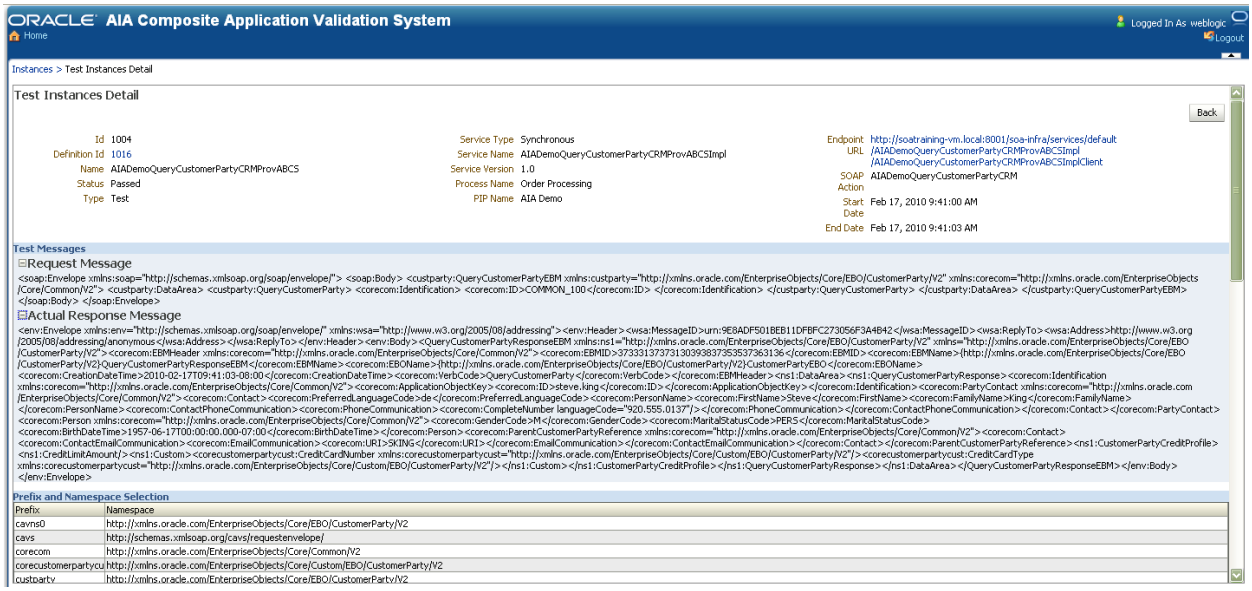
**For more information,** see How to Modify a Simulator Definition.

## 9.2. How to View Test Instance Details

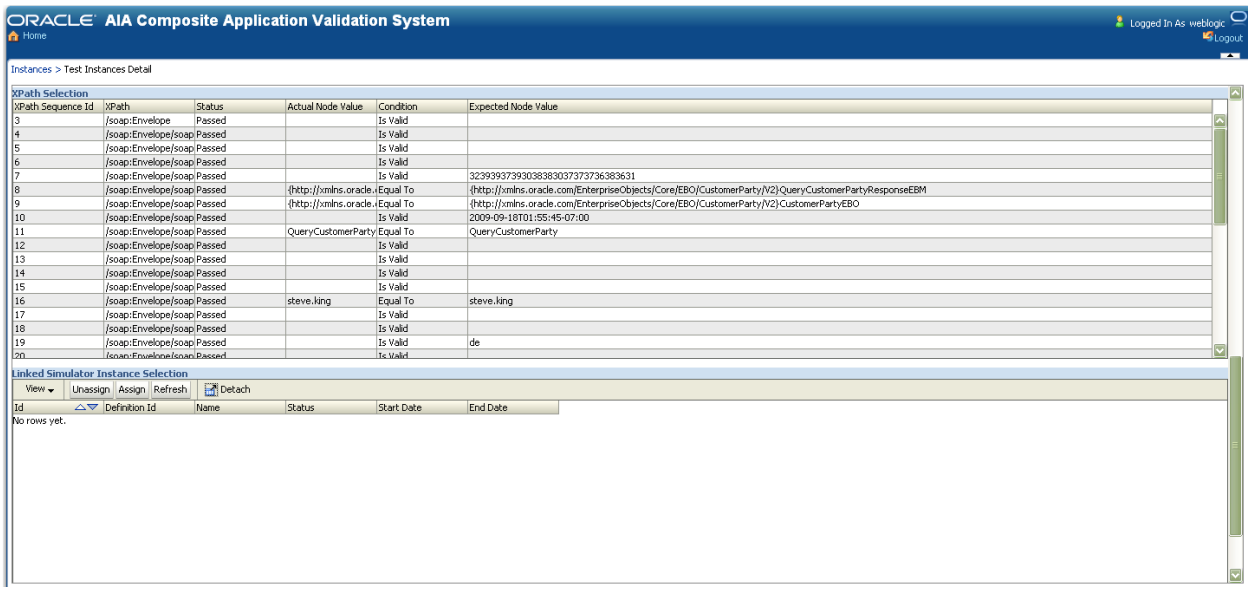
To view test instance details:

1. Access the Test Instances Detail page. To access the page, use one of the following navigation paths:
   - Select Execute in the Actions drop-down list box on the Modify Test Definition page. To access the Modify Test Definition page, access the AIA Home Page. In the Composite Application Validation System area, click the Go button. Select the Definitions tab. Click a test definition Id link.
   - Click an instance ID link in the Test Instances group box on the Modify Test Definition page. To access the Modify Test Definition page, access the AIA Home Page. In the Composite Application Validation System area, click the Go button. Select the Definitions tab. Click a test definition Id link.

- Click an instance ID link on the Instances page. To access the Instances page, access the AIA Home Page. In the Composite Application Validation System area, click the Go button. Select the Instances tab.




Test Instances Detail page (1 of 2)




Test Instances Detail page (2 of 2)

2. Use the following page elements on the Test Instances Detail page to view the details of a test instance.

| | |
|---|---|
| **Cancel** | Click to discard any updates to the page and return to the Instances page. |
| **Apply** | Click to apply and save any updates you have made to the page. |
| **Save** | Click to save any updates you have made to the page and go to the Instances page. |

| | |
|---|---|
| **Id** | Displays the unique ID assigned to the instance. |
| **Definition Id** | Displays the ID of the test definition that generated the test instance.<br><br>Click for an unlocked test definition to access the Modify Test Definition page.<br><br>Click for a locked test definition to access the View Test Definition page, where you can access a read-only view of the test definition.<br><br>**For more information,** see How to Modify a Test Definition. |
| **Name** | Displays the descriptive name associated with the test definition that generated the instance. |
| **Status** | Displays the status of the test instance.<br><br>***Ended:*** This status is only applicable to test instances that do not involve validations. Indicates that the instance has ended.<br><br>***Faulted:*** The test instance could not execute properly due to exceptions or faults.<br><br>***Failed:*** The test instance did not pass validation.<br><br>***Passed:*** The instance passed validation.<br><br>***Delayed:*** For an asynchronous two-way test instance, indicates that the test instance is still active and waiting for an asynchronous reply. |
| **Type** | Displays the type of definition that generated the test instance. On the Test Instances Detail page, this value will always be ***Test.*** |
| **Service Type** | Displays the business service pattern of the web service tested by the test definition that generated the test instance.<br><br>***Synchronous***<br><br>***Notify***<br><br>***Asynchronous two way*** |
| **Service Name** | Displays the name of the web service tested by the test definition that created the instance. |
| **Service Version** | Displays the version of the web service tested by the test definition that created the instance. |
| **Process Name** | Displays the name of the process associated with the test definition that created the instance. |
| **PIP Name (Process Integration Pack name)** | Displays the name of the PIP associated with the test definition that created the instance. |
| **Endpoint URL** | Displays the URL of the web service tested by the test definition that created the instance. |
| **SOAP Action** | Displays the operation called by the web service tested by the test definition that created the instance. |

| | |
|---|---|
| **Start Date** | Displays the date and time at which the test instance was initiated. |
| **End Date** | Displays the date and time at which the test instance ended. |

### Test Messages

Use the **Test Messages** group box to view the request and response XML messages associated with the test definition that generated the instance.

| | |
|---|---|
| **Request Message** | Displays request message XML defined for the test definition that generated the test instance. |

**For more information** about the **Request Message** field, see How to Create a Test Definition.

| | |
|---|---|
| **Actual Response Message** | Displays response message XML defined for the test definition that generated the test instance. |

**For more information** about the **Response Message** field, see How to Create a Test Definition.

### Prefix and Namespace Selection

Displays namespace data created for the test definition that generated the test instance. This namespace data is used in the XPath values defined in the **XPath Selection** grid.

**For more information** about the **Prefix and Namespace Selection** grid, see How to Modify a Test Definition.

### XPath Selection

Displays XPath data created for the test definition that generated the test instance. The values in this grid use the namespace values set in the **Prefix and Namespace Selection** grid.

**For more information** about the **XPath Selection** grid, see How to Modify a Test Definition.

### Linked Simulator instance Selection

Use the **Linked Simulator instance Selection** grid to work with associations between test instances and simulator instances.

If no correlation logic has been defined between the test definition and the simulator definition, the test and simulator instances will not always be reconcilable, especially when the same web service is invoked multiple times during a very short time period, as in during performance testing.

However, if a simulator definition is associated with a test definition, any test instances generated by the test definition will automatically reflect associations to simulator instances generated by associated simulator definitions.

You can manually adjust these associations in this grid area.

| | |
|---|---|
| **Unassign** | Select one or more simulator instance rows that you want to disassociate with the test instance. Click the Unassign button to execute the disassociation. |

**Assign**

Click to access the Search Instances - Simulator page, where you can search for a simulator instance that you want to manually associate with the test instance.

Once you have associated a simulator instance with the test instance using the Search Instances - Simulator page, the Test Instances Detail page displays the selected simulator instance.

**Refresh**

Click to refresh the Test Instances Detail page.

**Id**

Click to access the Simulator Instances Detail page.

**Definition Id**

Click to view details about the test definition that generated the test instance.

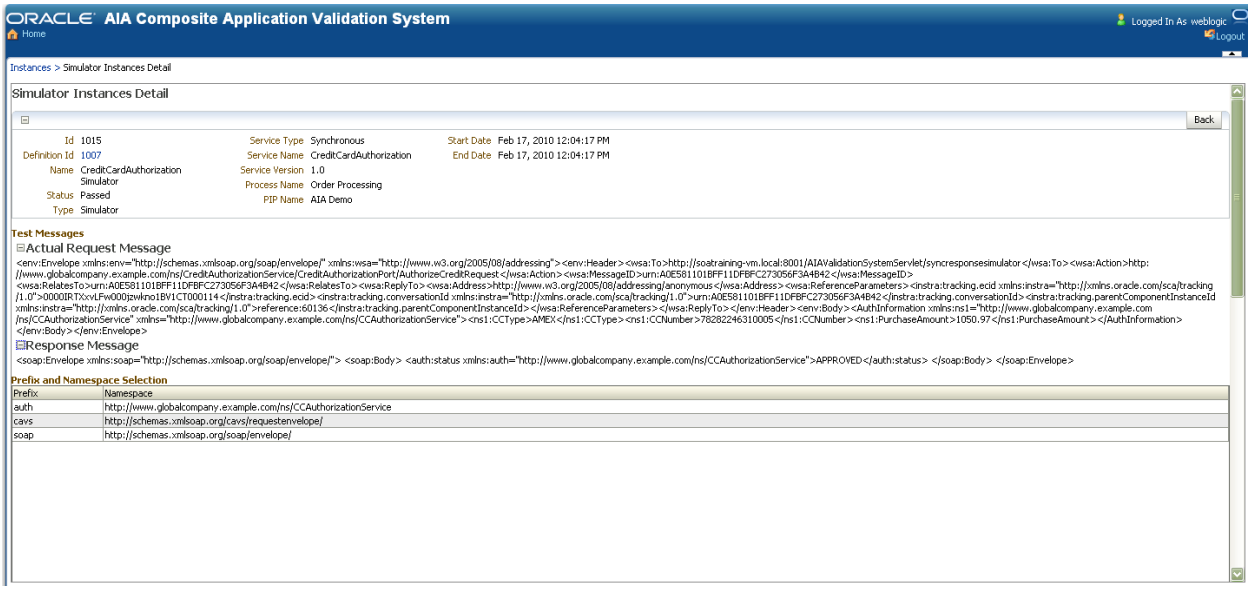
An unlocked test definitions display on the Modify Test Definition page. A locked test definition displays on the View Test Definition page.

**For more information,** see How to Modify a Test Definition.

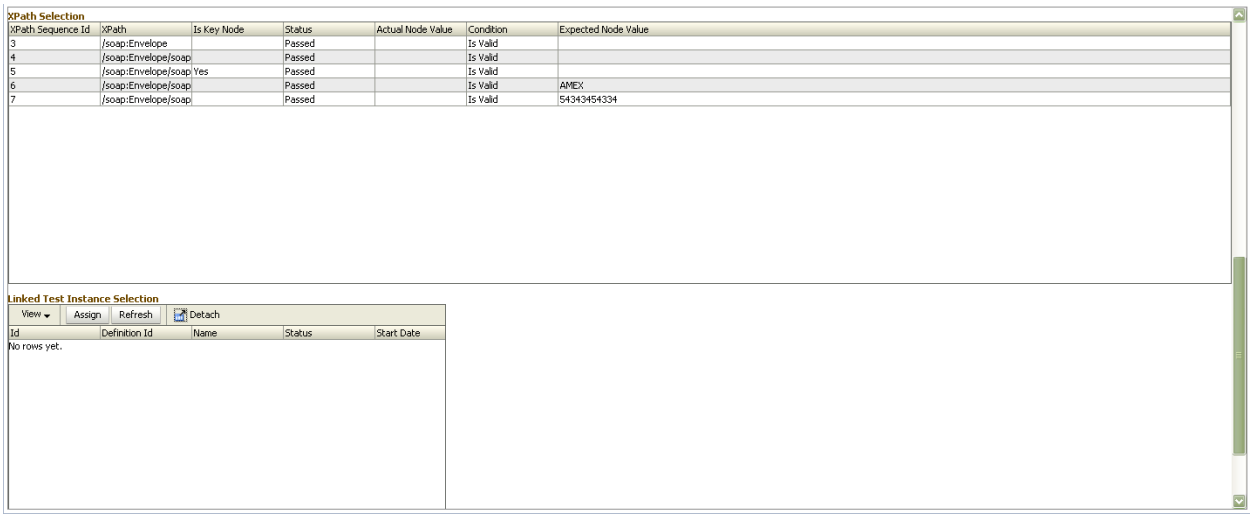
## 9.3. How to View Simulator Instance Details

To view simulator instance details:

1. Access the Simulator Instances Detail page. To access the page, click the Instance Id link for a simulator instance on the Instances page.




Simulator Instances Detail page (1 of 2)




Simulator Instances Detail page (2 of 2)

2. Use the following page elements on the Simulator Instances Detail page to view the details of a simulator instance.

**Cancel** — Click to discard any updates to the page and return to the Instances page.

**Apply** — Click to apply and save any updates you have made to the page.

**Save** — Click to save any updates you have made to the page and go to the Instances page.

**Id** — Displays the unique ID assigned to the instance.

**Definition Id** — Click for an unlocked simulator definition to access the Modify Simulator Definition page.

Click for a locked simulator definition to access the View Simulator Definition page, where you can access a read-only view of the simulator definition.

**Name** — Displays the descriptive name associated with the simulator definition that generated the instance.

**Status** — Displays the status of the simulator instance.

***Initiated:*** The simulator instance has been initiated.

***Ended:*** This status is only applicable to simulator instances that do not involve validations. Indicates that the instance has ended.

***Faulted:*** The simulator instance could not execute properly due to exceptions or faults.

***Failed:*** The simulator instance did not pass validation.

***Passed:*** The simulator instance passed validation.

**Type** — Displays the type of definition that generated the simulator instance. On the Simulator Instances Detail page, this value will always be ***Simulator.***

| | |
|---|---|
| **Service Type** | Displays the business service pattern of the web service simulated by the simulator definition that generated the instance.<br>***Synchronous***<br>***Notify***<br>***Asynchronous two way*** |
| **Service Name** | Displays the name of the web service simulated by the simulator definition that created the instance. |
| **Service Version** | Displays the version of the web service simulated by the simulator definition that created the instance. |
| **Process Name** | Displays the name of the process associated with the simulator definition that created the instance. |
| **PIP Name (Process Integration Pack name)** | Displays the name of the PIP associated with the simulator definition that created the instance. |
| **Start Date** | Displays the date and time at which the simulator instance was initiated. |
| **End Date** | Displays the date and time at which the simulator instance ended. |

### Test Messages

Use the **Test Messages** group box to view the request and response XML messages associated with the simulator definition that generated the instance.

| | |
|---|---|
| **Actual Request Message** | Displays request message XML defined for the simulator definition that generated the instance.<br>**For more information** about the **Request Message** field, see How to Create a Simulator Definition. |
| **Response Message** | Displays response message XML defined for the simulator definition that generated the instance.<br>**For more information** about the **Response Message** field, see How to Create a Simulator Definition. |

### Prefix and Namespace Selection

Displays namespace data created for the simulator definition that generated the simulator instance. This namespace data is used in the XPath values defined in the **XPath Selection** grid.

**For more information** about the **Prefix and Namespace Selection** grid, see How to Modify a Simulator Definition.

### XPath Selection

Displays XPath data created for the simulator definition that generated the instance. The values in this grid use the namespace values set in the **Prefix and Namespace Selection** grid.

**For more information** about the **XPath Selection** grid, see How to Modify a Simulator Definition.

### Linked Test Instance Selection

Displays the test instance with which the simulator instance is associated. This is a one-to-one association.

If no correlation logic has been defined between the test definition and the simulator definition, the test and simulator instances will not always be reconcilable, especially when the same web service is invoked multiple times during a very short time period, as in during performance testing.

However, if a simulator definition is associated with a test definition, any test instances generated by the test definition will automatically reflect associations to simulator instances generated by associated simulator definitions.

You can adjust the association between the simulator instance and a test instance using the controls on this page.

| | |
|---|---|
| **Unassign** | Select the test instance ID that you want to disassociate from the simulator instance and click the **Unassign** button to execute the disassociation. |
| **Assign** | Click to access the Search Instances - Test page, where you can search for a test instance that you want to manually associate with the simulator instance.<br><br>Once you have associated a test instance with the simulator instance using the Search Instances - Test page, the Simulator Instances Detail page displays the selected test instance. |
| **Refresh** | Click to refresh the Simulator Instances Detail page. |
| **Id** | Click to display the selected test instance on the Test Instances Detail page |
| **Definition Id** | Displays the ID of the test definition that generated the test instance.<br><br>Click for an unlocked test definition to access the Modify Test Definition page.<br><br>Click for a locked test definition to access the View Test Definition page, where you can access a read-only view of the test definition. |

**For more information,** see How to Modify a Test Definition.

# 10. Working with Group Instances

A group instance captures the details of the execution of a group definition.

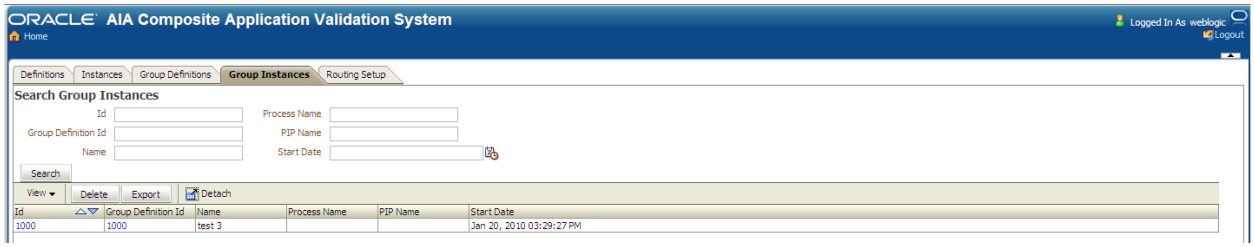
This chapter discusses:



## 10.1. How to View Group Instances

To view group instances:

1. Access the Group Instances page. To access the page, access the Oracle Application Integration Architecture (AIA) Home Page. In the Composite Application Validation System area, click the Go button. Select the Group Instances tab.




Group Instances page

2. Use the following page elements on the Group Instances page to search for group instances. Access a page you can use to view group instance details.

### Search Group Instances

Use the **Search Group Instances** group box to enter search criteria to find the group instance you are searching for.

| | |
|---|---|
| **Id** | Enter the unique key identifier assigned to the group instance. |
| **Group Definition Id** | Enter the unique key ID assigned to the group definition that generated the instance. |
| **Name** | Enter a descriptive name assigned to the group definition. |
| **Process Name** | Enter the name of the process associated with the group definition that generated the instance. |
| **PIP Name (process integration pack)** | Enter the name of the Process Integration Pack (PIP) associated with the group definition that generated the instance. |
| **Start Date** | Enter a start date and time that you want to use as search criteria. The search will look for all group instances that were created on |

and after the given date and time.

**Search** — Click to execute a search for group instances using the search criteria entered in the **Search Group Instances** group box.

### Search Result Selection

Use the **Search Result Selection** grid to work with group instances returned in your search results. Upon accessing this page, the grid is populated by all group instances.

**Delete** — Select one or more group instances that you want to delete and click the **Delete** button to execute the deletion.

**Export**

**For more information** exporting group instances, see Exporting and Importing CAVS Definitions and Instances.

**Id** — Click to access the Group Instances Detail page.

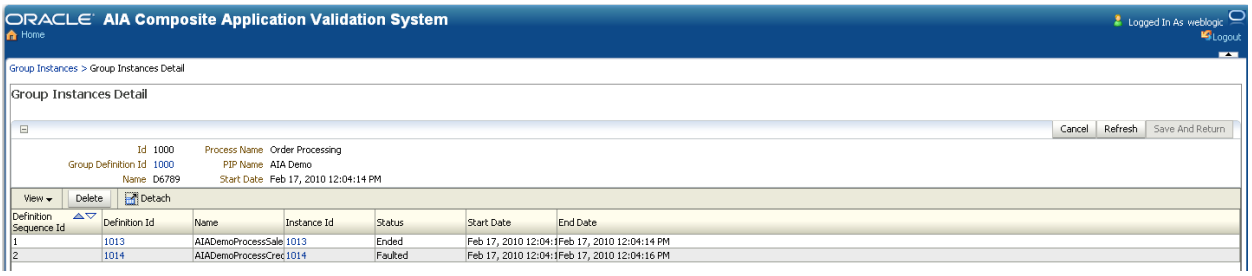
**Group Definition Id** — Click to access the Group Definition Detail page.

**For more information** about the Group Definition Detail page, see Working with Group Definitions.

## 10.2. How to View Group Instance Details

To view group instance details:

1. Access the Group Instances Detail page. To access the page, click a group instance Id link on the Group Instances page.




Group Instances Detail page

2. Use the following page elements on the Group Instances Detail page to view the details of a group instance.

**Id** — Displays the unique key identifier assigned to the group instance.

**Group Definition Id** — Click to access the Group Definition Detail page.

**Name** — Displays the descriptive name assigned to the group definition.

**Process Name** — Displays the name of the process associated with the group definition that generated the instance.

| | |
|---|---|
| **PIP Name** | Enter the name of the PIP associated with the group definition that generated the instance. |
| **Start Date** | Displays the date and time at which the group instance was initiated. |
| **Delete** | Select one or more test instance rows that you want to delete and click the **Delete** button to execute the deletion. |
| **Definition Sequence Id** | Indicates the sequence in which the test definitions were initiated by the group definition that generated the group instance. |
| **Definition Id** | Click to access the Modify Test Definition page.<br><br>**For more information** about the Modify Test Definition page, see How to Modify a Test Definition. |
| **Instance Id** | Click to access the Test Instances Detail page.<br><br>**For more information** about the Test Instances Detail page, see How to View Test Instance Details. |
| **Status** | Displays the status of the test instance in the group instance.<br><br>***Initiated:*** The test instance has been initiated.<br><br>***Ended:*** This status is only applicable to test instances that do not involve validations. Indicates that the instance has ended.<br><br>***Faulted:*** The test instance could not execute properly due to exceptions or faults.<br><br>***Failed:*** The test instance did not pass validation.<br><br>***Passed:*** The instance passed validation. |
| **Start Date** | Displays the date and time at which the test instance was initiated. |
| **End Time** | Displays the date and time at which the test instance ended. |

# 11. Purging CAVS-Related Cross Reference Entries to Enable Rerunning of Test Scenarios

When a participating application is involved in a Composite Application Validation System (CAVS) testing flow, execution of tests can potentially modify data in a participating application. Therefore, consecutive running of the same test may not generate the same results. The CAVS is not designed to prevent this kind of data tampering because it supports the user's intention to include a real participating application in the flow. The CAVS has no control over modifications that are performed in participating applications.

However, this issue does not apply if your CAVS test scenario uses test definitions and simulator definitions to replace all participating applications and other dependencies. In this case, all cross-reference data is purged once the test scenario has been executed. This enables rerunning of the test scenario.

This chapter discusses How to Purge CAVS-Related Cross Reference Entries to Enable Rerunning of Test Scenarios.

## 11.1. How to Purge CAVS-Related Cross Reference Entries to Enable Rerunning of Test Scenarios

To purge CAVS-related cross reference entries to enable rerunning of test scenarios:

1. Process integration packs (PIPs) that are delivered to work with Oracle Application Integration Architecture (AIA) Foundation Packs are delivered with cross-reference systems in place. They are named *CAVS_<XYZ>,* where *<XYZ>* is the participating application system. For example, for systems EBIZ and SEBL, the PIP is delivered with cross-reference systems *CAVS_EBIZ* and *CAVS_SEBL.*

2. For every system type defined on the System - Application Registry page for which you want to make test scenarios rerunnable (*<XYZ>*), create a related CAVS system (*CAVS_<XYZ>*). The System Type field value for the CAVS-related entry should match the name of the system for which it is created.

   **For more information** about the System – Application Registry page, see *Oracle Application Integration Architecture Foundation Pack: Development Guide,* "Building AIA Integration Flows."

3. When testing a provider Application Business Connector Service (ABCS) in isolation, the Enterprise Business Message (EBM) will be passed from the CAVS to the provider ABCS with the NamespacePrefixedEBMName/EBMHeader/Target/ID element set as CAVS_<XYZ>.

4. When testing a requester ABCS in isolation, the element in the Application Business Message (ABM) that normally contains the Internal ID value will now contain the CAVS-specific Internal ID value set for the system on the System – Application Registry page.

5. When testing an entire flow (requester ABCS-to-Enterprise Business Service [EBS] -to-provider ABCS), you must set the Default.SystemID property of the provider ABCS to CAVS_<XYZ>, where <XYZ> is the system.

**a.** To do this, edit the Default.SystemID property value in the AIAConfigurationProperties.xml file in the <AIA_HOME>/aia_instances/$INSTANCE_NAME/AIAMetaData/config directory.

**b.** Reload updates to the AIAConfigurationProperties.xml file.

> **For more information** about reloading updates to AIAConfigurationProperties.xml, see *Oracle Application Integration Architecture Foundation Pack: Development Guide*, "Building AIA Integration Flows."

**c.** You can now commence testing the entire flow.

**Note:** If the test scenario is an entire flow that includes multiple instances of the same system, this approach will not work. In this case, data created in the cross reference will remain making the same test case non-rerunnable.

# 12. Exporting and Importing CAVS Definitions and Instances

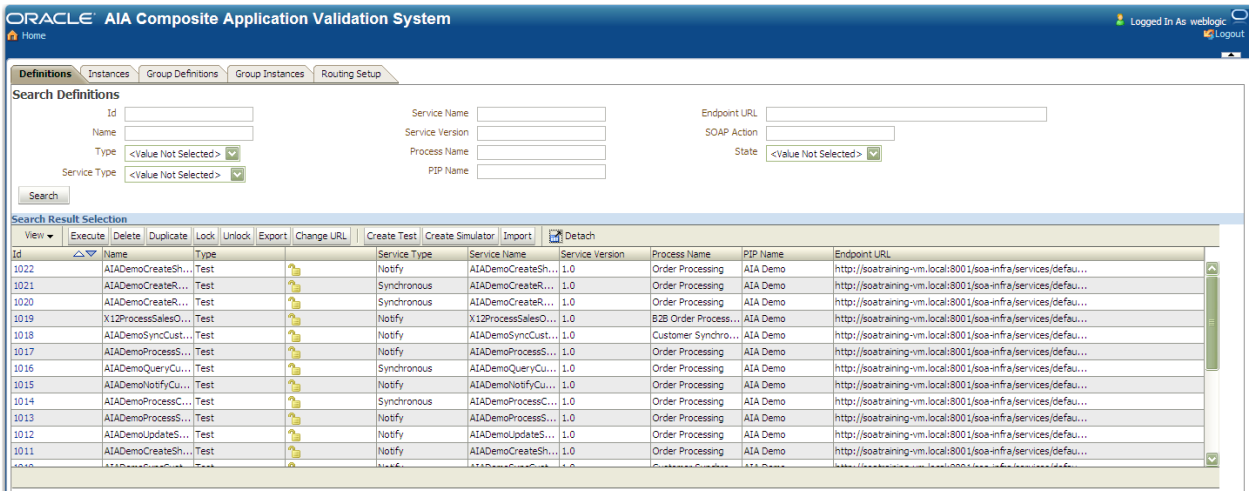
This chapter discusses:

- How to Export and Import Definitions
- How to Export Test and Simulator Instances
- How to Export Group Instances

## 12.1. How to Export and Import Definitions

To export and import CAVS definitions:

1. Access the Definitions page. To access the page, access the Oracle Application Integration Architecture (AIA) Home Page. In the Composite Application Validation System area, click the Go button. Select the Definitions tab.




Definitions page

2. Use the following page elements on the Definitions page to search for, execute, migrate, and manage existing test and simulator definitions. You can also access pages you can use to create and modify test and simulator definitions.

**Export**

Use the **Export** and **Import** buttons on this page to migrate test definitions, simulator definitions, and any associated group definitions in XML flat-file format between instances running on the same version of Foundation Pack.

Examples of uses for this export and import functionality include:

- QA may want to certify a set of definitions that have been run in one build in other builds.

- Support analysts and customers may want to exchange definition files.
- You may want to verify validity of new environments.

Select one or more definitions and click the **Export** button to initiate the export. The following options display:

***Export selected Definition(s) only***

***Export selected Definition(s) and associated Group Definition(s)***

***Export selected Definition(s), associated GroupDefinition(s) and Test Definition(s) that belong to the associated GroupDefinition(s) but are not selected***

Select an option and click the **Proceed** button to create and save the definitions to a location on your local system. The default file name for the exported definition(s) is ***Definitions.xml.***

If a test definition that you are exporting is associated with a routing setup ID, the routing setup information will also be exported.

If that routing setup is associated with one or more simulator definitions, which were provided when the Route To CAVS option was set to true, then these simulator definitions will also be exported.

**For more information** about the structure of the Definitions.xml file created by the CAVS export definition feature, see Appendix: XML Structures of Exportable CAVS Definitions and Instances.

**Import**

Use the **Import** button to upload a test, simulator, or group definition in the XML flat-file format generated by Composite Application Validation System (CAVS) export functionality. You can generate these files by clicking the **Export** button on this page. The definition file to be uploaded must be accessible by the local system being used to perform the upload.

Click the **Import** button and browse for the file you want to upload. The CAVS validates the structure of the file being uploaded. If the structure is invalid, an error will be raised.

If a test definition that you are importing is associated with a routing setup ID, the routing setup information will also be imported.

If that routing setup is associated with one or more simulator definitions, which were provided when the Route To CAVS option was set to true, then these simulator definitions will also be imported.

**For more information** about the valid structure of the Definitions.xml file created by the CAVS export definition feature, see Appendix: XML Structures of Exportable CAVS Definitions and Instances.

Imported definitions will still reference endpoint URLs pointing to tested web services in the source system. You must update imported definition endpoint URL values to point to tested web services in the target system. The CAVS enables you to update these URLs directly on the following pages:

Use the **Change URL** button on this page.

Update the Endpoint URL field value on the Modify Test Definitions page.

**For more information** about the Endpoint URL field, see How to Create a Test Definition.

Because the sequential definition IDs assigned in the source system may not be valid in the target system, new sequential definition IDs will be assigned by the target system. As a result, associations between definitions will be severed in the target system and will need to be reestablished.

Because test, simulator, and group instance details that may be associated with definitions in the source system are not valid in the target system, they will not be imported.

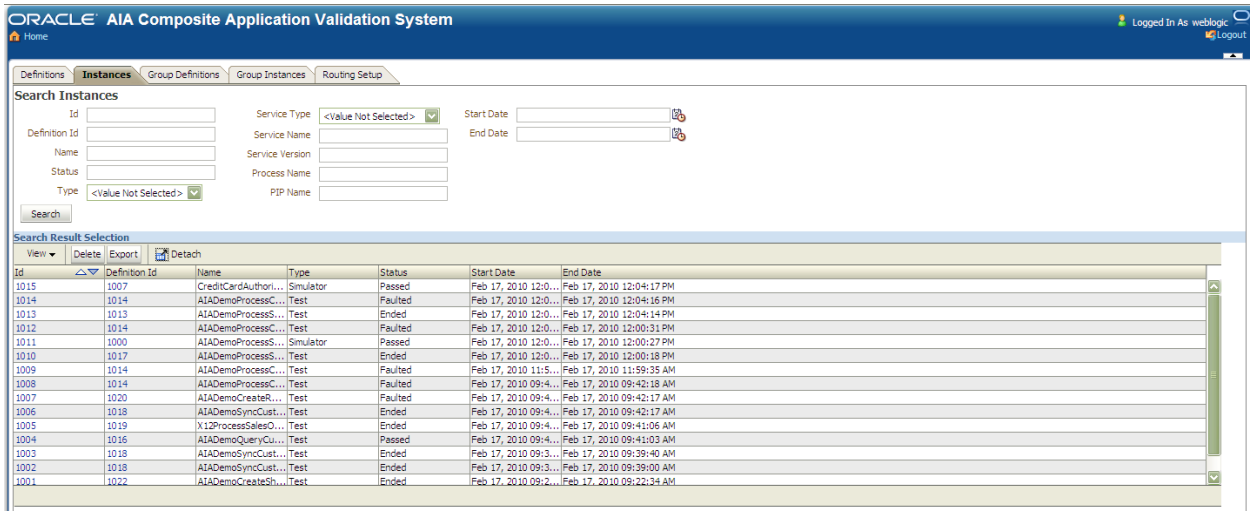
If the same definition is uploaded multiple times, multiple duplicate definitions will be created in the target system.

**For more information** about the Definitions page, see Searching for Test and Simulator Definitions.

## 12.2. How to Export Test and Simulator Instances

To export test and simulator instances:

1. Access the Instances page. To access the page, access the AIA Home Page. In the Composite Application Validation System area, click the Go button. Select the Instances tab.




Instances page

2. Use the following page elements on the Instances page to search for test and simulator instances. Access pages you can use to view test and simulator instance details.

**Export**

Use to export instances in XML format. You can use XML-based reporting tools to generate reports of test and simulator executions using these XML files.

Select one or more instances and click the **Export** button to initiate the export.

Click Save to create and save the definitions to a location on your local system. The default file name for the exported definition(s) is ***Instances.xml.***

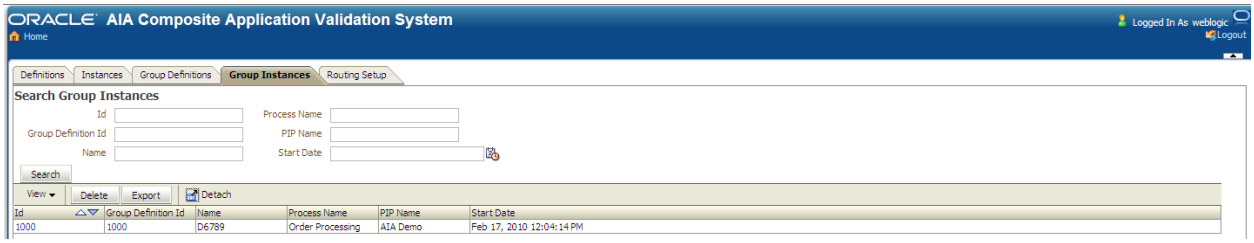
**For more information** about the structure of the Definitions.xml file created by the CAVS export instance feature, see Appendix: XML Structures of Exportable CAVS Definitions and Instances.

**For more information** about the Instances page, see Working with Test and Simulator Instances.

# 12.3. How to Export Group Instances

To export group instances:

1. Access the Group Instances page. To access the page, access the AIA Home Page. In the Composite Application Validation System area, click the Go button. Select the Group Instances tab.




Group Instances page

2. Use the following page element son the Group Instances page to search for group instances. Access a page you can use to view group instance details.

| | |
|---|---|
| **Export** | Select one or more group instances that you want to export and click the **Export** button to execute the download. |

**For more information** about the structure of the Definitions.xml file created by the CAVS export definition feature, see Appendix: XML Structures of Exportable CAVS Definitions and Instances.

**For more information** about the Group Instances page, see How to View Group Instances.

# Part: Setting Up and Using Error Handling and Logging

# 13. Introduction to Oracle AIA Error Handling

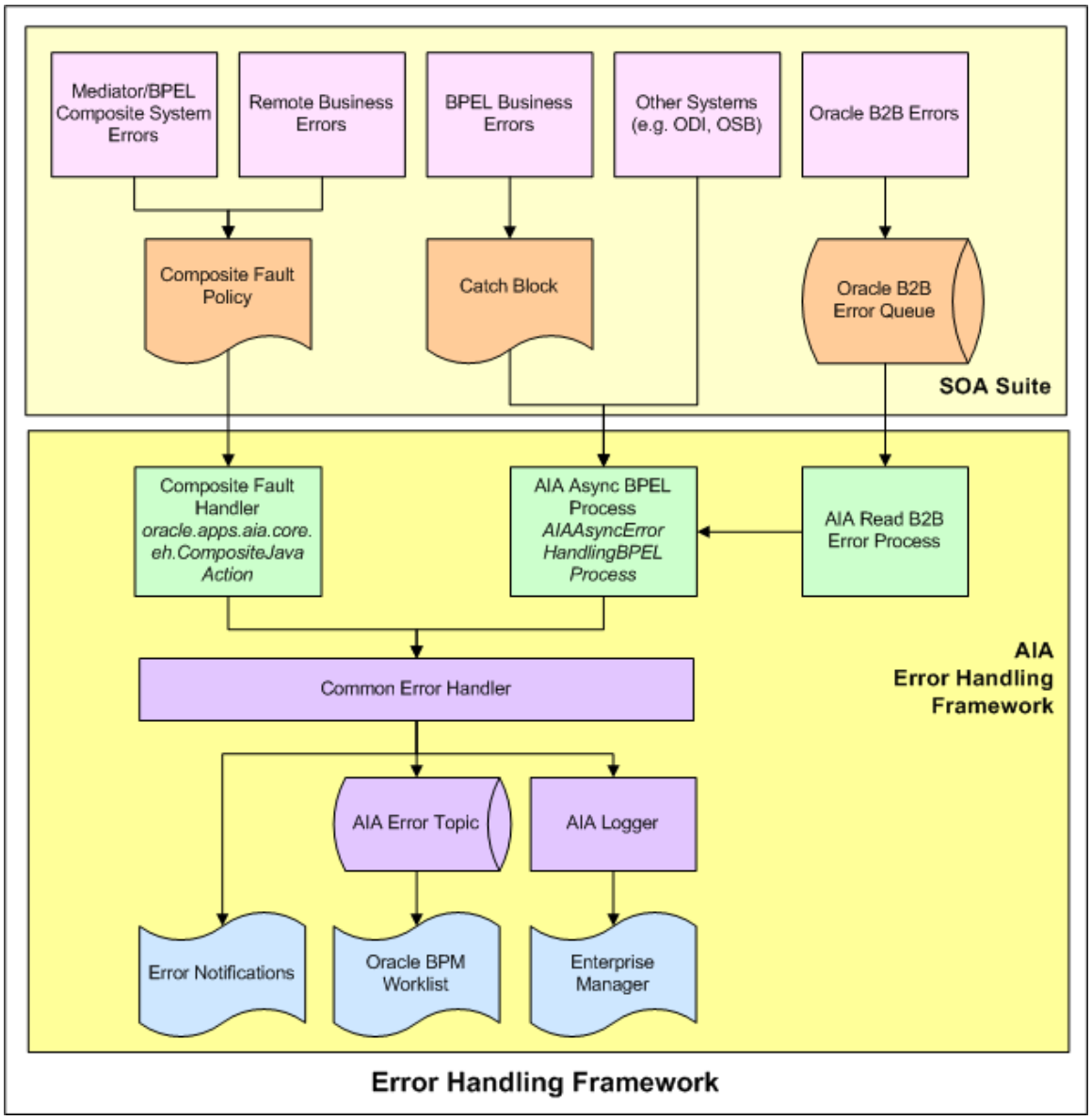
This chapter discusses the following topics:

- Introduction to the Error Handling Framework
- Introduction to Error Handling for Business Faults
- Introduction to Error Handling for BPEL and Mediator System Faults
- Introduction to Error Handling for Oracle B2B Errors

## 13.1. Introduction to the Error Handling Framework

The Error Handling Framework provides error handling and logging components to support the needs of integration services operating in an Oracle Application Integration Architecture (AIA) ecosystem.

This diagram provides a high-level overview of the Error Handling Framework.




## Error Handling Framework components Key Features

The AIA Error Handling Framework provides the following key features for integration services operating in an AIA ecosystem.

### Unified Error Handling Approach

- Works across technologies, including BPEL and Mediator components, business-to-business (B2B), and ODI.
- Works across categories of faults, including business and system/runtime/technical faults.
- Works across integration patterns.
- Adopts the Oracle SOA Suite 11g tech stack.

### Error Notifications

- Error notifications are emailed to suitable actor roles, such as integration administrators, and FYI roles, such as customer service representatives.
- Provides visibility into error context.
  - Drill-down to the Oracle Enterprise Manager Console Flow Trace page from the error notification email.
  - View errors in the context of an AIA flow trace.
- Enables customization of error notification content.
  - Add key fields to the error notification body.
  - Add or remove fields from error notification content.
  - Issue error notifications to suitable Actor and FYI roles.
  - Provide a link to Oracle BPM Worklist for error details, if desired.
- Enables error notification throttling.
  - Control the number of error notifications issued for a specific error.
  - Regulate the issuance of error notifications by time interval and number of errors.

### Oracle BPM Worklist Integration

- Centralized user interface to access error details that are assigned for resolution or for informational purposes.
- Accessible to administrators and end-users.
- Decoupled from the Error Notification Framework.
  - Oracle BPM Worklist is not tied to error notifications.
  - Oracle BPM Worklist can be used as an optional component.

### Error Logging

- Logs messages non-intrusively in a consistent schema.
- Logs can be searched, sorted, and filtered using Oracle Enterprise Manager.

### B2B Error Handling

- Errors in the Oracle B2B component of Oracle Fusion Middleware are routed to the AIA Error Handling Framework.
- The AIA fault definition captures B2B-specific details of a failed AIA flow.

### Extensible Framework

- Provides the ability to extend error handling capabilities.

#### Automated Error Actions

- Automatically acts upon the errored object to provide automated retry actions, error notifications, and logging.
- Error actions include retry, rethrow, replay, abort, Java action, and human-intervention.

### 13.1.1. Fault Categories

There are two categories of faults:

- Business faults

  Business faults are generated when there is a problem with the information being processed. For example, a credit card number is invalid.

  Error actions for business faults that are internal to BPEL are configured in catch blocks. These are business faults that are thrown by a throw activity. Error notifications and logging for these business faults are handled by AIAAsyncErrorHandlingBPELProcess.

  Error actions for business faults from external applications and services are configured using the Composite Fault Policy Framework. These are business errors that are returned by an invoked service or application when using a BPEL invoke activity. Error notifications and logging for these business faults are handled by *oracle.apps.aia.core.eh.CompositeJavaAction.*

- System faults

  System faults occur as a result of problems within the running of the BPEL process or Mediator service component. For example, data cannot be copied properly because the variable name is incorrect or because of transformation errors.

  Error actions for system faults are configured using the Composite Fault Policy Framework. Error notifications and logging for system faults are handled by *oracle.apps.aia.core.eh.CompositeJavaAction.*

## 13.2. Introduction to Error Handling for Business Faults

This section discusses error handling for two types of business faults:

- Local business faults
- Remote business faults

#### Local Business Faults

If a BPEL process or Mediator component needs to issue a business error, such as a validation error, the process must be developed to issue the error explicitly, catch it in a catch block, and invoke the AIAAsyncErrorHandlingBPELProcess. The input to the process is a fault message in the AIA fault message schema. This is also true for business errors for Oracle Data Integrator, Oracle Service Bus, third-party B2B, and other external systems that want to leverage the AIA Error Handling and Logging framework.

### Remote Business Faults

If an invoked service or application responds to a request with a business fault, the Oracle SOA Suite captures these types of errors using the Composite Fault Policy Framework. The AIA Error Handling framework provides a custom Java action, *oracle.apps.aia.core.eh.CompositeJavaAction*, which can be configured as the Java action for all policies.

By configuring fault policies to include this Java action, the AIA Error Handling framework can perform all necessary error logging and notifications.

**For more information,** see *Oracle Application Integration Architecture Foundation Pack: Development Guide*, "Configuring Oracle AIA Processes for Error Handling and Trace Logging."

## 13.3. Introduction to Error Handling for BPEL and Mediator System Faults

These types of errors are captured using the Composite Fault Policy Framework. The AIA Error Handling framework provides a custom Java action, *oracle.apps.aia.core.eh.CompositeJavaAction*, which can be configured as the Java action for all policies.

By configuring fault policies to include this Java action, the AIA Error Handling framework can perform all necessary error logging and notifications.

**For more information,** see *Oracle Application Integration Architecture Foundation Pack: Development Guide*, "Configuring Oracle AIA Processes for Error Handling and Trace Logging."
**For more information** about the Composite Fault Policy framework, see *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*, "Using Fault Handling in a BPEL Process."

## 13.4. Introduction to Error Handling for Oracle B2B Errors

The Oracle AIA Error Handling Framework is automatically triggered when there is an error in Oracle B2B.

Oracle B2B can encounter errors while exchanging B2B documents with trading partners. Some common reasons for errors in the Oracle B2B layer include the following scenarios:

- Documents fail schema validation in the B2B layer.
- Incorrect or missing trading partner agreements in Oracle B2B.
- Incorrect or missing document-type definitions in Oracle B2B.
- Network errors or unavailability of a trading partner system.
- Authentication failures, for example invalid digital certificates, and so forth.

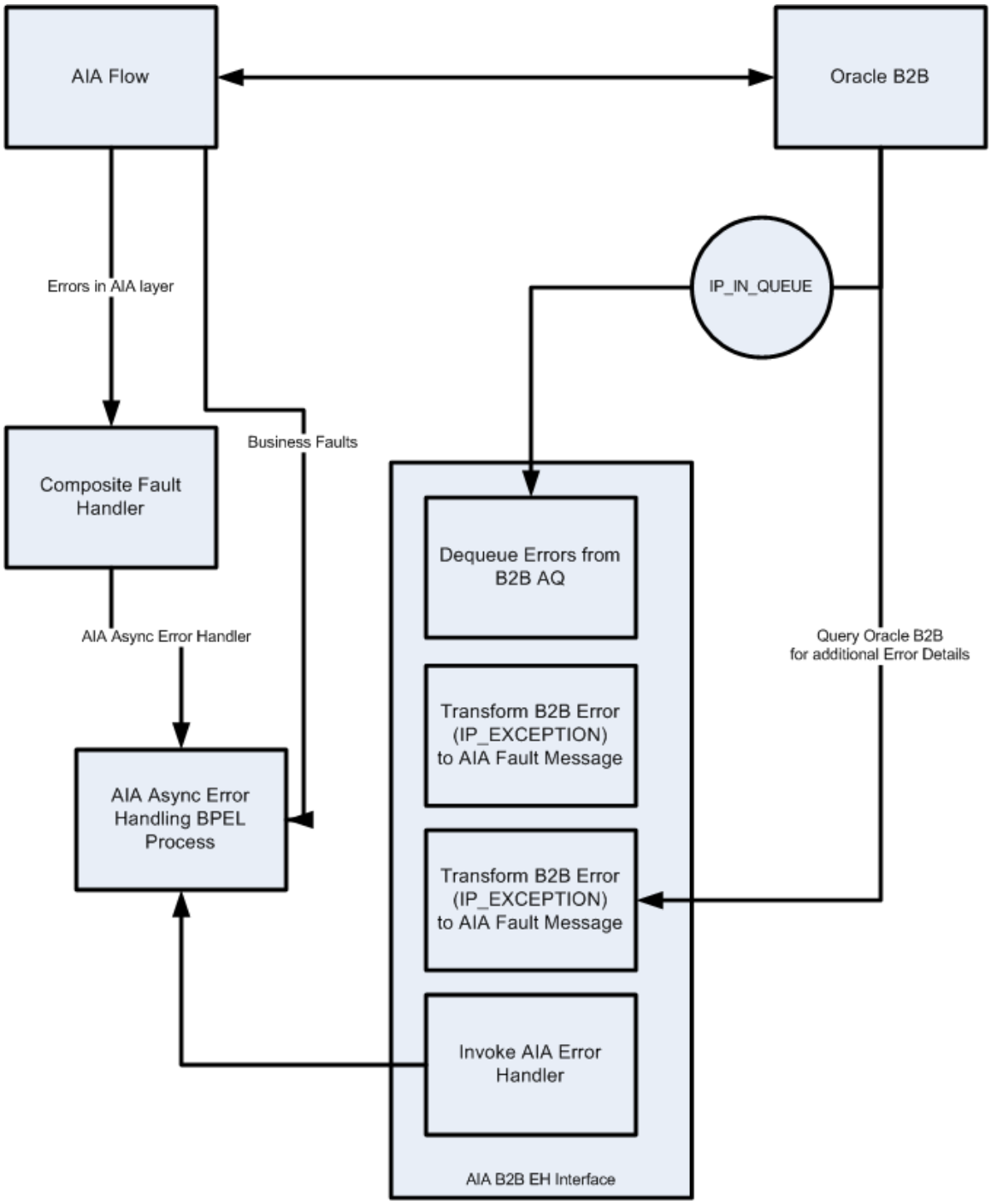
**Note.** Business process failures, such as an order being rejected by the trading partner if the ordered item is not in stock, are not considered to be Oracle B2B errors. Response or acknowledgement messages from trading partner applications containing these failures are treated as independent flows.

When Oracle B2B encounters these system errors, its default behavior is to publish the error to the Oracle Advanced Queuing (AQ) queue defined in the Oracle B2B infrastructure schema.

The details of the AQ to which Oracle B2B posts errors are as follows:

| Queue Name | IP_IN_QUEUE |
|---|---|
| Database Schema | SH_SOAINFRA |
| Queue Consumer | b2berroruser |
| Data Source | jdbc/SOADatasource |

The following diagram illustrates the way in which AIA's error handling framework captures B2B errors:




## Error handling framework support for capturing B2B errors

These errors can be viewed in error reports available in the Oracle B2B console.

**For more information** about viewing Oracle B2B error reports, see Accessing Oracle B2B Errors.

For Oracle B2B inbound and outbound flows, when an error occurs within the Oracle B2B server and not in the AIA layer, the AIA fault has the capacity to capture only the B2B-specific details.

Enterprise Business Message (EBM) details will not be available. However, in the case of an error in an outbound flow, AIA is able to track the EBMID and include that information in the fault.

# 14. Setting Up Error Handling

This chapter discusses the following topics:



## 14.1. Introduction to Setting Up Error Handling

Setting up error handling involves configuring the following items:

- Error notification enablement

  Error notification functionality is enabled by default.

  **For more information** about disabling error notification functionality, see Disabling Error Notifications.

- Oracle BPM Worklist enablement

  Oracle BPM Worklist functionality is disabled by default.

  **For more information** about enabling Oracle BPM Worklist functionality, see How to Enable the Oracle BPM Worklist.

- Error handling user roles

  Create user roles in WebLogic Server Administration Console to receive error notifications and Oracle BPM Worklist task assignments.

  **For more information,** see How to Create Error Handling User Roles.

- Error handling user role email addresses

  Use Oracle User Messaging Service to associate email addresses with error handling user roles. Error notifications will be sent to the email addresses specified.

  **For more information,** see How to Associate Email Addresses with Error Handling User Roles.

- Notification configuration details

  Configure details that enable error notification emails to be sent.

**For more information,** see How to Configure Notification Details.

- Error handling configuration details

  Define and modify error handling configuration details, including Error Notification and Oracle Worklist roles and responsibilities for processes operating in an Oracle Application Integration Architecture (AIA) ecosystem.

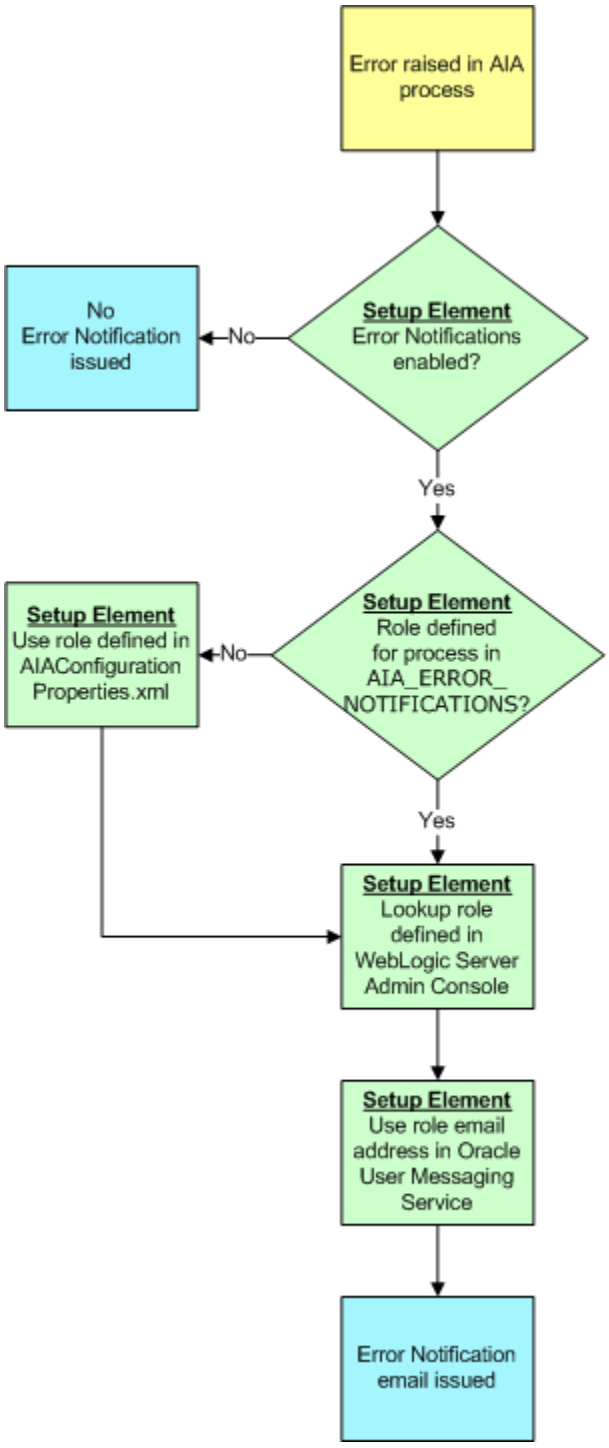
  **For more information,** see How to Set Up AIA Error Handling Configuration Details.

- Error handling responsibilities

  If you do not want to assign Actor and FYI user roles for specific error scenarios, you can assign default Actor and FYI user roles in AIAConfigurationProperties.xml.

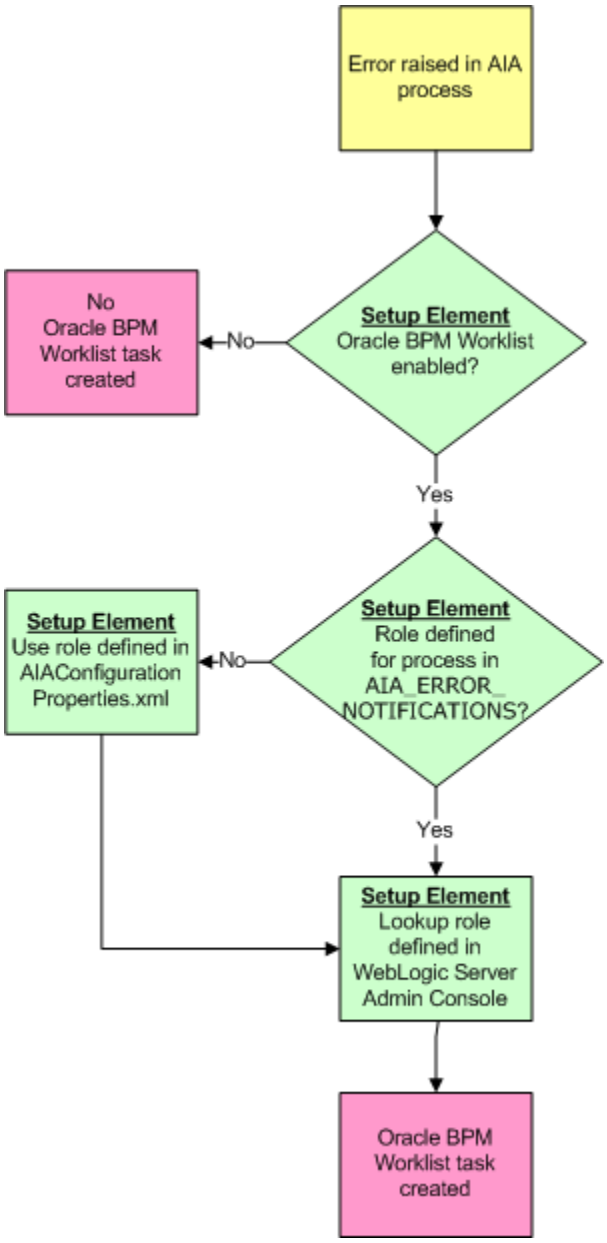
  **For more information,** see *Oracle Application Integration Architecture Foundation Pack: Development Guide*, "Configuring Oracle AIA Processes for Error Handling and Trace Logging."

The following diagram illustrates the way in which error handling setup elements enable Error Notification functionality.




## Error handling setup elements that enable Error Notification functionality

The following diagram illustrates the way in which error handling setup elements enable Oracle Worklist functionality.




Error handling setup elements that enable Oracle Worklist functionality

## 14.2. How to Create Error Handling User Roles

To create error handling user roles:

1. Access the Oracle WebLogic Server Administration Console: http://<host>:<port>/console.
2. In the Domain Structure menu, click Security Realms.
3. On the Summary of Security Realms page, select ***myrealm***.
4. On the Settings for myrealm page, select the Users and Groups tab.

5. Select the Users tab.
6. Create and modify user roles for use with the Error Handling Framework. For error handling notification and worklist functionality to work as designed, ensure that you are using user roles and not groups.

**For more information** about setting up user roles**,** see *Oracle WebLogic Server Documentation,* "Administration Console Online Help," Manage Users and Groups.

**Note.** Any user roles you create in the WebLogic Server Administration Console are stored in the Oracle WebLogic Server's embedded LDAP server. You may integrate a third-party LDAP solution to the embedded LDAP server.

**For more information** about Oracle WebLogic Server's embedded LDAP server, see *Oracle Fusion Middleware Understanding Security for Oracle WebLogic Server*, "Security Providers," Security Provider Databases.

## 14.3. How to Associate Email Addresses with Error Handling User Roles

To associate email address with error handling user roles:

1. Access the My Messaging Channels page in the Oracle User Messaging Service standalone user interface: http://<soa-host>:<soa-port>/sdpmessaging/userprefs-ui.

**For more information** about creating, updating, and deleting a message channel, see *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite,* "User Messaging Preferences," How to Manage Messaging Channels.

2. Associate an email address with an error handling user role.

**For more information** about creating user roles, see How to Create Error Handling User Roles.

3. Ensure that the messaging channel name you enter corresponds to an error handling user role name you have created according to information in How to Create Error Handling User Roles.

## 14.4. How to Configure Notification Details

To configuration notification details:

1. Set up workflow notification properties in the Oracle Enterprise Manager.

**For more information** about how to set up these properties, see *Oracle Fusion Middleware Administrator's Guide for Oracle SOA Suite,* "Configuring Human Workflow Service Components and Engines," Configuring Human Workflow Notification Properties .

2. Configure an email messaging channel. This enables the messaging service to resolve the email address when trying to send a notification to a user.

> **For more information** about how to configure an email messaging channel, see Oracle WebLogic Communication Services Developer's Guide.

3. Set the sender address for email notifications to a valid email address. Set this value in the FROM.EMAIL.ID property in the Error Handling Module section of the AIAConfigurationProperties.xml file. For example:

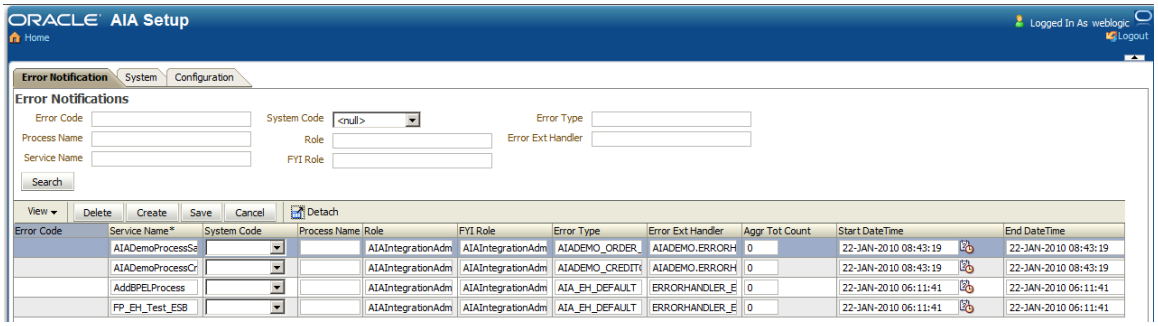
```
<Property name="FROM.EMAIL.ID">Email:AIA-Error-
Handling@oracle.com</Property>
```

> **For more information** about requirements for working with AIAConfigurationProperties.xml, see *Oracle Application Integration Architecture Foundation Pack: Development Guide*, "Building AIA Integration Flows," How to Set Up AIA Workstation.

## 14.5. How to Set Up AIA Error Handling Configuration Details

To set up AIA error handling configuration details:

1. Access Error Notifications page. To access the page, access the AIA Home Page. In the Setup area, click the Go button. Select the Error Notifications tab.




Error Notifications page

2. Use the page elements to define and modify error handling configuration details for processes operating in an Oracle AIA ecosystem, including Error Notification and Oracle Worklist roles and Error Notification throttling parameters.

The error handling configurations you define on the Error Notifications page are stored in the AIA_ERROR_NOTIFICATIONS table.

**Note.** For a given process, if no entry is found in the AIA_ERROR_NOTIFICATIONS table, the Actor and FYI roles specified in AIAConfigurationProperties.xml are used for Error Notifications and Oracle Worklist assignments, if enabled. By default, the Actor role is set to AIAIntegrationAdmin. Therefore, you are not required to populate the AIA_ERROR_NOTIFICATIONS table unless there is an explicit need.

**For more information,** see *Oracle Application Integration Architecture Foundation Pack: Development Guide*, "Configuring Oracle AIA Processes for Error Handling and Trace Logging."

Following are descriptions of key fields on the page:

| | |
|---|---|
| **ErrorCode** | For BPEL and Mediator process system error notifications, this is the fault code.<br><br>For business errors using catch blocks, this is the business error code you are catching. This is user-defined, for example, OUT_OF_INV. |
| **SystemCode** | This is the system code of the participating application. |
| **ProcessName** | This is the business process in which the service is participating. |
| **ServiceName** | For BPEL and Mediator services, this is the name of the service that experiences the error for which you are defining error notification details. For example, ***SampleBPELProcess.*** |
| **NotificationRole** | If you have enabled Error Notifications, specify the user role that you want to receive Actor error notifications for a process.<br><br>If you have enabled Oracle Worklist functionality, specify the role to which you want to assign Actor tasks for a process.<br><br>The Actor role is responsible for taking action to correct the error that generated the notification.<br><br>For Error Notifications or Oracle Worklist functionality, ensure that the role you specify here has a corresponding entry in the Oracle WebLogic Server Administration Console's user store.<br><br>**For more information**, see How to Create Error Handling User Roles.<br><br>For Error Notifications functionality, ensure that the user role has an email address defined in the Oracle WebLogic User Messaging Service.<br><br>**For more information,** see How to Associate Email Addresses with Error Handling User Roles. |
| **FyiNotificationRole** | If you have enabled Error Notifications, specify the user role that you want to receive FYI error notifications for a process. |

If you have enabled Oracle BPM Worklist functionality, specify the role to which you want to assign FYI tasks for a process.

This is the role that will be given information about the error, but will not be responsible for taking any actions to correct the error that generated the notification.

For Error Notifications or Oracle BPM Worklist functionality, ensure that the role you specify here has a corresponding entry in your implementation's user management store. By default, the AIA user management store is WebLogic's embedded LDAP server.

**For more information**, see How to Create Error Handling User Roles.

For Error Notifications functionality, ensure that the user role has an email address defined in Oracle User Messaging Service preferences.

**For more information,** see How to Associate Email Addresses with Error Handling User Roles.

**ErrorType**

The default value is AIA_EH_DEFAULT. Use this value if you want to use the AIA default error listener as the consuming component for this error notification.

Enter a unique value here if you are using extended error handling functionality.

**For more information** about extending error handling, see *Oracle Application Integration Architecture Foundation Pack: Development Guide,* "Configuring Oracle AIA Processes for Error Handling and Trace Logging."

If you want to use default and extended error handling functionality in a single error notification definition, add multiple Error Type values separated by commas. For example, ***AIA_EH_DEFAULT, ORDER_FO***, where ***AIA_EH_DEFAULT*** is the default Oracle AIA follow-through action, and ***ORDER_FO*** identifies the custom JMSCorrelationID for the extended error handling implementation. The listeners and associated actions for both of these error types will be executed at runtime.

**ErrorExtHandler (error extension handler)**

The default value is ***ERRORHANDLER_EXT***. Use this value for the error notification if you are not using an extended handler and the fault message will be generated based on the default fault message schema.

**For more information** about extending fault messages, see *Oracle Application Integration Architecture Foundation Pack: Development Guide,* "Configuring Oracle AIA

Processes for Error Handling and Trace Logging."

If you are using an extended handler to extend the fault message for this error notification, enter a unique value to identify the extension handler that will be used to enrich the fault message.

**AggrCountTot** (aggregation count total)

Error notification throttling must be enabled before this field value can be used to control the issuance of error notifications.

**For more information,** see How to Enable Error Notification Throttling.

Enter the total number of error notifications you want the system to suppress during a specific time interval for the given error scenario. The count is valid only during the specified time interval.

An error notification email is issued for the first error during the time interval. After reaching the count value, the count is reset to 0 and another error notification email is issued.

**StDatetime/EndDatetime**

Error notification throttling must be enabled before these field values can be used to control the issuance of error notifications.

**For more information,** see How to Enable Error Notification Throttling.

Enter the start and end date-and-time intervals to which you want the count value to apply.

For example, if you set the **AggrCountTot** field value to ***100***, the start date and time to ***30-Oct-2009 18:00:00***, and the end date and time to ***01-Nov-2009 17:00:00***, one error notification email will be sent out on the first occurrence of an error in the time interval. When the count value entered in the **AggrCountTot** field is reached, the count is reset to 0 and another error notification email is issued.

The date and time values used to track the time interval are derived from the database. The date and time displayed in the fields are derived from your browser time. Hover over the field values to view the database time.

## 14.5.1. What You Need to Know about Setting Up Error Handling Configurations

The Error Handling Framework uses runtime values and the data you enter on this page to execute the following hierarchical logic to determine the appropriate Error Notification and Oracle BPM Worklist assignment roles for an error.

1. If all four runtime values (SYSTEM CODE, ERROR CODE, SERVICE NAME, and PROCESS_NAME) are available and they map to an entry in this table, use the specified roles.
2. If the ERROR CODE, SERVICE NAME, and PROCESS NAME are available and map to an entry in this table, use the specified roles.
3. If the SERVICE_NAME and PROCESS_NAME are available and map to an entry in this table, use the specified roles.
4. If the SERVICE_NAME is available and maps to an entry in this table, use the specified roles.
5. If none of these values are available, the default values are fetched from the AIAConfigurationProperties.xml file.

# 15. Using Error Notifications

This chapter discusses the following topics:

- Introduction to Error Notifications
- Setting Up Error Notification Throttling
- Customizing Error Notification Emails
- Disabling Error Notifications

## 15.1. Introduction to Error Notifications

By default, error notification functionality is enabled. However, there are setup steps that must be completed.

**For more information** about setting up error notifications, see Setting Up Error Handling.

Error notification functionality generates and delivers email notifications to configured user roles.

**For more information** about configuring user roles for error notifications, see How to Create Error Handling User Roles and How to Associate Email Addresses with Error Handling User Roles.

You can define a user role to receive error notifications for a specific error scenario on the Error Notifications page.

**For more information,** see How to Set Up AIA Error Handling Configuration Details.

You can control the number of error notifications issued for an error scenario over a specific interval of time using error notification throttling functionality.

**For more information,** see Setting Up Error Notification Throttling.

Following sample text from an error notification email:

An error has occurred during the processing of AIA Integration Error in AIA SamplesCreateCustomerSiebelReqABCSImplProcess Process requires your attention. Please access the details from the url mentioned below.

---

==========

Please click on the following URL To view the instance details in the em console :

==========

http://sdc68063crm.us.oracle.com:7024/em/faces/ai/soa/messageFlow?target=/Farm_soa_domain/soa_domain/soa_server1/SamplesCreateCustomerSiebelReqABCSImpl+[1.0]&type=oracle_soa_composite&soaContext=default/SamplesCreateCustomerSiebelReqABCSImpl!1.0/140032

==========

==========

Please access the task in the Worklist Application :

==========

http://sdc68063crm.us.oracle.com:8024/integration/worklistapp/faces/home.jspx

==========

### Sample error notification email

As delivered, error notification emails contain a link to the Oracle Enterprise Manager Console, where recipients can view error information in the context of its flow trace. If the Oracle BPM Worklist is also enabled, error notification emails also contain a link to the Oracle BPM Worklist.

**For more information** about enabling the Oracle BPM Worklist, see How to Enable the Oracle BPM Worklist.

Error notification actor and FYI emails are generated based on content and configurations in AIAEHNotifications.xml file located in <AIA_HOME>/aia_instances/$INSTANCE_NAME/AIAMetaData/config. You can customize email content by editing this file.

**For more information** about customizing error notifications, see Customizing Error Notification Emails.

## 15.2. Setting Up Error Notification Throttling

This section provides an Introduction to Error Notifications and discusses:

- How to Enable Error Notification Throttling
- How to Configure Error Notification Throttling Parameters

### 15.2.1. Introduction to Error Notification Throttling

The Error Handling Framework is capable of sending out an error notification email each time an error scenario arises, however you may choose to use error notification throttling functionality to control the number of error notification emails sent during a specific time interval for a specific error scenario.

For example, if you know that a particular high-volume transaction will be down for an extended period, you can configure notification throttling settings to control the number of error notification emails that are sent for the error scenario during the transaction's down-time. This will help you avoid the onslaught of error notification emails that would have been triggered by the down-time had throttling configurations not been set.

## 15.2.2. How to Enable Error Notification Throttling

### Objective

Enable error notification throttling. By default, error notification throttling is disabled.

### Prerequisites

Ensure that error notification functionality is enabled. By default, error notifications are enabled. To verify that this functionality is enabled, access AIAConfigurationProperties.xml located in <AIA_HOME>/aia_instances/$INSTANCE_NAME/AIAMetaData/config. Ensure that the EH.INVOKE.NOTIFY property value is set to ***true.***

### Actor

Integration administrator

To enable error notification throttling:

1. Access AIAConfigurationProperties.xml in <AIA_HOME>/aia_instances/$INSTANCE_NAME/AIAMetaData/config.
2. Set Property Name = EH.AGGR.NOTIFY to ***true***.

   If error notification throttling functionality is disabled by setting this property value to ***false***, an error notification email is issued each time an error scenario arises.

3. Reload updates to the AIAConfigurationProperties.xml file.

**For more information** about reloading updates to AIAConfigurationProperties.xml**,** see *Oracle Application Integration Architecture Foundation Pack: Development Guide*, "Building AIA Integration Flows."

## 15.2.3. How to Configure Error Notification Throttling Parameters

### Objective

Configure the parameters by which you want error notification throttling to occur for an error scenario.

### Prerequisites

- Ensure that error notification functionality is enabled. By default, error notifications are enabled. To verify that this functionality is enabled, access AIAConfigurationProperties.xml located in <AIA_HOME>/aia_instances/$INSTANCE_NAME/AIAMetaData/config. Ensure that the EH.INVOKE.NOTIFY property value is set to ***true.***
- Ensure that error notification throttling is enabled.

**For more information,** see How to Enable Error Notification Throttling.

**Actor**

Integration administrator

To configure the parameters by which you want error notification throttling to occur:

1. Access the Error Notifications page. To access the page, access the AIA Home Page. In the Setup area, click the Go button. Select the Error Notifications tab.
2. For a given error scenario, defined by a set of **ErrorCode**, **SystemId**, **ProcessName**, and **ServiceName** values, enter **AggrCountTot**, **StDatetime**, and **EndDatetime** values.

    - In the **AggrCountTot** (aggregation count total) field, Enter the total number of error notifications you want the system to suppress during a specific time interval for the given error scenario. The count is valid only during the specified time interval.

      An error notification email is issued for the first error during the time interval. After reaching the count value, the count is reset to 0 and another error notification email is issued.

    - In the **StDatetime** and **EndDatetime** fields enter the start and end date-and-time intervals to which you want the total count value to apply.

      For example, if you set the **AggrCountTot** field value to ***100***, the start date and time to ***30-Oct-2009 18:00:00***, and the end date and time to ***01-Nov-2009 17:00:00***, one error notification email will be sent out on the first occurrence of an error in the time interval. When the count value entered in the **AggrCountTot** field is reached, the count is reset to 0 and another error notification email is issued.

      The date and time values used to track the time interval are derived from the database. The date and time displayed in the fields are derived from your browser time. Hover over the field values to view the database time.

**For more information** about the options on the Error Notifications page, see How to Set Up AIA Error Handling Configuration Details.

## 15.3. Customizing Error Notification Emails

This section provides an Introduction to Error Notification Customization and discusses:

- How to Customize the Subject Line of Error Notification Emails
- How to Customize the Body Text of Error Notification Emails
- How to Customize Additional URLs Provided in Error Notification Email Body Text

**Note.** These customizations will apply to all emails issued by error notification functionality.

## 15.3.1. Introduction to Error Notification Customization

You can customize the subject line and body text of emails issued by error notification functionality by editing the AIAEHNotifications.xml file located in <AIA_HOME>/aia_instances/$INSTANCE_NAME/AIAMetaData/config. The text of the file is shown here.

```
<?xml version="1.0" encoding="UTF-8"?>
<AIAEHNotification xmlns="http://schemas.oracle.com/aia/notify"
version="1.0">
     <EMAIL>
          <SUBJECT>Error in AIA #@#XPATH.{/default:Fault/default:
           FaultNotification/default:FaultingService/default:ID}#@#
           Process</SUBJECT>
          <BODY>An error has occurred during the processing of AIA
           Integration Error in AIA #@#XPATH.{/default:Fault/default
           :FaultNotification/default:FaultingService/default:ID}#@#
           Process requires your attention. Please access the
           details from the url mentioned below.</BODY>
     </EMAIL>
     <FYI_EMAIL>
          <SUBJECT>Error in AIA #@#XPATH.{/default:Fault/default:
           FaultNotification/default:FaultingService/default:ID}#@#
           Process FYI</SUBJECT>
          <BODY>An error has occurred during the processing of AIA
           Integration Error in AIA #@#XPATH.{/default:Fault/default
           :FaultNotification/default:FaultingService/default:ID}#@#
           Process requires your attention. Please access the
           details from the url mentioned below.</BODY>
     </FYI_EMAIL>
     <URL>
          ============================================================
          ========================
          Please click on the following URL To view the instance
          details in the em console :
          ============================================================
          ========================
          @ http://$adminHost:$adminPort/em/faces/ai/soa/message
          Flow?target=/Farm_$domainName/$domainName/$targetServer/
          #@#PROPS.{compositeName}#@#+[#@#PROPS.{composite Revision}
          #@#]%26type=oracle_soa_composite%26soaContext=#@#PROPS.
          {compositeDN}#
          @#/#@#PROPS.{compositeInstanceID}#@#
          ============================================================
          ========================
     </URL>
     <EXT_URL>
          ==================================================
          ==========
          Please access the task in the Worklist Application :
          ==================================================
          ==========
          @ http://$managedHost:$managedPort/integration/
          worklistapp/faces/home.jspx
          ==================================================
          ==========
```

```
        </EXT_URL>
    </AIAEHNotification>
```

All elements can be customized. All elements shown are required for error notifications to work as designed, even if you choose to leave some of them blank.

#### 15.3.1.1. EMAIL Element

You can customize the EMAIL element in AIAEHNotifications.xml to provide content that appears in error notification emails to Actor roles.

```
<EMAIL>
     <SUBJECT>Error in AIA #@#XPATH.{/default:Fault/default:
      FaultNotification/default:FaultingService/default:ID}#@#
      Process</SUBJECT>
     <BODY>An error has occurred during the processing of AIA
      Integration Error in AIA #@#XPATH.{/default:Fault/default
      :FaultNotification/default:FaultingService/default:ID}#@#
      Process requires your attention. Please access the
      details from the url mentioned below.</BODY>
</EMAIL>
```

The SUBJECT element provides the subject line of the error notification email. As delivered, the subject line is set to reference the ID of the service that experienced the error.

The BODY element provides the body text of the error notification email. As delivered, the body text is set to reference the ID of the service that experienced the error.

#### 15.3.1.2. FYI_EMAIL Element

You can customize the FYI_EMAIL element in AIAEHNotifications.xml to provide content that appears in error notification emails to FYI roles.

```
<FYI_EMAIL>
     <SUBJECT>Error in AIA #@#XPATH.{/default:Fault/default:
      FaultNotification/default:FaultingService/default:ID}#@#
      Process FYI</SUBJECT>
     <BODY>An error has occurred during the processing of AIA
      Integration Error in AIA #@#XPATH.{/default:Fault/default
      :FaultNotification/default:FaultingService/default:ID}#@#
      Process requires your attention. Please access the
      details from the url mentioned below.</BODY>
</FYI_EMAIL>
```

The SUBJECT element provides the subject line of the error notification email. As delivered, the subject line is set to reference the ID of the service that experienced the error.

The BODY element provides the body text of the error notification email. As delivered, the body text is set to reference the ID of the service that experienced the error.

#### 15.3.1.3. URL Element

As delivered, the URL element in AIAEHNotification.xml is used to provide a link to the composite instance flow trace details in the Oracle Enterprise Manager Console for your AIA implementation. You can customize this element to suit your implementation's needs.

$hostname, $adminport, and $domain tokens shown in the sample below are populated with implementation-specific values by the Oracle AIA Installer upon installation of Foundation Pack.

```
<URL>
     ============================================================
     ========================
     Please click on the following URL To view the instance
     details in the em console :
     ============================================================
     ========================
     @ http://$adminHost:$adminPort/em/faces/ai/soa/message
     Flow?target=/Farm_$domainName/$domainName/$targetServer/
     #@#PROPS.{compositeName}#@#+[#@#PROPS.{composite Revision}
     #@#]%26type=oracle_soa_composite%26soaContext=#@#PROPS.
     {compositeDN}#
     @#/#@#PROPS.{compositeInstanceID}#@#
     ============================================================
     ========================
</URL>
```

#### 15.3.1.4. EXT_URL Element

As delivered, the EXT_URL (external system URL) element in AIAEHNotifications.xml is used to provide a link to the Oracle BPM Worklist application, where, if enabled for AIA, the user can view their assigned AIA error-related tasks. You can customize this element to suit your implementation's needs.

$hostname and $port tokens shown in the sample below are populated with implementation-specific values by the Oracle AIA Installer upon installation of Foundation Pack.

```
<EXT_URL>
     ==================================================
     ==========
     Please access the task in the Worklist Application :
     ==================================================
     ==========
     @ http://$managedHost:$managedPort/integration/
     worklistapp/faces/home.jspx
     ==================================================
     ==========
</EXT_URL>
```

**For more information** about enabling Oracle BPM Worklist functionality, see How to Enable the Oracle BPM Worklist.

## 15.3.2. How to Customize the Subject Line of Error Notification Emails

### Objective

Customize the subject line of error notification emails.

### Prerequisites

Ensure that error notification functionality is enabled. By default, error notifications are enabled. To verify that this functionality is enabled, access AIAConfigurationProperties.xml located in <AIA_HOME>/aia_instances/$INSTANCE_NAME/AIAMetaData/config. Ensure that the EH.INVOKE.NOTIFY property value is set to ***true.***

### Actor

Integration administrator

## To customize the subject line of error notification emails:

1. Access the AIAEHNotifications.xml file located in <AIA_HOME>/aia_instances/$INSTANCE_NAME/AIAMetaData/config.
2. To customize the subject line used in error notification emails to Actor roles, edit the values in the <EMAIL><SUBJECT> element. To customize the subject line used in error notification emails to FYI roles, edit the values in the <FYI_EMAIL><SUBJECT> element.
3. To customize the AIA fault message schema value being displayed in the subject line, edit the XPATH value to use a different token. The token notation should use this format: ***#@#XPATH.{ACTUAL_XPATH_VALUE}#@#***. Error notification functionality will parse this file and replace the tokens with dynamic content. Enter as many or as few tokens as needed.

```
<?xml version="1.0" encoding="UTF-8"?>
<AIAEHNotification xmlns="http://schemas.oracle.com/aia/notify"
version="1.0">
     <EMAIL>
          <SUBJECT>Error in AIA #@#XPATH.{/default:Fault/default:
           FaultNotification/default:FaultingService/default:ID}#@#
           Process</SUBJECT>
          <BODY>An error has occurred during the processing of AIA
           Integration Error in AIA #@#XPATH.{/default:Fault/default
           :FaultNotification/default:FaultingService/default:ID}#@#
           Process requires your attention. Please access the
           details from the url mentioned below.</BODY>
     </EMAIL>
     <FYI_EMAIL>
          <SUBJECT>Error in AIA #@#XPATH.{/default:Fault/default:
           FaultNotification/default:FaultingService/default:ID}#@#
           Process FYI</SUBJECT>
          <BODY>An error has occurred during the processing of AIA
           Integration Error in AIA #@#XPATH.{/default:Fault/default
           :FaultNotification/default:FaultingService/default:ID}#@#
           Process requires your attention. Please access the
           details from the url mentioned below.</BODY>
     </FYI_EMAIL>
     <URL>
          ============================================================
          ========================
          Please click on the following URL To view the instance
          details in the em console :
          ============================================================
          ========================
          @ http://$adminHost:$adminPort/em/faces/ai/soa/message
```

```
            Flow?target=/Farm_$domainName/$domainName/$targetServer/
            #@#PROPS.{compositeName}#@#+[#@#PROPS.{composite Revision}
            #@#]%26type=oracle_soa_composite%26soaContext=#@#PROPS.
            {compositeDN}#
            @#/#@#PROPS.{compositeInstanceID}#@#
            ============================================================
            ========================
      </URL>
      <EXT_URL>
            ==================================================
            ==========
            Please access the task in the Worklist Application :
            ==================================================
            ==========
            @ http://$managedHost:$managedPort/integration/
            worklistapp/faces/home.jspx
            ==================================================
            ==========
      </EXT_URL>
</AIAEHNotification>
```

**For more information** about the AIA fault message schema, see *Oracle Application Integration Architecture Foundation Pack: Development Guide,* "Configuring Oracle AIA Processes for Error Handling and Trace Logging."

4. If you have implemented fault message schema extensions, you can customize the subject line to use these schema values as well.

**For more information** about extending the fault schema, see *Oracle Application Integration Architecture Foundation Pack: Development Guide,* "Configuring Oracle AIA Processes for Error Handling and Trace Logging."

5. Reload updates to the AIAEHNotifications.xml file.

**For more information** about reloading updates to AIAEHNotifications.xml**,** see *Oracle Application Integration Architecture Foundation Pack: Development Guide*, "Building AIA Integration Flows."

## 15.3.3. How to Customize the Body Text of Error Notification Emails

### Objective

Customize the body text of error notification emails.

### Prerequisites

Ensure that error notification functionality is enabled. By default, error notifications are enabled. To verify that this functionality is enabled, access AIAConfigurationProperties.xml located in <AIA_HOME>/aia_instances/$INSTANCE_NAME/AIAMetaData/config. Ensure that the EH.INVOKE.NOTIFY property value is set to ***true.***

### Actor

Integration administrator

**To customize the body text of error notification emails:**

1. Access the AIAEHNotifications.xml file located in <AIA_HOME>/aia_instances/$INSTANCE_NAME/AIAMetaData/config.
2. To customize the body text used in error notification emails to Actor roles, edit the values in the <EMAIL><BODY> element. To customize the body text used in error notification emails to FYI roles, edit the values in the <FYI_EMAIL><BODY> element.
3. To customize the AIA fault message schema values being displayed in the body text, edit the XPATH value to use a different token. The token notation should use this format: ***#@#XPATH.{ACTUAL_XPATH_VALUE}#@#***. Error notification functionality will parse this file and replace the tokens with dynamic content. Enter as many or as few tokens as needed.

```
<?xml version="1.0" encoding="UTF-8"?>
<AIAEHNotification xmlns="http://schemas.oracle.com/aia/notify"
version="1.0">
     <EMAIL>
          <SUBJECT>Error in AIA #@#XPATH.{/default:Fault/default:
           FaultNotification/default:FaultingService/default:ID}#@#
           Process</SUBJECT>
          <BODY>An error has occurred during the processing of AIA
           Integration Error in AIA #@#XPATH.{/default:Fault/default
           :FaultNotification/default:FaultingService/default:ID}#@#
           Process requires your attention. Please access the
           details from the url mentioned below.</BODY>
     </EMAIL>
     <FYI_EMAIL>
          <SUBJECT>Error in AIA #@#XPATH.{/default:Fault/default:
           FaultNotification/default:FaultingService/default:ID}#@#
           Process FYI</SUBJECT>
          <BODY>An error has occurred during the processing of AIA
           Integration Error in AIA #@#XPATH.{/default:Fault/default
           :FaultNotification/default:FaultingService/default:ID}#@#
           Process requires your attention. Please access the
           details from the url mentioned below.</BODY>
     </FYI_EMAIL>
     <URL>
          ============================================================
          ========================
          Please click on the following URL To view the instance
          details in the em console :
          ============================================================
          ========================
          @ http://$adminHost:$adminPort/em/faces/ai/soa/message
          Flow?target=/Farm_$domainName/$domainName/$targetServer/
          #@#PROPS.{compositeName}#@#+[#@#PROPS.{composite Revision}
          #@#]%26type=oracle_soa_composite%26soaContext=#@#PROPS.
          {compositeDN}#
          @#/#@#PROPS.{compositeInstanceID}#@#
          ============================================================
```

```
            ========================
        </URL>
        <EXT_URL>
            ==================================================
            ==========
            Please access the task in the Worklist Application :
            ==================================================
            ==========
            @ http://$managedHost:$managedPort/integration/
            worklistapp/faces/home.jspx
            ==================================================
            ==========
        </EXT_URL>
    </AIAEHNotification>
```

**For more information** about the AIA fault message schema, see *Oracle Application Integration Architecture Foundation Pack: Development Guide,* "Configuring Oracle AIA Processes for Error Handling and Trace Logging."

4. If you have implemented fault message schema extensions, you can customize the body text to use these schema values as well.

**For more information** about extending the fault schema, see *Oracle Application Integration Architecture Foundation Pack: Development Guide,* "Configuring Oracle AIA Processes for Error Handling and Trace Logging."

5. Reload updates to the AIAEHNotifications.xml file.

**For more information** about reloading updates to AIAEHNotifications.xml**,** see *Oracle Application Integration Architecture Foundation Pack: Development Guide*, "Building AIA Integration Flows."

## 15.3.4. How to Customize Additional URLs Provided in Error Notification Email Body Text

### Objective

Customize additional URLs provided in error notification email body text.

### Prerequisites

- Ensure that error notification functionality is enabled. By default, error notifications are enabled. To verify that this functionality is enabled, access AIAConfigurationProperties.xml located in <AIA_HOME>/aia_instances/$INSTANCE_NAME/AIAMetaData/config. Ensure that the EH.INVOKE.NOTIFY property value is set to ***true.***
- As delivered, error notification email body text includes a link to the Oracle BPM Worklist. To enable users to access AIA-related error tasks in the Oracle BPM Worklist, ensure that Oracle BPM Worklist functionality is enabled.

**For more information** about enabling Oracle BPM Worklist to work with AIA error handling, see How to Enable the Oracle BPM Worklist.

### Actor

Integration administrator

To customize application links in body text of error notification emails:

1. Access the AIAEHNotifications.xml file located in <AIA_HOME>/aia_instances/$INSTANCE_NAME/AIAMetaData/config.
2. To customize the URLs provided in the error notification email body text, edit the values in the <URL> and <EXT_URL> elements.
3. As delivered, the <URL> element provides a link to flow trace details for the composite instance in the Oracle Enterprise Manager Console for your AIA implementation. The flow trace provides details about all of the services, references, and components across composites that are participating in the flow.

**For more information** about viewing flow trace details in Oracle Enterprise Manager, see *Oracle Fusion Middleware Administrator's Guide for Oracle SOA Suite*, "Monitoring BPEL Process Service Components and Engines," Viewing the Audit Trail and Process Flow of a BPEL Process Service Component.

An error has occurred during the processing of AIA Integration Error in AIA SamplesCreateCustomerSiebelReqABCSImplProcess Process requires your attention. Please access the details from the url mentioned below.

---

==========

Please click on the following URL To view the instance details in the em console :

==========

http://sdc68063crm.us.oracle.com:7024/em/faces/ai/soa/messageFlow?target=/Farm_soa_domain/soa_domain/soa_server1/SamplesCreateCustomerSiebelReqABCSImpl+[1.0]&type=oracle_soa_composite&soaContext=default/SamplesCreateCustomerSiebelReqABCSImpl!1.0/140032

==========

==========

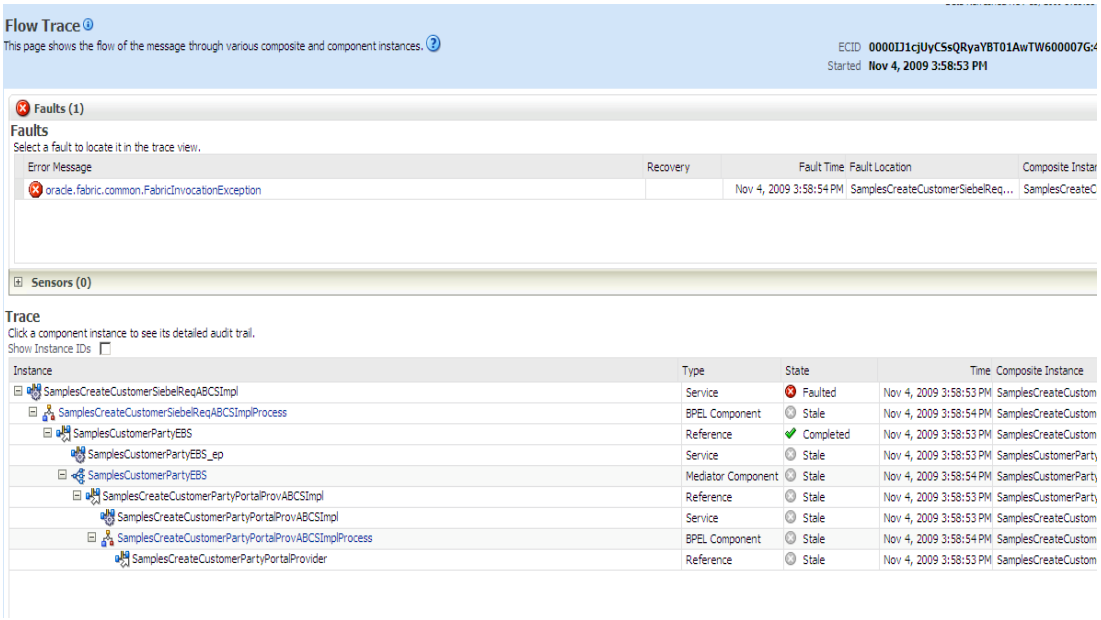
Please access the task in the Worklist Application :

==========

http://sdc68063crm.us.oracle.com:8024/integration/worklistapp/faces/home.jspx

==========

Sample Error Notification email text providing a link to flow trace details in the Oracle Enterprise Manager Console




### Fault details on the Enterprise Manager Console Flow Trace page

We deliver the following parameters that enable this drill-down into the Oracle Enterprise Manager Console:

- #@#PROPS.{compositeName}#@#
- #@#PROPS.{compositeRevision}#@#
- #@#PROPS.{compositeDN}#@#
- #@#PROPS.{compositeInstanceID}#@#

For system errors configured in fault policy files, these parameters will be automatically derived to build the URL for inclusion in the error notification email. Specifically, by default, remote and binding faults are configured in the fault policy file.

For business errors, you must configure impacted processes to populate the fault message with the execution context ID (ECID). Error notification functionality will derive these parameters to build the URL based on this ECID value.

**For more information** about programming guidelines to populate fault messages with ECID value, see *Oracle Application Integration Architecture Foundation Pack: Development Guide,* "Configuring Oracle AIA Processes for Error Handling and Trace Logging."

4. As delivered, the <EXT_URL> element provides a link to the Oracle BPM Worklist.

   If you are not using Oracle BPM Worklist as a part of your AIA implementation and do not want error notification emails to include this link to Oracle BPM Worklist, access AIAConfigurationProperties.xml located in <AIA_HOME>/aia_instances/$INSTANCE_NAME/AIAMetaData/config and set the EH.INVOKE.HWF property to ***false***. This setting will remove any content expressed in the <EXT_URL> element, including the Oracle BPM Worklist default link, from error notification emails.

```
<?xml version="1.0" encoding="UTF-8"?>
<AIAEHNotification xmlns="http://schemas.oracle.com/aia/notify"
version="1.0">
     <EMAIL>
          <SUBJECT>Error in AIA #@#XPATH.{/default:Fault/default:
           FaultNotification/default:FaultingService/default:ID}#@#
           Process</SUBJECT>
          <BODY>An error has occurred during the processing of AIA
           Integration Error in AIA #@#XPATH.{/default:Fault/default
           :FaultNotification/default:FaultingService/default:ID}#@#
           Process requires your attention. Please access the
           details from the url mentioned below.</BODY>
     </EMAIL>
     <FYI_EMAIL>
          <SUBJECT>Error in AIA #@#XPATH.{/default:Fault/default:
           FaultNotification/default:FaultingService/default:ID}#@#
           Process FYI</SUBJECT>
          <BODY>An error has occurred during the processing of AIA
           Integration Error in AIA #@#XPATH.{/default:Fault/default
           :FaultNotification/default:FaultingService/default:ID}#@#
           Process requires your attention. Please access the
           details from the url mentioned below.</BODY>
     </FYI_EMAIL>
     <URL>
          ============================================================
          ========================
          Please click on the following URL To view the instance
          details in the em console :
          ============================================================
          ========================
          @ http://$adminHost:$adminPort/em/faces/ai/soa/message
          Flow?target=/Farm_$domainName/$domainName/$targetServer/
          #@#PROPS.{compositeName}#@#+[#@#PROPS.{composite Revision}
          #@#]%26type=oracle_soa_composite%26soaContext=#@#PROPS.
          {compositeDN}#
          @#/#@#PROPS.{compositeInstanceID}#@#
          ============================================================
          ========================
     </URL>
     <EXT_URL>
          ==================================================
          ==========
          Please access the task in the Worklist Application :
          ==================================================
          ==========
          @ http://$managedHost:$managedPort/integration/
          worklistapp/faces/home.jspx
          ==================================================
          ==========
     </EXT_URL>
</AIAEHNotification>
```

5. Reload updates to the AIAEHNotifications.xml file.

**For more information** about reloading updates to AIAEHNotifications.xml**,** see *Oracle Application Integration Architecture Foundation Pack: Development Guide*, "Building AIA Integration Flows."

## 15.4. Disabling Error Notifications

By default, error notification functionality is enabled. You can disable this functionality in AIAConfigurationProperties.xml.

To disable error notifications:

1. Access AIAConfigurationProperties.xml located in <AIA_HOME>/aia_instances/$INSTANCE_NAME/AIAMetaData/config.
2. Set the EH.INVOKE.NOTIFY property value to ***false.***
3. Reload updates to AIAConfigurationProperties.xml.

**For more information** about reloading updates to AIAConfigurationProperties.xml**,** see *Oracle Application Integration Architecture Foundation Pack: Development Guide*, "Building AIA Integration Flows."

When error notification functionality is disabled, the Error Handling Framework does not issue error notification emails, but continues to log errors and assemble fault messages in the AIA Error Topic.

While error notifications are disabled, the AIA fault message remains available for input in the AIA Error Topic. This enables the Error Handling Framework to support a fully customized error handling solution.

# 16. Using the Oracle BPM Worklist

This chapter discusses the following topics:



## 16.1. Introduction to the Oracle BPM Worklist

The Oracle BPM Worklist application can be used to provide an error console for the Oracle Application Integration Architecture (AIA). You can enable this functionality in AIAConfigurationProperties.xml.

**Note.** Oracle BPM Worklist functionality will be accessible even if you have not enabled it to work with Oracle AIA. However, the Oracle BPM Worklist will not include any AIA-specific error tasks.

**For more information** about enabling Oracle BPM Worklist functionality, see How to Enable the Oracle BPM Worklist.

The Oracle BPM Worklist application is a user interface (UI) that Actor roles, such as integration administrators, and FYI roles, such as customer service representatives (CSRs), can use to access details about AIA ecosystem service errors that have been assigned to them for resolution or for informational purposes only. Users will not receive email notifications regarding Oracle BPM Worklist task assignments unless error notifications are enabled.

**For more information**, see Using Error Notifications.

Based on their roles, users will be able to interact with the following types of tasks in the Oracle BPM Worklist:

- Single-approver task

  Actor roles, such as integration administrators, are assigned single-approver tasks in the Oracle BPM Worklist. Typically, this role is responsible for taking action to resolve the error and must update the error task with activity and status details. Therefore, for Actor roles, the Oracle BPM Worklist provides an editable UI.

- FYI task

  FYI roles, such as customer service representatives, are assigned FYI tasks in the Oracle BPM Worklist. Typically, this role only needs a view of information about the status of the errored end-to-end transaction. Therefore, for FYI roles, the Oracle BPM Worklist provides a display-only UI. The FYI role is not responsible for taking any particular action to resolve the error.

The Oracle BPM Worklist provides the following error details that can assist in the troubleshooting process:

- EBMID
- EBMName
- EBOName
- Verb Code
- Business Scope Reference ID
- Business Scope Reference Instance ID
- Enterprise Service Name
- Enterprise Service Operation Name
- Sender Reference ID
- Sender Message ID
- Sender Reference Transaction Code
- Sender Object Identification ID
- Context ID
- EBOID
- Reporting Date Time
- Corrective Action
- Fault Message Code
- Fault Message Text
- Severity
- Stack
- Faulting Service ID
- Faulting Service Implementation Code
- Faulting Service Instance ID
- B2B Fault Element
- B2BMReference/B2BMID
- B2BMReference/B2BDocumentType/DocumentTypeCode
- B2BMReference/B2BDocumentType/DocumentTypeVersion
- B2BMReference/SenderTradingPartner/TradingPartnerID
- B2BMReference/ReceiverTradingPartner/TradingPartnerID

## 16.2. How to Enable the Oracle BPM Worklist

By default, Oracle BPM Worklist functionality is disabled. You can enable this functionality in AIAConfigurationProperties.xml.

To enable the Oracle BPM Worklist:

1. Access AIAConfigurationProperties.xml located in <AIA_HOME>/aia_instances/$INSTANCE_NAME/AIAMetaData/config.
2. Set the EH.INVOKE.HWF property value to ***true.***
3. Reload updates to the AIAConfigurationProperties.xml file.

**For more information** about reloading updates to AIAConfigurationProperties.xml, see *Oracle Application Integration Architecture Foundation Pack: Development Guide*, "Building AIA Integration Flows."

The AIAReadJMSNotification BPEL process will now listen to the AIA Error Topic Java message service (JMS) topic, which is populated by the Error Handling Framework. Relevant errors will be aggregated by the AIAReadJMSNotification BPEL process and displayed in the Oracle BPM Worklist.

If error notification is also enabled, error notification emails will contain a link to the Oracle BPM Worklist.

**For more information** about error notifications, see Using Error Notifications.

## 16.3. How to Use the Oracle BPM Worklist

Once you have been assigned an AIA error task that you need to view or act upon to resolve, you can use the details provided by the Oracle BPM Worklist to troubleshoot the error.

Access the Oracle BPM Worklist: http://<host>:<SOA server port>/integration/worklistapp.

Your assigned tasks display on the My Tasks page. You can filter your assigned tasks using various criteria and search for assigned tasks by title, priority, and status. Click an assigned task to access complete task details.

**For more information**, see *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*, "Using Oracle BPM Worklist."

FYI user roles can view a task in read-only mode in the Oracle BPM Worklist.

Actor user roles can work on a task by acquiring the task. They can also enter comments against the task and update the task status. For example, when the error has been resolved, the user can set the task action to *COMPLETED.* Setting this value in the Actions field completes the task.

**For more information** about message resubmission, see Using the Message Resubmission Utility.

# 17. Using the Message Resubmission Utility

This chapter discusses the following topics:

- Introduction to the Message Resubmission Utility
- How to Use the Message Resubmission Utility.

## 17.1. Introduction to the Message Resubmission Utility

To use the Message Resubmission Utility, you must implement error handling and recovery for the asynchronous message exchange pattern.

**For more information**, see *Oracle Application Integration Architecture Foundation Pack: Development Guide,* "Configuring Oracle AIA Processes for Error Handling and Trace Logging."

According to this implementation method, when a message cannot be delivered to a service or component in the flow of a global transaction, the message is rolled back to the appropriate source milestone. This source milestone corresponds to an Oracle Advanced Queue or JMS topic. It is here that the message will be persisted until it can be resubmitted for delivery to the service or component.

At the same time, a fault is raised by the Error Handling framework and, if enabled, error notifications and Oracle BPM Worklist tasks regarding the fault are created to alert administrators.

**For more information** about the Oracle BPM Worklist, see Using the Oracle BPM Worklist.
**For more information** about error notifications, see Using Error Notifications.

Once notified, the most natural course of action is for the administrator to bring up the failed service or component. Once the service or component is back up and running, the administrator can use the Message Resubmission Utility to recover the faulted message from the source milestone. The Message Resubmission Utility changes the state of the faulted message to the ***Ready*** state, enabling it to be picked up by the consumer process.

## 17.2. How to Use the Message Resubmission Utility

This section discusses how to use the Message Resubmission Utility to resubmit a faulted message.

For message resubmission scenarios that involve Oracle Advanced Queue, we provide the MSG_RESUBMIT stored procedure. This procedure assumes that the message type is ***SYS.AQ$_JMS_MESSAGE***.

If the message type is JMS, the following must be set in the WebLogic Console for the JMS resource (Queue or Topic) in the Delivery Failure section:

- Redelivery Limit: Set to specify the number of retries before the message is moved to the error resource.
- Expiration Policy: Set to ***Redirect***.
- Error Destination: Set to a valid JMS resource.

In addition, there must be a JMS Connection Factory with the JNDI Name ***<JMS Resource JNDI Name>CF***. For example, the JNDI Name of the Error Destination Resource plus ***CF***.

If the message type being used is not ***SYS.AQ$_JMS_MESSAGE,*** you must change the data type for the ***MSG*** variable in the MSG_RESUBMIT stored procedure and then recompile the procedure. You can then use the Message Resubmission Utility for resubmission based on message ID.

**For more information** about configuring a queue with AQ to support resubmission, see *Oracle Fusion Middleware Configuring and Managing JMS for Oracle WebLogic Server*, "Interoperating with Oracle AQ JMS," Configure AQ JMS Foreign Server Destinations.

## To use the Message Resubmission Utility:

1. Access the Oracle AIA log file, <DOMAIN_HOME>/servers/<SOA Server Name>/logs/aia-error.log to look up the following values included in the IntermediateMessageHop element for the message that requires resubmission:
   - SenderResourceTypeCode
   - SenderResourceID
   - SenderMessageID

   **For more information** about these values in the context of the Oracle AIA fault message schema, see *Oracle Application Integration Architecture Foundation Pack: Development Guide,* "Configuring Oracle AIA Processes for Error Handling and Trace Logging."

   Alternatively, you can also look up the aia-error.log in the Oracle Enterprise Manager. Under WebLogic Domain, <*domain name*>, right-click the manage server entry (usually ***soa_server1***). Navigate to Logs, View Log Messages. On the Log Message page, provide search criteria (optional) and click the Search button.

   **For more information** about viewing the Oracle AIA log in Oracle EM, see Using Trace and Error Logs.

2. Set these values in the ResubmissionParams.properties file located in $AIA_HOME/util/AIAMessageResubmissionUtil.

3. For Windows, execute $AIA_INSTANCE\bin\aiaenv.bat. For Linux, source $AIA_INSTANCE/bin/aiaenv.sh.

4. Navigate to $AIA_HOME/util/ AIAMessageResubmissionUtil and execute the following ant command:

   ```
   ant –buildfile MessageResubmit.xml
   ```

   The MessageResubmit.xml script references the edited ResubmissionParams.properties file. Once run, the script resets the message status back to a ready state so that the transaction can resume its flow.

Following is the ResubmissionParams.properties file for AIA JMS sample:

```
@ jms.app.hostName=sdc60024sems.us.oracle.com
jms.app.admin.port=7097
jms.app.soa.port=8097
jms.app.userName=weblogic
@ jms.app.password=weblogic#1
jms.aq=false
jms.moduleName=AIAJMSModule
#QUEUE/TOPIC - 1/2
resourceType=1
#queueName/topicName/routingServiceName
resourceName=AIA_SiebelCustomerJMSQueue
#messageID/groupID
messageID=ID:<983029.1264581138423.0>
#queueTableName/topicTableName
aq.resourceTableName=AIASamples
@ aq.db.driverName=oracle.jdbc.driver.OracleDriver
aq.db.jdbcURL=jdbc:oracle:thin:@localhost:1521:XE
aq.db.userName=aia
@ aq.db.password=aia
```

# 18. Using Trace and Error Logs

This chapter discusses the following topics:



## 18.1. Introduction to Trace and Error Logging

The Oracle Application Integration Architecture (AIA) enables you to generate trace and error log files that provide a detailed view of services running in your AIA ecosystem. These logs can be especially informative when troubleshooting service processing issues.

- Trace

  Trace logs capture chronological recordings of a service's general activities. The trace log is created by configuring the service to make an explicit call using the trace logging custom XPath or Java API.

  **For more information,** see *Oracle Application Integration Architecture Foundation Pack: Development Guide,* "Configuring Oracle AIA Processes for Error Handling and Trace Logging."

- Error

  Error logs capture a recording of errors that occur during a service's activities. No specific configurations are required to make BPEL and Mediator services eligible for error logging. The Error Handling Framework is designed to trigger an error logging event for errors occurring in any of the Oracle AIA services, whether they are BPEL- or Mediator-based. The Error Handling Framework does this logging non-intrusively.

## 18.2. How to Enable Trace Logging

Trace logging is enabled via configurations in the AIAConfigurationProperties.xml file located in <AIA_HOME>/aia_instances/$INSTANCE_NAME/AIAMetaData/config.

Logging can be set at the system or service level. The logging property set at the service level overrides the property set at the system level.

To enable trace logging for the entire system:

1. Access the AIAConfigurationProperties.xml file.
2. Set the TRACE.LOG.ENABLED property at the system level to *TRUE*.

To enable trace logging for an individual service:

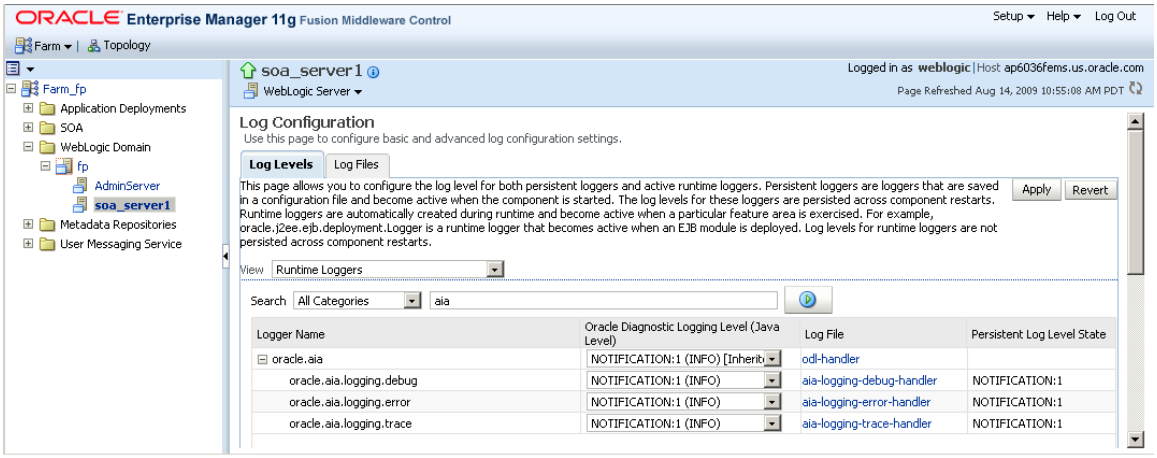
1. Access the AIAConfigurationProperties.xml file.
2. Set the TRACE.LOG.ENABLED property for the service to *TRUE*.
3. Reload updates to AIAConfigurationProperties.xml.

**For more information** about reloading updates to AIAConfigurationProperties.xml, see *Oracle Application Integration Architecture Foundation Pack: Development Guide,* "Building AIA Integration Flows."

## 18.3. How to Set Trace Log Levels

To set trace log levels:

1. Access the Oracle Enterprise Manager console (http://<host>:<port>/em).
2. Expand the **WebLogic** domain and navigate to your domain. Right-click on your domain and select Logs, Log Configuration.
3. Select the **Log Levels** tab on the Log Configuration page.




Log Configuration page

4. In the **View** drop-down list box, select ***Runtime Loggers***.
5. In the **Search** drop-down list box, select ***All Categories***. Enter ***aia*** in the **Search** field and execute the search.
6. Locate **Logger Name** value ***oracle.aia -> oracle.aia.logging.trace*** and set the **Oracle Diagnostic Logging Level (Java Level)** field value accordingly. The type and amount of information written to trace log files is determined by the message type and log level specified.
7. Select from one of the following values, ordered from highest to lowest severity. The lower the severity level, the more information is written to the log file.

| | |
|---|---|
| ***INCIDENT_ERROR:1 (SEVERE+100)*** | A serious problem, such as one from which you cannot recover. The problem may be caused by a bug in the product and should be reported to Oracle Support. |
| ***ERROR:1 (SEVERE)*** | A serious problem that requires immediate attention from the administrator and is not caused by a bug in the product. |
| ***WARNING:1 (WARNING)*** | A potential problem, such as invalid parameter values or a specified file that does not exist, that should be reviewed by the administrator. |
| ***NOTIFICATION:1 (INFO)*** | A major lifecycle event such as the activation or deactivation of a primary subcomponent or feature. |
| ***NOTIFICATION:16 (CONFIG)*** | A finer level of granularity for reporting normal events. |
| ***TRACE:1 (FINE)*** | Trace or debug information for events that are meaningful to end-users of the product, such as public API entry or exit points. |
| ***TRACE:16 (FINER)*** | Detailed trace or debug information that can help Oracle Support diagnose problems with a particular subsystem. |
| ***TRACE:32 (FINEST)*** | Very detailed trace or debug information that can help Oracle Support diagnose problems with a particular subsystem. |

# 18.4. How to Access Trace and Error Logs

In this section, we discuss:

- Accessing Oracle AIA Logs in the Oracle Enterprise Manager Console
- Searching for Oracle AIA Log Messages
- Accessing Oracle AIA Log XML Files

## 18.4.1. Accessing Oracle AIA Logs in the Oracle Enterprise Manager Console

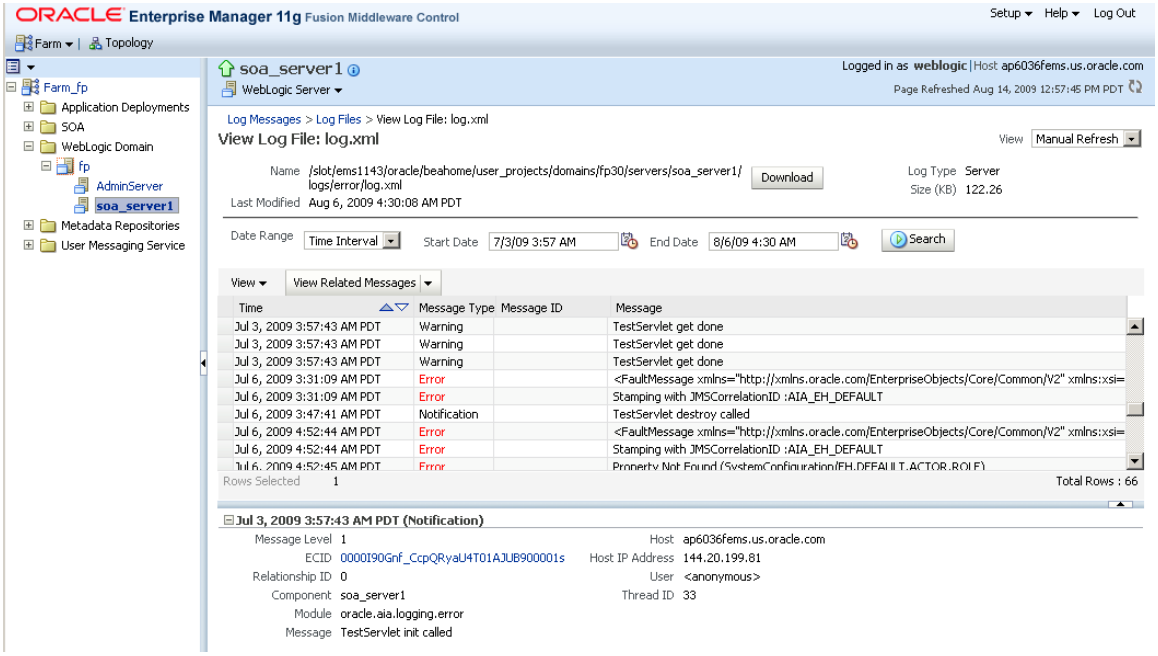
Log files can be accessed using the Oracle Enterprise Manager user interface, in much the same way that standard log files generated by various components of the Oracle SOA Suite can be handled in Oracle Enterprise Manager. Using Oracle Enterprise Manager as the user interface for the logs enables searching, sorting, and filtering of logs.

To access Oracle AIA trace and error log files:

1. Access the Oracle Enterprise Manager console (http://<host>:<port>/em).
2. Expand the **WebLogic** domain and navigate to your domain. Right-click on your domain and select Logs, View Log Messages.

3. Click the **Target Log Files** button. The error log file, aia-error.log, can be found under ${domain.home}/servers/${weblogic.Name}/logs. The trace log file, aia-trace.log, can be found under ${domain.home}/servers/${weblogic.Name}/logs.

4. To view a log file, select the file row and click the **View Log File** button.




View Log File page

5. To download a log file, select the file row and click the **Download** button.

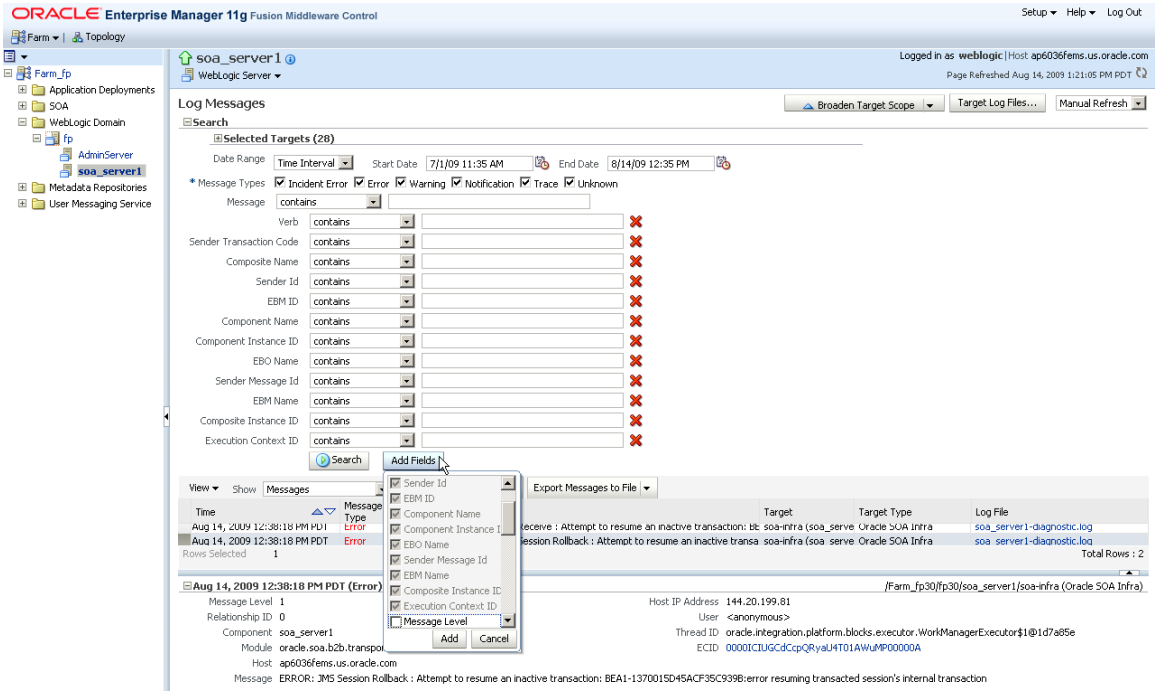
## 18.4.2. Searching for Oracle AIA Log Messages

To search for Oracle AIA trace and error log messages:

1. Access the Oracle Enterprise Manager console (http://<host>:<port>/em).

2. Expand the **WebLogic** domain and navigate to your domain. Right-click on your domain and select Logs, View Log Messages.

3. Search for specific log messages using the search parameters available in the **Search** area on the Log Messages page.




Log Messages page

## 18.4.3. Accessing Oracle AIA Log XML Files

You can access Oracle AIA trace and error log XML files directly in the following directories:

- ${domain.home}/servers/${weblogic.Name}/logs/aia-error.log
- ${domain.home}/servers/${weblogic.Name}/logs/aia-trace.log

# 19. Accessing Oracle B2B Errors

To access error reports about failed business-to-business (B2B) transactions in the Oracle B2B console:

1. Access the Oracle B2B console: http://<soa-host>:<port>/b2b.
2. Log in using the B2B administrator user name and password.
3. Select the Reports tab.
4. Select the Errors tab.
5. Enter search criteria or leave fields blank. Click Search. Once you have located the failed message, you can use the B2B error monitor to:
   - View details about the error
   - View the payload of the failed message
   - Retry processing of the failed message

**Note.** Business process failures, such as an order being rejected by the trading partner if the ordered item is not in stock, are not considered to be Oracle B2B errors. Response or acknowledgement messages containing these failures are treated as independent flows.

**For more information** about Oracle B2B errors in Oracle Application Integration Architecture (AIA), see Introduction to Error Handling for Oracle B2B Errors and *Oracle Application Integration Architecture Development Guide,* "Understanding B2B Integration Using AIA."

# Part: Working with Oracle AIA Developer Tools

# 20. Introduction to AIA Developer Tools

This chapter provides an Overview of AIA Developer Tools

## 20.1. Overview of AIA Developer Tools

Oracle Application Integration Architecture (AIA) Developer Tools equip Process Integration Pack (PIP) developers, architects and quality assurance engineers with tools to help them effectively and efficiently develop PIPs that are in compliance with open standards and AIA best practices in design and coding.

Governance, whether it is financial, business, legal, or IT, is about getting people to do the right thing at the right time. It is about encouraging behaviors that will achieve your business goals. The foundation of a service-oriented architecture (SOA) implementation is good SOA governance. AIA Developer Tools assist in establishing SOA governance as a service so that your organization can reap all of the benefits of solid SOA governance process.

AIA Developer Tools expose mapping information that exist in cryptic XSLT files; enable PIP validations per The Open Group Architecture Framework (TOGAF) standards, and provide a PIP artifact reuse matrix. These tools are used by Oracle PIP teams, partners, and customers alike.

For PIP-specific reports, AIA Developer Tools need to be able to identify the artifact inventory for each PIP, including all of its services, cross references, domain value maps, and any other related artifacts. All AIA Developer Tools covered in this part expect the same input XML file, GenerateScriptInput.xml file, also known as the inputMetaFile. To produce any PIP-specific reports, ensure that you have created a GenerateScriptInput.xml file.

**For more information** about the GenerateScriptInput.xml file, see Appendix: Understanding GenerateScriptInput.xml.

# 21. Using the XSL Mapping Analyzer

This chapter discusses the following topics:

- Overview of XMAN
- Generating XMAN Reports
- Adding XMAN Annotations to XSLT Files

## 21.1. Overview of XMAN

The reuse of artifacts and effective information sharing are key principles of SOA governance. The XSL Mapping Analyzer (XMAN) analyzes mapping information that exists in cryptic Application Business Connector Service (ABCS) XSLT files and provides it in a more readable format so that existing connector mappings can be easily considered for reuse.

Being able to comprehend the mappings between an Application Business Message (ABM) and an Enterprise Business Message (EBM) becomes imperative when developing a connector based on existing connectors.

True interoperability can be only be achieved by adopting a consistent mapping practice using the appropriate domain value map (DVM), cross-reference (XREF), and translation functions; which this tool encourages.

When preparing for an upgrade, use XMAN to compare customized mappings to Oracle-supplied mappings. Evaluate the results of these comparisons and make any necessary changes before performing the upgrade.

Following is a sample XMAN mapping report.

**XMAN Mapping Report**

| | | |
|---|---|---|
| **File name :** | CustomerEBMList_To_ContactABMList.xsl | Download CSV Report |

| | |
|---|---|
| **EBO :** | CustomerPartyEBO |
| **Application :** | CRMOD |
| **ServiceName :** | SyncCustomerCRMODProvABCSImpl |
| **Operation :** | SyncCustomer |
| **Created on :** | Jul 29 2009 08:31:45AM GMT |
| **XSL File Path :** | $AIA_HOME/AIASource/RV2.5/aia/PIPS/Core/CRMOD/ProviderABCS/SyncCustomerCRMODProvABCSImpl/bpel/CustomerEBMList_To_ContactABMList.x |

| Source | DVM | XRef | Target |
|---|---|---|---|
| lookupXRef - SyncCustomerPartyListEBM/DataArea/SyncCustomerPartyList/PartyContact/Identification/BusinessComponentID | | CUSTOMERPARTY_PARTYCONTACTID | ListOfContact/Contact/ContactId |
| 'dummy' | | | ListOfContact/Contact/ContactId |
| SyncCustomerPartyListEBM/DataArea/SyncCustomerPartyList/PartyContact/Contact/JobTitle | | | ListOfContact/Contact/JobTitle |
| SyncCustomerPartyListEBM/DataArea/SyncCustomerPartyList/PartyContact/Contact/PersonName/FirstName | | | ListOfContact/Contact/ContactFirstName |
| SyncCustomerPartyListEBM/DataArea/SyncCustomerPartyList/PartyContact/Contact/PersonName/FamilyName | | | ListOfContact/Contact/ContactLastName |
| SyncCustomerPartyListEBM/DataArea/SyncCustomerPartyList/PartyContact/Contact/PersonName/MiddleName | | | ListOfContact/Contact/MiddleName |
| SyncCustomerPartyListEBM/DataArea/SyncCustomerPartyList/PartyContact/Contact/PersonName/Salutation | CONTACT_SALUTATION | | ListOfContact/Contact/MrMrs |

### Sample XMAN Mapping Report

XMAN can be configured to run against a Process Integration Pack (PIP), an application folder, or a specific XSL file. The output is produced in XML format and can be further rendered in HTML and comma-separated values (CSV) outputs.

If a particular mapping is too complex to analyze or is not constructed as per standard guidelines, an error message is shown and the output is highlighted to inform you that the mapping is a best effort and that more detailed analysis may be required. In the case of a code bug, you can fix the XSL and rerun the report to produce more a meaningful report.

XMAN reports are indexed and categorized using various approaches, by Enterprise Business Object (EBO) or by application, for example. This allows the reports to satisfy various governance and documentation use cases. Viewing reports by EBO can help to identify the mappings as they exist in a given connector. Viewing reports by application can provide insight into the kinds of mappings that exist for a given application, such as Oracle E-Business Suite.

The PIP Auditor also uses XMAN for validating rules that cover coding best practices in XSLT files.

**For more information** about the PIP Auditor, Using the PIP Auditor.

## 21.2. Generating XMAN Reports

This section discusses:

- Overview of Optional XMAN Command Line Switches
- How to Invoke XMAN in Single File Mode
- How to Invoke XMAN in Directory Mode
- How to Invoke XMAN in PIP Mode
- How to Invoke XMAN in All-PIP Mode

- How to Import XMAN CSV Output into Microsoft Excel

## 21.2.1. Overview of Optional XMAN Command Line Switches

The following optional command line switches can be used when generating XMAN reports:

- ***-outputFormat*** <output format>

  This is an optional switch. Valid values are ***xml***, ***html***. or ***csv***. If no output format is specified, output will be generated in all three formats.

  **Note.** For CSV output, XMAN uses the tilde (~) as a delimiter. When importing the CSV data into a Microsoft Excel spreadsheet, you must specify that the tilde is used as the delimiter in the file you are importing.

  **For more information,** see How to Import XMAN CSV Output into Microsoft Excel.

- ***-outputDir*** <output directory>

  This is an optional switch. This switch indicates where the generated reports should be created. If no output directory is specified, the reports will be created in the current directory.

- ***-version***

  This is an optional switch. Indicates the version of XMAN that is being invoked.

## 21.2.2. How to Invoke XMAN in Single File Mode

To generate an XMAN report based on a single XSL file:

1. Access a command line and invoke XMAN from $AIA_HOME/DeveloperTools/XMAN/bin.
2. Use the following switch to generate an XMAN report based on a single given file:
   - ***-inputFile*** *<file path to the ABCS XSL transformation file for which you want to run the report>*

     This is a required switch. Use this switch to provide the location of the input XSL file.
3. Optionally, you can also set other switches described in Overview of Optional XMAN Command Line Switches.

For example:

- Windows: ***xman –inputFile d:\aia\PIPS\abcs.xsl –outputFormat html –outputDir d:\output***
- Linux: ***sh xman.sh –inputFile $AIA_HOME/aia/PIPS/abcs.xsl –outputFormat html –outputDir /output***

### 21.2.3. How to Invoke XMAN in Directory Mode

When you run XMAN in directory mode, output reports are generated for the XSL files that are present in the folders or subfolders of folders whose names end with “ABCSImpl.” Any subfolders within the folders whose names end with “ABCSImpl” will be included in the report. Here, ABCSImpl is the name of the ABCS directory. XMAN will recursively process all XSL files in the directories.

To generate an XMAN report in directory mode:

1. Access a command line and invoke XMAN from $AIA_HOME/DeveloperTools/XMAN/bin.
2. Use the following switch to generate an XMAN report based on a given directory.
   - ***-inputDir*** *<path to the ABCS XSL directory for which you want to run the report>*

     This is a mandatory switch. Use this switch to provide the location of the input directory.
3. Optionally, you can also set other switches described in Overview of Optional XMAN Command Line Switches.

For example:

- Windows: ***xman –inputDir d:\aia\PIPS –outputFormat html –outputDir d:\output***
- Linux: ***sh xman.sh –inputDir $AIA_HOME/aia/PIPS –outputFormat html –outputDir /output***

### 21.2.4. How to Invoke XMAN in PIP Mode

When you run XMAN in PIP mode, it generates mapping reports based on XSL files that belong to a particular PIP. You identify these XSL files by providing the location of the GenerateScriptInput.xml PIP inventory meta file, to XMAN.

This file contains information about the directories for a given PIP, as well as the names of all services used in the PIP. XMAN uses this data to identify the XSL files for which it needs to generate reports.

To generate an XMAN report in PIP mode:

1. Access a command line and invoke XMAN from $AIA_HOME/DeveloperTools/XMAN/bin.
2. Use the following switch to generate an XMAN report based on a given PIP inputMetaFile.
   - ***-inputDir*** *<path to the root directory of the PIP source code>*

     This is a mandatory switch. You must define this input directory here as all inputMetaFile entries are defined relative to this root.
   - ***-inputMetaFile*** <file path to the GenerateScriptInput.xml file for the PIP>

     This is a mandatory switch. When this option is specified, the –inputDir switch should point to the root directory of the PIP source code, because all entries in the metafile are relative to this root.
3. Optionally, you can also set other switches described in Overview of Optional XMAN Command Line Switches.

For example:

- Windows: ***xman –inputDir D:\PIPS –inputMetaFile d:\aia\PIPS\ GenerateScriptInput.xml –outputDir d:\output***

- Linux: ***sh xman.sh –inputDir $AIA_HOME/aia –inputMetaFile $AIA_HOME/aia/PIPS/GenerateScriptInput.xml -outputDir /output***

## 21.2.5. How to Invoke XMAN in All-PIP Mode

When you run XMAN in all-PIP mode, XMAN generates mapping reports for all PIPs that have GenerateScriptInput.xml files placed in the PIPS/Core/Setup/[PIP Name]/Install directory for core PIPs and in the PIPS/Industry/[Industry Name]/Setup/[PIP Name]/Install directory for industry-specific PIPs.

Generated reports will be stored as follows:

- Core PIP reports will be placed in *<outputDir>*/Core/*<PIPName>*
- Industry PIP reports will be placed in *<outputDir>*/Industry/*<IndustryName>*/*<PIPName>*

To generate an XMAN report in all-PIP mode:

1. Access a command line and invoke XMAN from $AIA_HOME/DeveloperTools/XMAN/bin.
2. Use the following switches to generate XMAN reports in all-PIP mode.
   - ***-inputDir*** *<parent folder of the Service Location attribute value specified in GenerateInputScript.xml>*

     This is a mandatory switch.
   - ***-inputMetaFile ALL***

     This is a mandatory switch.
3. Optionally, you can also set other switches described in Overview of Optional XMAN Command Line Switches.

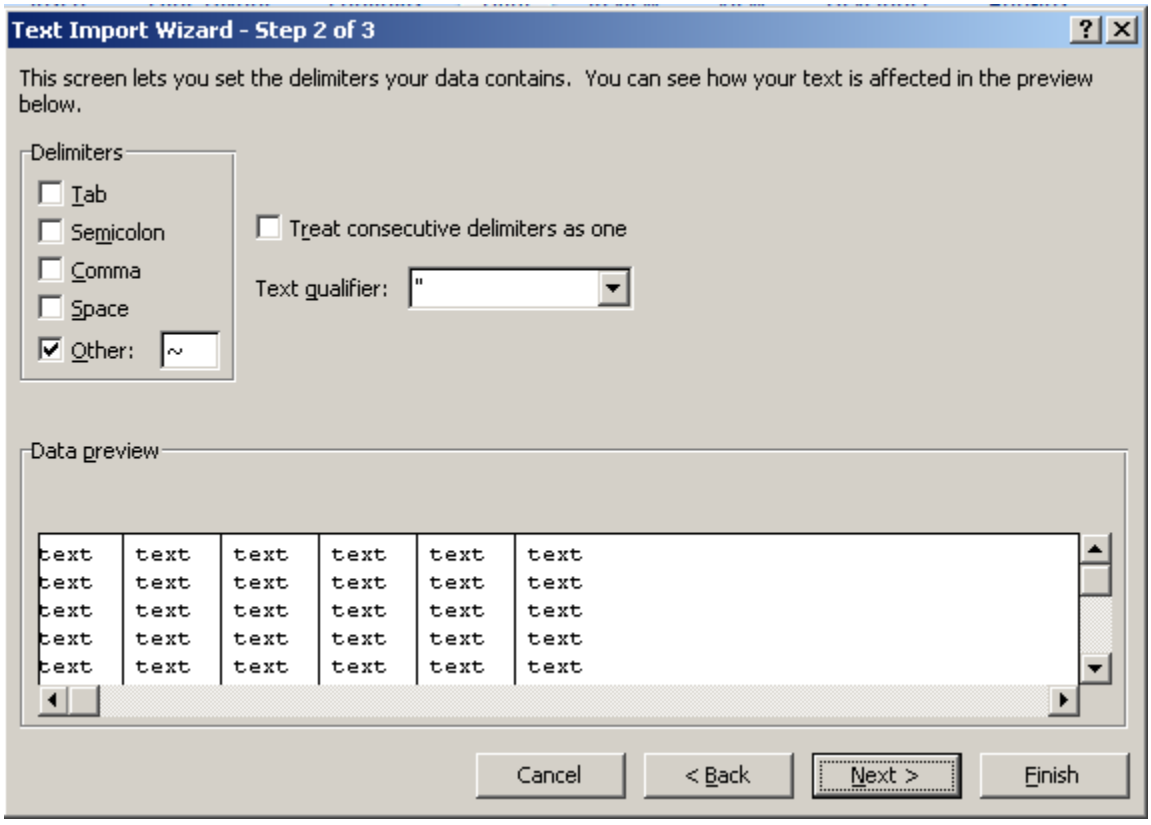
For example:

- Windows: ***xman –inputDir D:\aia –inputMetaFile ALL –outputDir d:\output***
- Linux: ***sh xman.sh –inputDir $AIA_HOME/aia –inputMetaFile ALL –outputDir /output***

## 21.2.6. How to Import XMAN CSV Output into Microsoft Excel

To import XMAN CSV report output into a Microsoft Excel spreadsheet:

1. Access Microsoft Excel and open a new file.
2. Navigate to Data, Get External Data, From Text.
3. The Import Text File dialog box displays. Select the XMAN CSV output file that you want to import.
4. The Text Import Wizard dialog box displays. In the Original data type group box, select the Delimited option and click Next.
5. In the **Delimiters** group box, select **Other** and enter **~** in the text box. In the **Text qualifier** field, accept the default value.




Text Import Wizard dialog box

6. Click Next and then click Finish.
7. The Import Data dialog box displays. Select the Existing Worksheet option and click OK. XMAN report data displays in the spreadsheet.

# 21.3. Adding XMAN Annotations to XSLT Files

This section discusses:

- Overview of XMAN Annotations in XSLT Files
- Describing XMAN Annotation Structure and Placement in XSLT Files

## 21.3.1. Overview of XMAN Annotations in XSLT Files

XMAN may not be able to handle every XPath in your ABCS implementation XSLT files. There are many scenarios in which XMAN cannot decipher the correct XPath used in the mapping due to the dynamic handling of elements or the use of expressions that can only be deciphered with the help of an input XML file.

When XMAN encounters a mapping for which it cannot decipher the complete or correct XPath, it displays it with a yellow background in the output HTML report.

To be able to generate reports that provide accurate mapping information for all XPaths, you can add XMAN annotations directly in the source XSLT files (ABCS implementation XSLT files) for the mappings that are not correctly displayed in the output report.

Once the XMAN annotation is provided in the format described in this section, XMAN ignores the derived XPath and replaces it with the annotated value in the report output.

## 21.3.2. Describing XMAN Annotation Structure and Placement in XSLT Files

The structure and placement of the XMAN annotation element and the information needs to provide to XMAN vary depending upon the available XSLT elements and style of coding in the XSLT.

The general structure of the XMAN annotation is as follows:

```
<?xman:annotation
     <xman:annotation xmlns:xman="http://www.oracle.com/aia/xman">
          <xman:source>SupplierRef/supplierSite/supplierSiteAddress/addr_
           Xref_Key</xman:source> --  optional
          <xman:dvm>myDVM</xman:dvm> -- optional
          <xman:xref>myXREF</xman:xref> -- optional
          <xman:target>TargetEBM/DataArea/addr_Xref_element</xman:target>
           -- optional
     </xman:annotation>
?>
```

Follow these placement guidelines when working with XMAN annotations in XSLT files:

- Identify the element in the source XSLT that is causing the yellow background entry in the HTML report. Focus on the <xsl:value-of> element that populates the value for the particular element. Place the <xman:annotation> element containing the annotation details before the <xsl:value-of> element.
- XPaths should be expressed without prefixes in the XMAN annotation. For example:

  ```
  <xman:source>SupplierRef/supplierSite/supplierSiteAddress/addr_Xref_Key</x
  man:source>
  ```

- If a certain undecipherable expression leads to a hardcoded value, for example, a table name is referenced in a variable, the hardcoded value should be expressed in the XMAN annotation as follows:

  ```
  <xman:source>ADDRESS_COUNTRYID</xman:source>
  ```

- If there are conditional elements enclosing the <xsl:value-of> element, you can ignore them and place the XMAN annotation element between the conditional element and the <xsl:value-of> element. For example:

  ```
  <corecom:BizId>
  <xsl:if test="$testResult='true'">
  <?xman:annotation
       <xman:annotation xmlns:xman="http://www.oracle.com/aia/xman">
            <xman:source>SuppABM/Supplier/asd</xman:source>
       <xman:annotation>
  ?>
  <xsl:value-of select="*/*[name()='abm:asd']"/>
  </xsl:if>
  ```

```
</corecom:BizId>
```

- If multiple <xsl:value-of> elements are present, each under an <xsl:when> element, place the XMAN annotation above each <xsl:value-of> element.

### 21.3.2.1. XMAN Annotation Structure for XPaths

The XMAN annotation structure for an XPath is as follows:

```
<?xman:annotation
     <xman:annotation xmlns:xman="http://www.oracle.com/aia/xman">
          <xman:source>source xpath</xman:source>
     </xman:annotation>
?>
```

### 21.3.2.2. XMAN Annotation Structure for XREFs

The XMAN annotation structure for an XREF mapping is as follows:

```
<?xman:annotation
     <xman:annotation xmlns:xman="http://www.oracle.com/aia/xman">
          <xman:source>reference value (in case of lookupXref, third
           parameter) or actual value (in case of populateXref, 5th
           parameter of xref function call) </xman:source>
          <xman:xref>xref_table_name</xman:xref>
     </xman:annotation>
?>
```

<xman:source> is the reference value that would be passed as the third argument to the lookXref() function call, or the actualValue parameter that would be passed as the fifth argument of the populateXref() call.

<xman:xref> is the XREF table name.

### 21.3.2.3. XMAN Annotation Structure for DVMs

The XMAN annotation structure for DVM mappings that are undecipherable by XMAN is as follows:

```
<?xman:annotation
     <xman:annotation xmlns:xman="http://www.oracle.com/aia/xman">
          <xman:source>xpath of the reference value</xman:source>
          <xman:dvm>dvm table name</xman:dvm>
     <xman:annotation>
?>
```

### 21.3.2.4. Example XMAN Annotation for an XREF Mapping

The following code resulted in a yellow background entry in a XMAN output HTML report:

```
<itemABO:Number>
     <xsl:value-of select="xref:lookupXRef ($xrefTableName,
      $xrefReferenceColumnName,$xrefReferenceValue, $xrefColumnName,
      false())"/>
</itemABO:Number>
```

The XMAN annotation used to address the undecipherable XREF call shown above is as follows:

```
<itemABO:Number>
     <?xman:annotation
          <xman:annotation xmlns:xman="http://www.oracle.com/aia/xman">
               <xman:source>/UpdateItemBalanceListEBM/DataArea/UpdateItem
                BalanceList/InventoryBalance/ItemReference/ItemId/Biz
                CompID</xman:source>
               <xman:xref>ITEM_ITEMID</xman:xref>
          </xman:annotation>
     ?>
     <xsl:value-of select="xref:lookupXRef ($xrefTableName,$xrefReference
      ColumnName,$xrefReferenceValue, $xrefColumnName, false())"/>
</itemABO:Number>
```

# 22. Using the PIP Auditor

This chapter discusses the following topics:

- Overview of the PIP Auditor
- Generating PIP Auditor Reports

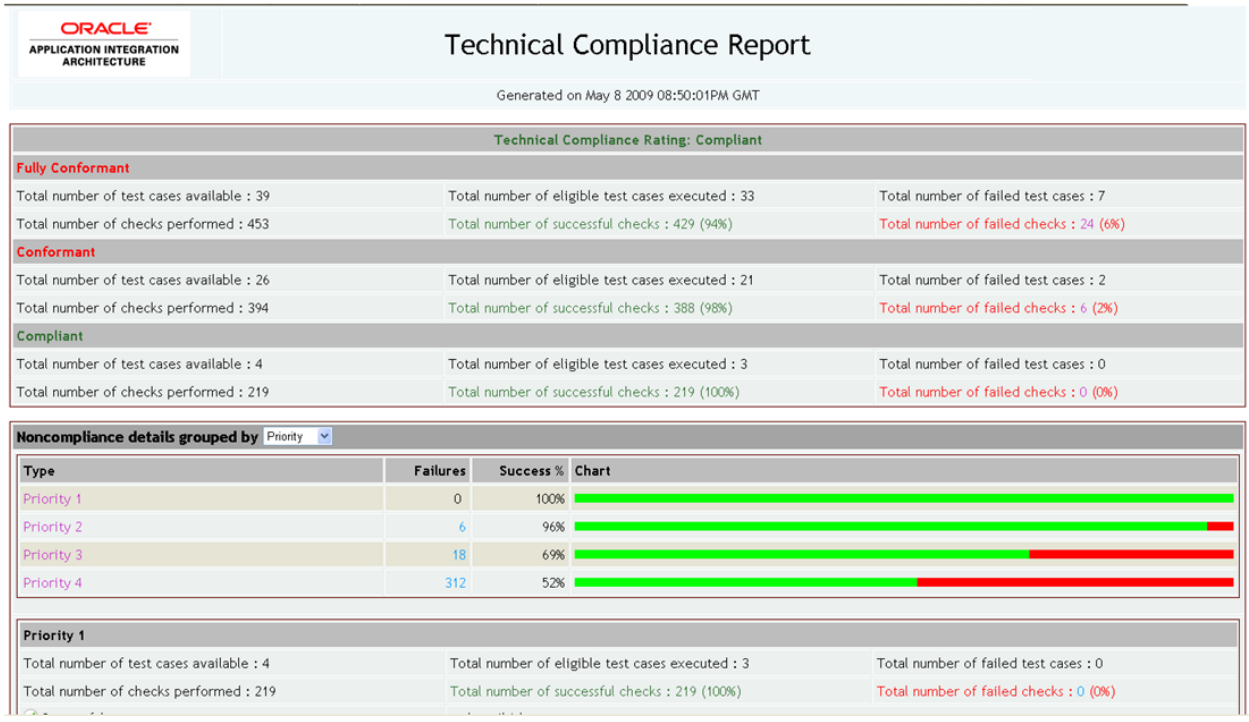
## 22.1. Overview of the PIP Auditor

Adherence to open standards and the enforcement of good coding practices are key principles of SOA governance. The Process Integration Pack (PIP) Auditor is a tool that checks for good coding practices in a PIP and qualifies the PIP as Compliant, Conformant, or Fully Conformant as per The Open Group Architecture Framework (TOGAF) standard guidelines based on the pass criteria of the highest priority rules.

PIP Auditor uses a predefined set of rules based on Oracle Application Integration Architecture (AIA) integration developer guidelines to check PIP design and code for design consistency and good coding and documentation practices.

PIP Auditor test results detail the level of compliance and pass and fail percentages. A graphical bar chart groups results by category, priority, and test suites. Results also include a list of the top 10 violating projects. The overall TOGAF compliance score for a PIP is shown in the header section of the report.

Following is a sample PIP Auditor report.



ORACLE APPLICATION INTEGRATION ARCHITECTURE

Technical Compliance Report

Generated on May 8 2009 08:50:01PM GMT

| Technical Compliance Rating: Compliant | | |
|---|---|---|
| **Fully Conformant** | | |
| Total number of test cases available : 39 | Total number of eligible test cases executed : 33 | Total number of failed test cases : 7 |
| Total number of checks performed : 453 | Total number of successful checks : 429 (94%) | Total number of failed checks : 24 (6%) |
| **Conformant** | | |
| Total number of test cases available : 26 | Total number of eligible test cases executed : 21 | Total number of failed test cases : 2 |
| Total number of checks performed : 394 | Total number of successful checks : 388 (98%) | Total number of failed checks : 6 (2%) |
| **Compliant** | | |
| Total number of test cases available : 4 | Total number of eligible test cases executed : 3 | Total number of failed test cases : 0 |
| Total number of checks performed : 219 | Total number of successful checks : 219 (100%) | Total number of failed checks : 0 (0%) |

**Noncompliance details grouped by** Priority

| Type | Failures | Success % | Chart |
|---|---|---|---|
| Priority 1 | 0 | 100% | |
| Priority 2 | 6 | 96% | |
| Priority 3 | 18 | 69% | |
| Priority 4 | 312 | 52% | |

| Priority 1 | | |
|---|---|---|
| Total number of test cases available : 4 | Total number of eligible test cases executed : 3 | Total number of failed test cases : 0 |
| Total number of checks performed : 219 | Total number of successful checks : 219 (100%) | Total number of failed checks : 0 (0%) |

PIP Auditor Technical Compliance Report

Priority 1 (P1) rules are the absolute basic rules that are the basis of the AIA philosophy. There are the "must have" rules, of which a PIP must satisfy 100% to be qualified as Compliant.

Priority 2 (P2) rules enforce more stringent criteria for certain design time patterns. A PIP meeting these rules is qualified as Conformant.

Priority 3 (P3) rules are the most stringent at the lowest levels of the technology. PIPs meeting at least a certain threshold of these rules are awarded the Fully Conformant rating.

Priority 4 (P4) rules are recently introduced rules, which may be qualified as P3, P2, or P1 rules in a future release. For the current release, P4 rules are "nice-to-have" rules and do not play a role in qualification of a PIP.

Note that these levels are additive in nature, so passing with a Priority 2 rating (conformant) means that the PIP must also pass Priority 1 rules (compliance) as well. A detailed list of predefined rules with different categories can be viewed in the delivered Rules.html file.

PIP Auditor uses a flexible XML format as input for the rule and test definitions, allowing you to customize existing rules and add new rules. The tool generates output results in XML format and renders it in HTML, providing the flexibility to modify the look and feel of the output.

# 22.2. Generating PIP Auditor Reports

This section discusses:

- How to Generate PIP Auditor Reports Using a Command Line
- How to Generate PIP Auditor Delta Reports Using a Command Line
- What You Need to Know about Generating PIP Auditor Reports

## 22.2.1. How to Generate PIP Auditor Reports Using a Command Line

For Windows, access a command line and invoke pipaudit.bat. For example: ***pipaudit –inputDir D:\AIAAudit\demo –outputDir D:\AuditOut***

For Linux, access a command line and invoke pipaudit.sh. For example: ***sh pipaudit.sh –inputDir /AIAAudit/demo –outputDir/AuditOut***

Use the following switches to configure your invocation:

- ***-inputDir*** *<PIP folder or folder that contains process(es)>*

  This is a mandatory switch that is used to indicate the location of the input directory.

  - If the ***-inputMetaFile*** switch is not specified, this input is not necessarily representative of a single PIP.
  - If the ***-inputMetaFile*** switch is provided, this specifies the PIP root directory, for example, the source folder containing the PIPs folder from AIA_HOME.

- ***-outputDir*** *<path to output folder where the audit report will generated>*

  This is a mandatory switch that is used to indicate the location where the output reports will be stored.

- ***-testFile*** <test suite file name>

This is an optional switch that is used to indicate which testSuite file PIP Auditor should run against. For example, TestSuite.3x.xml. The file should be available under [DeveloperTools install folder]/DeveloperTools/PIPAuditor/lib or [DeveloperTools install folder]/DeveloperTools/PIPAuditor/config (Tool class path), or embedded in pipauditor.jar.

- ***-testSuite*** *<test suite name>*

  This is an optional switch that is used to specify the name of the test suite on which to run a PIP audit. Using this switch will perform an audit on a specific test suite that you have defined. If you include the testSuite switch but do not provide a value, it will run the audit on all testSuites defined in your system.

- ***-testName*** *<test name>*

  This is an optional switch that is used to specify the name of the test on which to run a PIP audit. Using this switch will perform an audit on a specific test that you have defined. For example, ABCSTargetNameSpacesCheck.

- ***-inputMetaFile*** *<path to GenerateScriptInput.xml input file>*

  This is an optional switch. Use this switch if you want to run reports for a specific PIP. The GenerateScriptInput.xml input metafile contains paths pointing to the specific directories that the PIP Auditor needs to scan to generate output results that are specific to the PIP. This file contains the names of all of the services that are used in a given PIP.

  When you use this switch, you must set the *-inputDir* switch to point to the PIP root directory, since the directory paths defined in the input metafile are defined relative to this root.

  For Windows, for example: ***pipaudit –inputDir D:\AIAAudit\aia –inputMetaFile*** *<dir path of the file>****/GenerateScriptInput.xml -outputDir D:\AuditOut***

  For Linux, for example: ***sh pipaudit.sh –inputDir $AIA_HOME/aia –inputMetaFile*** *<dir path of the file>****/GenerateScriptInput.xml –outputDir /AuditOut***

- ***-inputMetaFile ALL***

  This is an optional switch. Use this switch if you want to run reports for all PIPs that have GenerateScriptInput.xml files placed in the PIPS/Core/Setup/[PIP Name]/Install directory for core PIPs and in the PIPS/Industry/[Industry Name]/Setup/[PIP Name]/Install directory for industry-specific PIPs.

  When you use this switch, you must set the *-inputDir* switch to point to the PIP root directory, since the directory paths defined in the input metafile are defined relative to this root. The output directory contains AuditSummary.xml, AuditSummany.csv, and the consolidated index.html.

  For Windows, for example: ***pipaudit –inputDir D:\AIAAudit\aia –inputMetaFile ALL -outputDir D:\AuditOut***

  For Linux, for example: ***sh pipaudit.sh –inputDir $AIA_HOME/aia –inputMetaFile ALL –outputDir /AuditOut***

  If you do not define an –outputDir switch value, generated reports will be stored as follows:

  - Core PIP reports will be placed in *<outputDir>*/Core/*<PIPName>*

  - Industry PIP reports will be placed in <*outputDir*>/Industry/<*IndustryName*>/<*PIPName*>

- ***-version***

  This is an optional switch. Indicates the version of the PIP Auditor tool that is being invoked.

## 22.2.2. How to Generate PIP Auditor Delta Reports Using a Command Line

A PIP Auditor delta report is generated by comparing a newly run PIP Auditor XML report to a previously run PIP Auditor XML report. Based on this comparison, the delta report provides details about newly compliant PIPs and newly non-compliant PIPs.

For example, this delta report can be generated during development to compare the report results of two versions of a PIP. It can also be run to provide a delta report comparing report results before and after PIP customization. The report provides information about fixes and violations introduced between the old and new reports that are being compared.

For Windows, access a command line and invoke ***PIPAuditdiff.bat***.

For Linux, access a command line and invoke ***pipauditdiff.sh***.

Use the following mandatory switches to configure your invocation:

- ***-newAuditXML*** *<file path to a newly generated PIP Auditor audit XML file>*
- ***-oldAuditXML*** *<file path to a previously generated PIP Auditor audit XML file>*
- ***-outputDir*** *<location of the output folder to which the PIP Auditor delta report will be generated>*

For Window, for example: ***PIPAuditdiff –newAuditXML D:\AIAAudit\Core_Audit.xml –oldAuditXML D:\AIAAudit\Core1_Audit.xml -outputDir D:\AuditOut***

For Linux, for example: ***sh pipauditdiff.sh –newAuditXML $AIA_HOME/Core_Audit.xml –oldAuditXML $AIA_HOME/Core1_Audit.xml –outputDir /AuditOut***

## 22.2.3. What You Need to Know about Generating PIP Auditor Reports

- The PIP Auditor reads the target namespace of a BPEL process and uses it as metadata to derive AIA-related information, such as application name, service name, service operation, industry, version, and so forth. Therefore, if the target namespace for the process has not been coded as per standards, audits will not work as designed.
- The PIP Auditor executes profile rules before executing audit rules. The profile rules derive contextual information about a project from the target namespace of a WSDL used in service components. This derived contextual information may be used in the definition of an auditing rule. Therefore, if the target namespace for the process has not been coded as per standards, audits will not work as designed. This is one of the reasons why PIP Auditor includes a targetNameSpace check as a Priority 1 check.

The profile rules store this derived AIA contextual information, such as application name, service name, service operation, industry, version, and so forth. For example, this contextual information can identify that a service is a provider Application Business Connector Service (ABCS) for participating application X and that it uses Enterprise Business Object (EBO) Y and is invoked by Enterprise Business Service (EBS) Z. This enables you to write a rule that is applicable only to provider ABCSs and not other projects, for example.

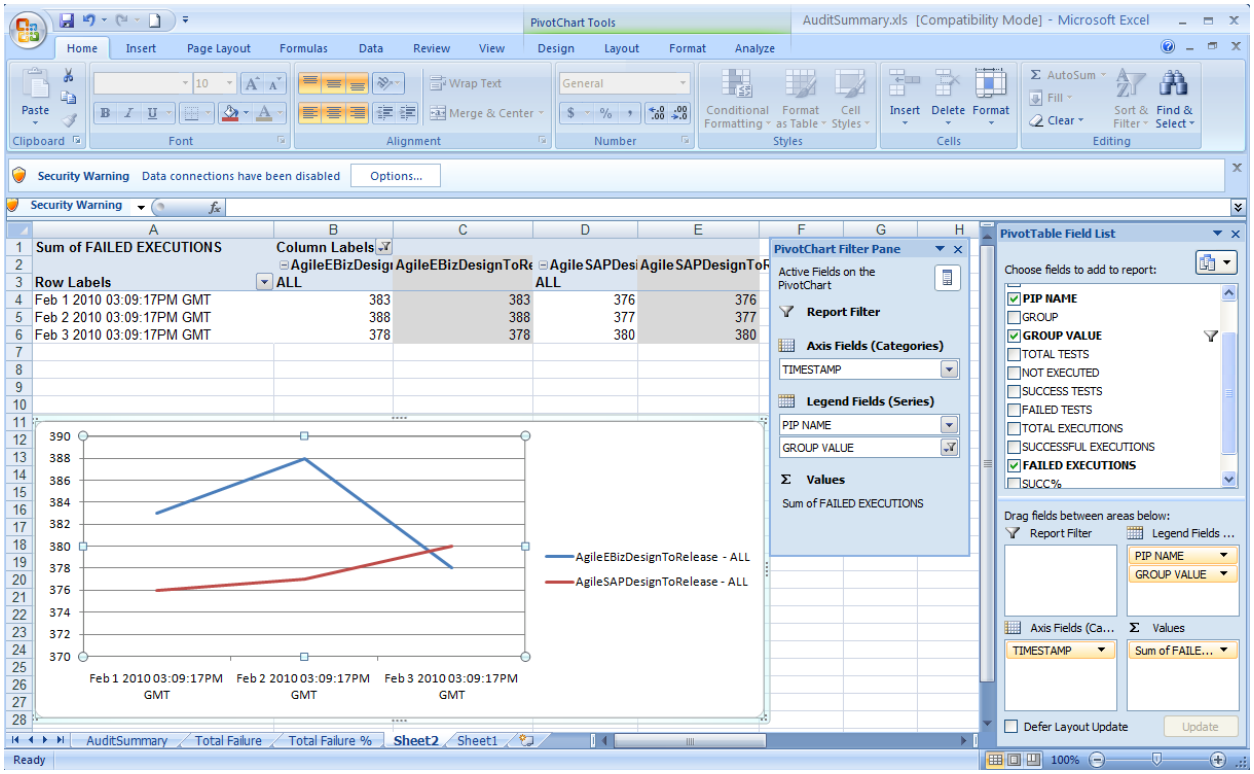
- Running PIP Auditor with the -inputDir switch value as a mapped network drive (a ClearCase mapped drive, for example) may cause performance issues. We recommend that you run PIP Auditor against local source folders.

## 22.3. Trend Analysis Chart

Users can generate trend analysis charts from AuditSummary.csv. The trend analysis chart illustrates how a selected PIP adhered to standards at different points in a timeline.

To generate a trend analysis chart in Microsoft Excel 2003:

1. Create a new Excel spreadsheet.
2. Import AuditSummary.csv into cell A1 of the newly created spreadsheet.
3. Insert a new pivot chart from the Insert menu. In the PivotTable and PivotChart Wizard, select the range of source data for which you want generate the chart. You may choose to select all imported data and filter it later while presenting the charts.
4. Select the ***New worksheet*** option to generate the chart and table in the new spreadsheet.
5. A list of PivotTable fields displays along with an empty pivot table and chart sheet. Drag and drop the DATE field to the Axis Fields area, the PIPNAME and GROUP VALUE fields to the Legend Fields area, and the FAILED EXECUTIONS field to the Values area.
6. Select the GROUP VALUE filter from the PivotChart Filter Pane and select only ***ALL*** in the drop-down list box.
7. The trend analysis chart displays.




Trend analysis chart

## 22.4. Changing Default PIP Auditor Configurations

You can change default PIP Auditor configurations by modifying the AuditorRuntime.properties file. This file is located in [DeveloperTools install folder]/DeveloperTools/PIPAuditor/config.

If any property value contains a placeholder like ${another-property-name}, the value of the property is calculated by substituting the calculated value of the property used in the placeholder. For example:

```
Version="2.5"
TestFile=Test${Version}.xml
```

The actual value of the TestFile property is Test2.5.xml, by substituting the value of version property.

| Property Name | Property Value Example | Description |
|---|---|---|
| env.VERSION | 2.5 | Version against which the PIP Auditor is going to run.<br>This default value can be overridden by defining an OS environment variable VERSION before executing PIP Auditor. |
| default.testSuiteFile | TestSuite.${env.VERSION}.xml | Used as the default value for the -testFile switch.<br>This default value can be overridden by defining a –testFile during execution of the PIP Auditor. |

| Property Name | Property Value Example | Description |
|---|---|---|
| default.ruleFile | Rules.${env.VERSION}.xml | Used as the default value for the -ruleFile switch.<br>This default value can be overridden by defining a –ruleFile switch during execution of the PIP Auditor. |
| logger.console.level | INFO | Used as the default value for the level of logging provided by the console handler.<br>The PIP Auditor uses the console handler to display execution status messages. |
| logger.file.level | INFO | Used as the default value for the level of logging for the file handler.<br>The PIP Auditor uses the file handler to write log information into the pipaudit.log file. The user can limit the amount of information logged by setting various logger levels. |
| metafile.patterns | GenerateInputScript.xml,DeployementPlan.xml | The PIP Auditor uses this property when the user defines the–inputMetaFile ALL switch.<br>The PIP Auditor will search the given input directory (using the –inputDir switch) for the metafile(s) provided in this property value. |
| metafile.components.xpaths | //Service/Location | The input metafile contains paths pointing to the specific directories that the PIP Auditor needs to scan so that output results are specific to the PIP.<br>To get a list of directories, the PIP Auditor uses this property value as an XPath to resolve the paths provided in the metafile. |
| metafile.logicalname.xpaths | //PIPName\|//ComponentName | The PIP Auditor uses this property value as an XPath to resolve the PIP Name based on information provided in the metafile. |

## 22.5. Creating Custom Rules for PIP Auditor

PIP Auditor uses the Rules.<release number>.xml file, which contains rules in a rule language that is specific to and optimized for PIP Auditor. Use the rules file specific to the release for which you want to run audits.

To create a custom rule one needs to understand the rule structure, select the appropriate rule executor, and write a new test case to execute a new rule.

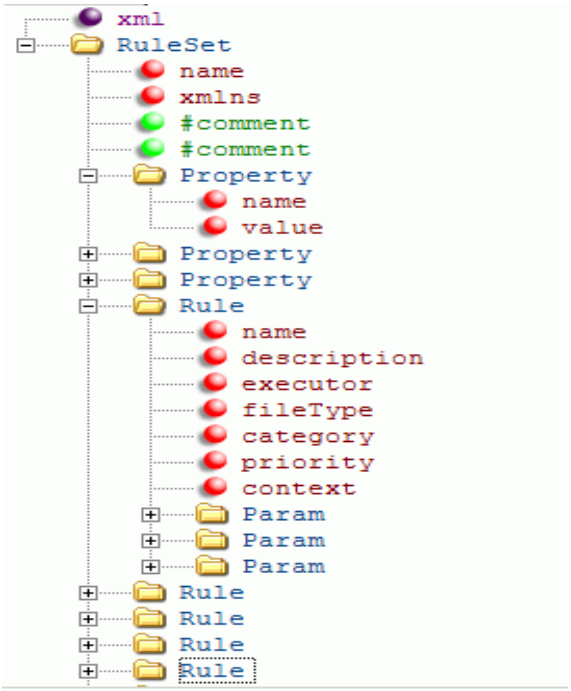
This section discusses the following topics:

- Describing a Rule
- Describing Rule Parameters
- Describing a Rule Executor
- Describing Tests and TestSuites
- Describing Rule Files
- Describing Test Suite Files
- Describing Rule and Test Releases and Customization
- How to Execute Newly Created and Customized Rules

## 22.5.1. Describing a Rule

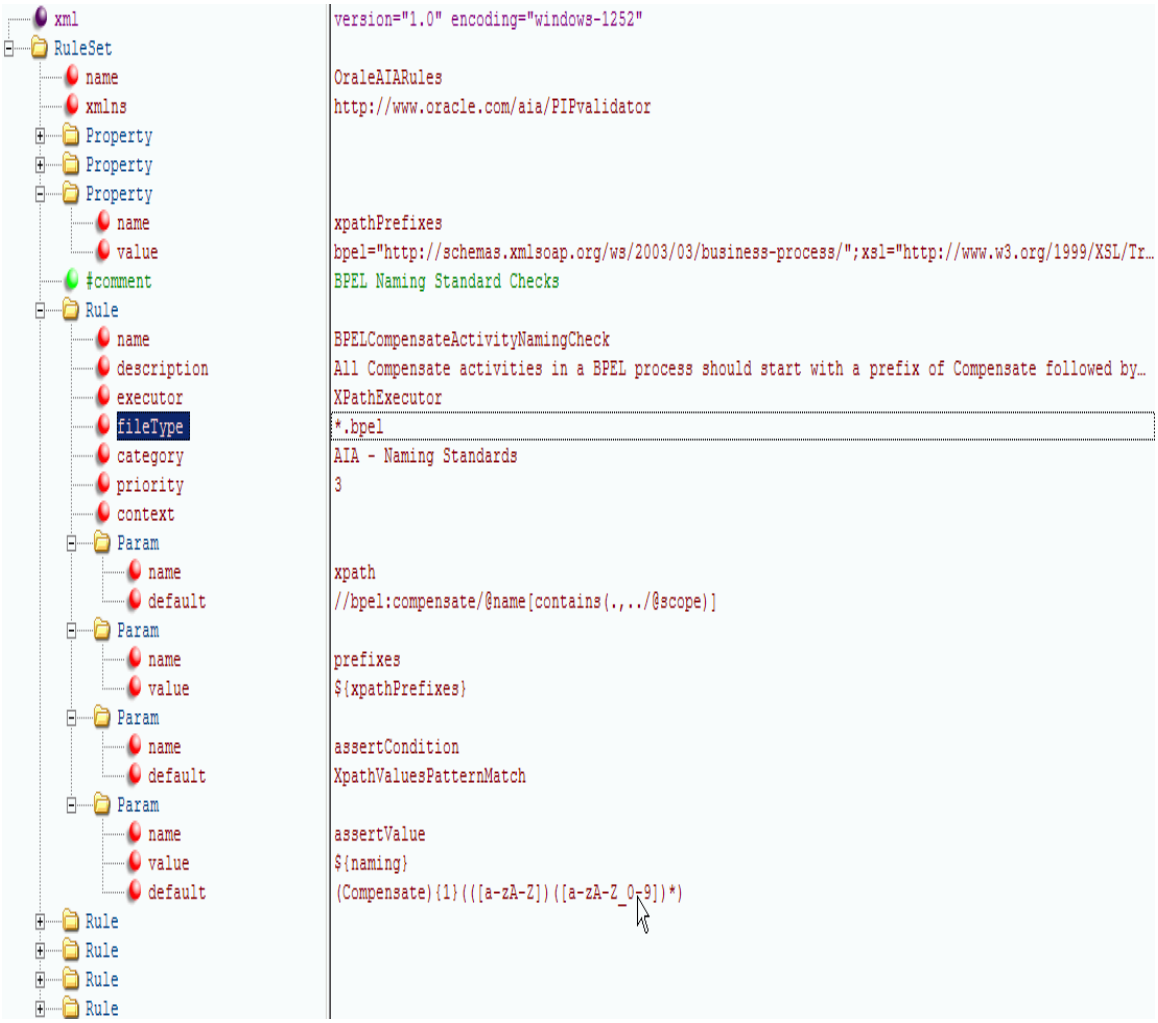
A rule is essentially a design-time governance policy in the form of an XML snippet. A rule is executed against a subset of input provided in the –inputDir and a report is generated based on compliance to the rule.

A rule has the following structure:




PIP Auditor Rule element structure

```
<Rule name=" " description=" " executor=" " fileType=" " category=" "
 priority=" " context=" ">
     <Param name=" " default=" " value=” ”/>
     <Param name=" " default=" " value=” ”/>
     <Param name=" " default=" " value=” ”/>
</Rule>
```




An example rule displayed in an XML editor

PIP Auditor executes a new rule when the TestSuite.<release number>.xml file includes a Test element with a rulename attribute set to a new rule name. Use the test suite file specific to the release for which you want to run audits. The Test element is a child of the TestSuite element, which is a placeholder for multiple tests that have been grouped together.

Use the UnitTestSuite element to test a new rule before placing it under the appropriate TestSuite element for actual use.

**For more information** about Test and TestSuite elements, see Describing Tests and TestSuites.

Following is a table of the attributes that make up the Rule element.

| Attribute | Description |
|---|---|
| name | This is the name of the rule. Note that because rules do not include an ID value, the rule name should be unique. The name value acts as the identifier for a rule. |
| description | This is a plain text description of what the rule checks. This value helps end users understand what the rule actually checks and what they need to do to |

| Attribute | Description |
|---|---|
| | achieve compliance to the rule. |
| executor | The rule engine executes rules using executors. Executors provide the base infrastructure for a rule developer to write rules. Various operations can be performed on executors. For example, XPathExecutor provides rule writers with different operations that they can perform on an XPath.<br><br>**For more information** about the executors shipped with PIP Auditor, see Appendix: Delivered PIP Auditor Rule Executors. |
| fileType | Every rule works on either files or directories. The fileType attribute gives users the flexibility to perform audits on only specific types of files. For example, fileType="*.wsdl" means only files with .wsdl extensions will be picked for the execution of a particular rule.<br>Users can choose to execute a particular rule on folders only by using "*". fileType="*" selects folders only. Note that fileType="*" is dependent on the executor. For example, fileType="*" cannot be used with XPathExecutor since XPath operations cannot be performed on a folder, whereas it can be used with FSExecutor to perform file-related operations. |
| category | This attribute is used to group sets of rules and categorize the audit results. When there are many rules, they are easier to maintain if they are categorized. It is a free-form text attribute. Audit results can be viewed based on these category values. |
| priority | This is another attribute that can be used to group the output of audit results. Numbers can be entered for this attribute and allow sorting of results. The priority of the rule is based on its importance. For example, a rule with priority="1" is critical and all PIPs must comply with the rule and produce only Compliant nodes in the _Audit.xml file. |
| context | This is used to provide additional filtering of matching files found in the input. For example, a rule writer may want to audit only utility BPEL processes for a certain rule. Assuming that all utility processes contain "util" in the process name, specifying filetype="*.bpel" and context="util" selects only utility BPEL processes for auditing. |

## 22.5.2. Describing Rule Parameters

Param is the element used to hold the parameters of the rule. These are the parameters that are passed on to the executor during the execution of the rule. Any substitutions are performed before the parameters are sent for rule execution.

```
<Param name=" " default=" " value=" "/>
```

These are the attributes of the Param element:

| Attribute | Description |
|---|---|
| name | This is the name of the parameter. |

| Attribute | Description |
|---|---|
| default | This is the default value that needs to be passed in the parameter. |
| value | This is the value of the parameter and can be expressed as either a variable or an expression. If no value is provided, then the value provided in the default attribute will be used as the actual value. |

## 22.5.3. Describing a Rule Executor

A rule executor is the underlying infrastructure provided by the PIP Auditor to write new rules. All rules are executed using one of the executors provided by the PIP Auditor. These executors provide a common mechanism to execute checks and expose different operations or methods that can be executed when supplied with a set of arguments.

For example, XPathExecutor performs different XPath-related operations on an XML file. XPathExecutor can be used to check if a value at a particular XPath location matches an input String and so forth.

Custom rules can be written using only executors delivered with PIP Auditors.

**For more information** about delivered rule executors, see Appendix: Delivered PIP Auditor Rule Executors.

## 22.5.4. Describing Tests and TestSuites

A Test is an invocation of a rule by passing parameters to be overridden in the rule, if required. For delivered tests and rules, usually nothing changes between a test and a rule. A test simply invokes the rule. However, the infrastructure supports the ability to use a test as a user-friendly interface to edit the behavior of a rule.

For example, consider a rule that checks the assign activity naming in BPEL. The delivered implementation of the rule is strict. If company XYZ wants to be more lenient when it comes to naming standards, they can simply modify the test by adding a <Param name="naming" value=" .* "/>. This change can be made directly in the rule file, but the implementation would be more complicated.

Another use case for tests would be writing multiple checks using the same rule. For example, company XYZ wants to define a character length check for an element, ABC:

- ABC length = > 10 characters = P1
- ABC length > 20 characters = P2
- ABC length < = 50 chars = P3

All three checks can be written with the help of same rule by passing different values for length.

Following is an example of a simple test:

```
<Test rulename="NoHardWiringUnamePwdInEndpointURICheck"/>
```

The value of the rulename attribute should be the same name as that given in the name attribute of the rule. In the Test element, no parameter values are passed from the test and the default values from the rule are used.

Now let's consider an example where test passes on parameters to override for execution of a rule.

Following is an example of a rule in which a test provides parameters to be used as overrides in the execution of the rule:

```
<Rule name="DocumentMinLengthCheck " description=" " executor=" "
 fileType=" " category=" " priority=" " context=" ">
     <Param name=" minLength" default=" 20" value=” ”/>
</Rule>
```

A test can be written for the rule

```
<Test rulename="DocumentMinLengthCheck ">
     <Param name=" minLength" value=" 30" />
</Test>
```

During the execution of this test, the default value of ‘20’ for the rule minLength parameter is overridden by the test minLength parameter value of ‘30’.

TestSuite is an element used to group tests. PIP Auditor results can be grouped by test suites, as well as by priority and category. Grouping of tests with the help of test suites can be helpful in prioritizing audit results.

Following is an example of a test suite:

```
<TestSuite name="FaultPolicyRelatedSuite">
     <Test rulename="FaultPolicyEnabledforABCSAndEBFCheck"/>
     <Test rulename="FaultPolicyFileExistsInABCSAndEBFCheck"/>
</TestSuite>
```

In the TestSuite element, the name attribute must have a unique value. Every TestSuite name in a TestSuite.xml file must be unique.

A TestSuite can invoke other TestSuites. This usage can be used for grouping purposes.

Following is an example of a group of TestSuites wrapped in a TestSuite:

```
<TestSuite name="AllTestSuite">
     <depends name="ABCSsecuritySuite"/>
     <depends name="SeedDataAndConfigSuite"/>
     <depends name="ESBProjectContentSuite"/>
     <depends name="BPELProjectContentSuite"/>
     <Test rulename="FaultPolicyEnabledforABCSAndEBFCheck"/>
     <Test rulename="FaultPolicyFileExistsInABCSAndEBFCheck"/>
</TestSuite>
```

For this reference mechanism, the depends element is used. The name attribute in the depends element should contain the name of the TestSuite.

Note that the parent TestSuite wrapper contains both depends and Test elements. This signifies that TestSuite can invoke other TestSuites, as well as Tests. If you execute PIP Auditor passing –testSuite AllTestSuite, this command will execute all test suites and tests specified in this example.

## 22.5.5. Describing Rule Files

A rule file is an XML file found either in the PIP Auditor’s classpath ([DeveloperTools install folder]/DeveloperTools/PIPAuditor/config) or embedded in the PIPAuditor.jar file. The .jar file can be found in [DeveloperTools install folder]/DeveloperTools/PIPAuditor/lib. The rule XML file found in [DeveloperTools install folder]/DeveloperTools/PIPAuditor/config takes precedence over the one found in the .jar file. The Oracle-delivered rule file is Rules.<release version>.xml.

The Oracle-delivered rule file can have a corresponding optional (Custom_<<base rule file name>>) custom rule XML file, which should be stored in [DeveloperTools install folder]/DeveloperTools/PIPAuditor/config. Users can create this custom rule file by copying and renaming the Custom_Rules.<release version>.xml file found in PIPAuditor/samples.

The rule XML file contains Oracle-delivered rules that are executed by PIP Auditor. A new rule or override for an existing rule should be added to the Custom rule XML file by inserting a new Rule node as a child node of the RuleSet element that has the attribute name="OracleAIARules".

```
<RuleSet name="OraleAIARules"
xmlns="http://www.oracle.com/aia/PIPvalidator">
```

Once changes to the rule file have been made, the new Custom_Rules.2.x.xml file must be placed in [DeveloperTools install folder]/DeveloperTools/PIPAuditor/config.

For a new rule added to the Custom_Rules.2.x.xml file to be executed, a corresponding Test needs to be added to the Custom_TestSuite xml file.

**For more information** about how to add a new Test to a custom TestSuite XML file, see Describing Test Suite Files.

There are three methods by which one can override an existing Oracle-delivered rule:

- Add the same rule to a custom rule XML file found in [DeveloperTools install folder]/DeveloperTools/PIPAuditor/config and modify values for parameters directly in the rule. In this case, there is no need to add a Test to the custom test suite XML file because there will already be one in the Oracle-delivered test suite XML file found in the .jar file. By default, Tests in the Oracle-delivered test suite XML file do not pass override parameters.
- Add the Test to a custom test suite XML file found in [DeveloperTools install folder]/DeveloperTools/PIPAuditor/config for a rule you are going to override. Pass parameters with override values from the Test for those parameters that have variables defined in the rule. In this case, there is no need to add a rule to the custom rule XML file.
- A less practical method would be to use a combination of the two previous methods wherein some parameters are overridden by passing them from the Test in the custom test suite XML file, while others are overridden by directly changing them in the rule added to the custom rule XML file.

**For more information** about rules and how to write them, see Describing a Rule.

## 22.5.6. Describing Test Suite Files

A test suite file is an XML file found either in the PIP Auditor's classpath ([DeveloperTools install folder]/DeveloperTools/PIPAuditor/config) or embedded in the PIPAuditor.jar file. The .jar file is located in [DeveloperTools install folder]/DeveloperTools/PIPAuditor/lib. The test suite XML file found in [DeveloperTools install folder]/DeveloperTools/PIPAuditor/config takes precedence over the one found in the .jar file. The Oracle-delivered test suite file is TestSuite.2.x.xml.

The Oracle-delivered TestSuite.2.x.xml file contains a top-level element called Validator, which contains an attribute called default. The value in default dictates which TestSuite needs to be invoked when PIP Auditor is run. This value can be overridden by the user while running PIP Auditor by using the -testSuite switch and the name of the TestSuite.

```
<Validator xmlns="http://www.oracle.com/aia/PIPvalidator" default="all">
```

The Oracle-delivered Test suite XML file can have a corresponding optional (Custom_<<base test suite file name>>) custom test suite file, which should be stored in [DeveloperTools install folder]/DeveloperTools/PIPAuditor/config. User can create this custom test suite file by copying and renaming the Custom_Testsuite.2.x.xml file in PIPAuditor/samples. The custom test suite file contains an overriding or new set of Tests that need to be executed by PIP Auditor, in addition to those already found in the Oracle-delivered TestSuite.2.x.xml. A new Test can be added by adding a Test node under the existing TestSuite element. In this way, the new Test will be executed under the test suite, along with the existing tests from this test suite found in the Oracle-delivered TestSuite.2.x.xml file.

**For more information** about Tests and TestSuites, see Describing Tests and TestSuites.

In summary, the Oracle-delivered TestSuite.2.x.xml file contains tests that will be executed when PIP Auditor runs. A test from TestSuite.2.x.xml invokes a rule from Rules.2.x.xml. Parameters can be passed from a Test to a Rule, if required. Therefore, only those rules from Rules.2.x.xml for which there is a corresponding test in the TestSuite.2.x.xml are executed. Note that the same rule can be invoked by multiple tests. This is useful if there is a need to pass different values to the same parameters of the same rule. This will override the default values for these parameters.

## 22.5.7. Describing Rule and Test Releases and Customization

Oracle delivers dedicated test suite and rule XML files with each release. While users may be able to download PIP Auditor once and use it across releases (unless otherwise specified), users must download release-specific test suite and rule files into the PIPAuditor/lib directory. You can download these files from My Oracle Support article 782351.1.

The dedicated release-specific test suite and rule XML files can be passed to PIP Auditor by using the –testFile switch command-line option accepting the test suite file name as an argument. When the user passes the –testFile command, PIP Auditor will try to identify the rule file name based on the testFile naming convention.

To avoid any annotation and merge/patch problems, it is recommended that a user make changes only in custom test suite or rule files that the PIP Auditor engine reads as overriding files. The PIP Auditor automatically identifies custom files based on their file naming patterns and their location, which should be the same location where Oracle-delivered base test and rule files are found, [DeveloperTools install folder]/DeveloperTools/PIPAuditor/config, for example.

Following are examples of test suite names and their matching patterns for rule file names:

- TestSuite.2.x.xml (Custom_TestSuite.2.x.xml) and Rules.2.x.xml (Custom_Rules.2.x.xml)
- TestSuite.25.xml (Custom_TestSuite.25.xml) and Rules.25.xml (Custom_Rules.25.xml)

If the Rules.25.xml file contains only base rules delivered by Oracle, then the Custom_Rules.25.xml file should contain all new and customized rules.

For examples of added and modified tests and rules, refer to Custom_Rules.2.x.xml and Custom_TestSuite.2.x.xml files located in DeveloperTools/PIPAuditor/sample.

### 22.5.7.1. Adding and Modifying Rules in a Custom Rule File

- To add a new rule for execution by PIP Auditor, add it in the Custom Rule file. A Corresponding Test must be added to the Custom Test suite file for a new rule to be executed.
- To customize an Oracle-delivered rule, copy the rule from the base Rule file and paste it in the Custom Rule file. Edit parameters of the rule as needed in the Custom Rule file.

- To remove a rule from execution by PIP Auditor, copy the rule from the base Rule file and paste it in the Custom Rule file. Replace the value for the executor attribute of the pasted rule to ***NAExecutor*** (Not available).

#### 22.5.7.2. Adding and Modifying Tests in a Custom Test Suite File

- To add a new test under an existing test suite, add it in the Custom Test suite file under the testSuite element for the existing test suite. If the testSuite is not present, the user must add the same node that is present in the base test suite file.
- To customize an Oracle-delivered test, copy the test from the base Test suite file and paste it in the Custom Test suite file. Edit parameters as needed in the Custom Test suite file.
- To remove a test from execution by PIP Auditor, copy the test from the base Test suite file and paste it in the Custom Test suite file. Add the active attribute with its value set to ***false***.

```
<test rulename="ABCSTargetNsCheck" active="false">
```

The Rules.html and TestSuite.html files found in the <<output directory>>/reports folder use a different text style to display a new or customized rule executed by PIP Auditor.

**Note.** Users should maintain custom rule files when they install and upgrade to the latest Developer Tools. Users must take a backup of the custom and base rule and test suite files from the [DeveloperTools install folder]/DeveloperTools/PIPAuditor/config directory and place them in the same location after the new Developer Tools installation has completed.

### 22.5.8. How to Execute Newly Created and Customized Rules

To execute newly created or customized rules, all users need to do is ensure that the base and custom Rule and Test suite files are available in the [DeveloperTools install folder]/DeveloperTools/PIPAuditor/config folder. PIP Auditor automatically detects base and custom files and executes new rules and customized rules, in addition to those available in the base Rule file.

# 23. Using the PIP Shared Artifact Analyzer

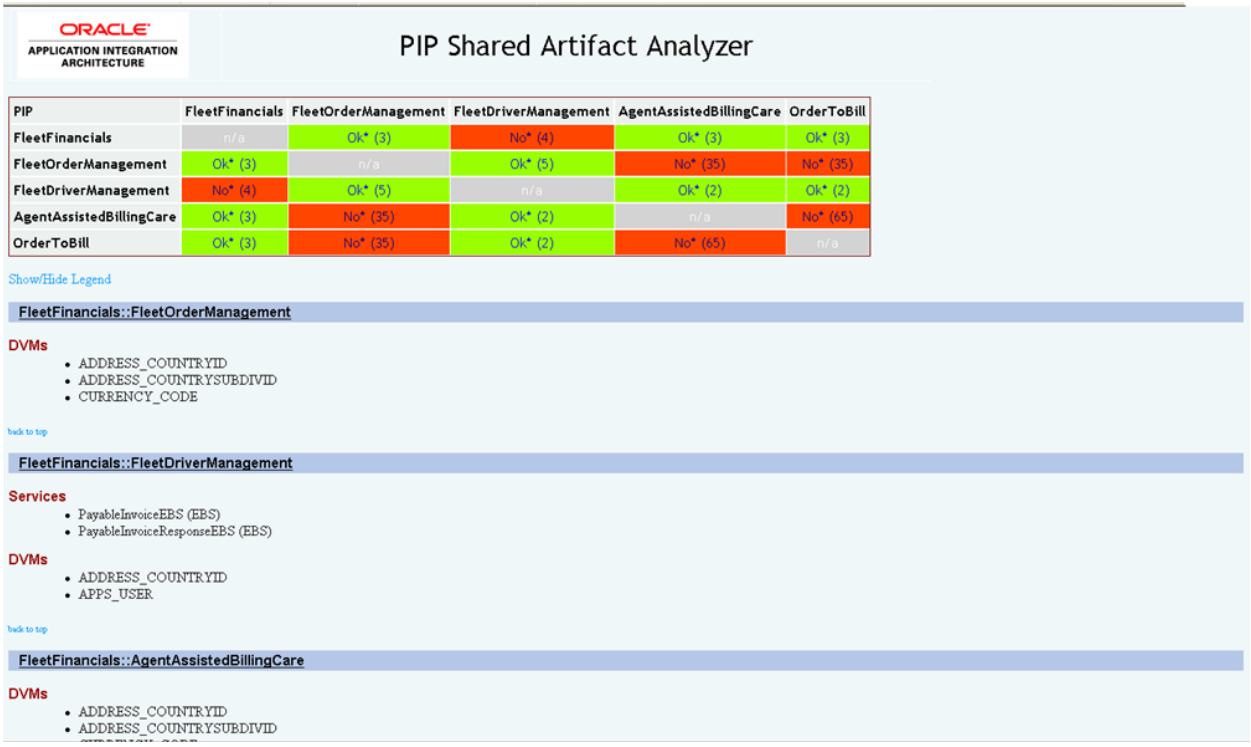
This chapter discusses the following topics:

- Overview of the PIP Shared Artifact Analyzer
- Generating PIP Shared Artifact Analyzer Reports

## 23.1. Overview of the PIP Shared Artifact Analyzer

The Process Integration Pack (PIP) Shared Artifact Analyzer processes a list of PIPs and produces a report that identifies the artifacts that are shared (reused) by each possible combination of two PIPs. The tool produces a matrix-style report that displays the number of overlapping artifacts for each PIP combination, along with an indicator of whether that PIP combination can be co-deployed to a single SOA service instance. At design-time, this information can help identify areas that may need reviews for technical standards compliance and potential co-deployment issues.

Following is a sample PIP Shared Artifact Analyzer report.



ORACLE
APPLICATION INTEGRATION ARCHITECTURE

PIP Shared Artifact Analyzer

| PIP | FleetFinancials | FleetOrderManagement | FleetDriverManagement | AgentAssistedBillingCare | OrderToBill |
|---|---|---|---|---|---|
| FleetFinancials | n/a | Ok* (3) | No* (4) | Ok* (3) | Ok* (3) |
| FleetOrderManagement | Ok* (3) | n/a | Ok* (5) | No* (35) | No* (35) |
| FleetDriverManagement | No* (4) | Ok* (5) | n/a | Ok* (2) | Ok* (2) |
| AgentAssistedBillingCare | Ok* (3) | No* (35) | Ok* (2) | n/a | No* (65) |
| OrderToBill | Ok* (3) | No* (35) | Ok* (2) | No* (65) | n/a |

Show/Hide Legend

FleetFinancials::FleetOrderManagement

DVMs
- ADDRESS_COUNTRYID
- ADDRESS_COUNTRYSUBDIVID
- CURRENCY_CODE

back to top

FleetFinancials::FleetDriverManagement

Services
- PayableInvoiceEBS (EBS)
- PayableInvoiceResponseEBS (EBS)

DVMs
- ADDRESS_COUNTRYID
- APPS_USER

back to top

FleetFinancials::AgentAssistedBillingCare

DVMs
- ADDRESS_COUNTRYID
- ADDRESS_COUNTRYSUBDIVID

PIP Shared Artifact Analyzer sample report

As with other AIA Developer Tools, the PIP Shared Artifact Analyzer uses the GenerateScriptInput.xml file to get an inventory of the services, domain-value maps (DVMs) and cross references (XRefs) that each PIP uses.

By default, the PIP Shared Artifact Analyzer looks for all GenerateScriptInput.xml files in the input source directory. Alternatively, a list of one or more specific GenerateScriptInput.xml files can be used as input via a properties file.

## 23.2. Generating PIP Shared Artifact Analyzer Reports

For Windows, access a command line and invoke psaa.bat. For example: ***psaa.bat -inputDir d:\aia -outputDir d:\temp\psaaout***

For Linux, access a command line and invoke psaa.sh. For example: ***psaa.sh -inputDir /aia -outputDir ~/psaaout***

Use the following switches to configure your invocation:

- ***-inputDir*** *<input AIA source code root directory>*

  This is a required switch. Use this switch to indicate the location of the root of the Oracle Application Integration Architecture (AIA) source code tree.

- ***-outputDir*** <output directory in which to generate the report files>

  This is an optional switch. Use this switch to indicate the location of the directory in which you want the reports to be generated. The default output directory for the report is the current working directory.

- ***-inputPropsFile*** <file path to the input properties file, including the name of the file>

  This is an optional switch. Use it to indicate the location of a properties file that you want the PIP Shared Artifact Analyzer to use as input. The properties file contains a list of the locations of the GenerateScriptInput.xml files for the PIPs that you want the tool to analyze.

  The input properties file format is as follows:

  ```
  ScriptGeneratorInputFile.count=2
  ScriptGeneratorInputFile.1=PIPS/Core/Setup/Fleet/Financials/Install/Genera
  teScriptInput.xml
  ScriptGeneratorInputFile.2=PIPS/Core/Setup/Fleet/OrderManagement/Install/G
  enerateScriptInput.xml
  …
  ```

  If this switch is not used, the tool will search the -inputDir directory tree for all GenerateScriptInput.xml files and these files will be used as input for the report.

  **Note.** The recommended approach is to NOT use the ***-inputPropsFile*** argument.

- ***-outputXML YES***

  This is an optional switch. Prior to generating the HTML report, an XML file containing all the gathered PIP data is created. By default, this XML file is not included in the output directory. Use this switch to indicate that you want this XML file to be made available in the output directory.

- ***-version***

This is an optional switch. Indicates the version of the PIP Shared Artifact Analyzer that is being invoked.

# 24. Using the XSD Flattener

This chapter discusses the following topics:

- Overview of the XSD Flattener
- Generating XSD Flattener CSV Files

## 24.1. Overview of the XSD Flattener

The XSD Flattener is a command line tool that flattens an XML schema element tree into a comma-separated values (CSV) file. This can be useful for documentation of a schema or for creating a mapping spreadsheet.

The XSD Flattener can flatten a single XSD schema file or a full or partial Oracle Application Integration Architecture (AIA) Enterprise Object Library (EOL) consisting of multiple Enterprise Business Objects (EBOs) and Enterprise Business Messages (EBMs).

The CSV files generated by the XSD Flattener contain multiple columns for each node of the input tree. The output of the tool is meant to provide as much information as possible so as to support many different uses.

The columns included in the XSD Flattener CSV output files are:

- Element name with indentation
- Element name without indentation
- Element namespace
- Element Type name
- Element Type namespace
- List of element attributes
- Custom element indicator
- Tree depth
- Cardinality minOccurs
- Cardinality maxOccurs
- Full XPath for the element
- Annotation/Documentation for the element

## 24.2. Generating XSD Flattener CSV Files

This section discusses:

- How to Flatten a Single XSD File into a CSV File

- How to Flatten a Full or Partial EOL into a CSV File

## 24.2.1. How to Flatten a Single XSD File into a CSV File

For Windows, access a command line and invoke xsd2csv.bat. For example: ***xsd2csv.bat -inputSchemaURL d:\CustomerPartyEBO.xsd -rootElement CustomerPartyEBO***

For Linux, access a command line and invoke xsd2csv. For example: ***xsd2csv.sh -inputSchemaURL /aia/CustomerPartyEBO.xsd -rootElement CustomerPartyEBO***

Use the following switches to configure your invocation:

- ***-inputSchemaURL*** *<URL or path to the XML schema file>*

  This is a required switch. Use it to indicate the location of the schema file that you want to flatten. This value can be an HTTP URL or a local file path.

- ***-rootElement*** *<name of the root element to flatten>*

  This is a required switch. Use it to indicate the name of the root element in the schema that you want to flatten and output to the CSV file.

- ***-outputFile*** *<path and name of the output CSV file>*

  This is an optional switch. Use it to specify the file name and location of the output CSV file. If you do not use this switch, the output file will be created in the current working directory and will take the file name format *<rootElement>*.csv.

- ***-csvDelimiter*** *<delimiter to use for the CSV file>*

  This is an optional switch. Use it to specify the CSV delimiter. If you do not use this switch, the default delimiter will be a comma ",".

- ***-version***

  This is an optional switch. Indicates the version of the XSD Flattener being invoked.

## 24.2.2. How to Flatten a Full or Partial EOL into a CSV File

For Windows, access a command line and invoke eol2csv.bat. For example, ***eol2csv.bat -inputDir d:\aia\EnterpriseObjectLibrary\Schemas -outputDir d:\temp\eoloutput***

For Linux, access a command line and invoke eol2csv.sh. For example, ***eol2csv.sh -inputDir /aia/EnterpriseObjectLibrary?Schemas -outputDir ~/eoloutput***

Use the following switches to configure your invocation:

- ***-inputDir*** *<EOL directory to process>*

  This is a required switch. Use it to indicate the location of the EOL directory on which you want to run the XSD Flattener. Any EBO or EBM schemas found below this directory tree will be processed. Unlike with the XSD2CSV utility, this must be a path on a local file system.

- ***-outputDir*** *<output directory path>*

This is an optional switch. Use it to specify the directory in which you want the tool to create the EBO and EBM CSV files. If you do not use this switch, the files will be created in the current working directory. The same directory tree structure found under the -inputDir value will be recreated in the -outputDir location and CSV files will be placed accordingly.

- ***-csvDelimiter*** *<delimiter to use for the CSV file>*

  This is an optional switch. Use it to specify the CSV delimiter. If you do not use this switch, the default delimiter will be a comma “,”.

- -version

  This is an optional switch. Indicates the version of the XSD Flattener being invoked.

# 25. Hosting Mapping and Technical Compliance Reports

To support technical compliance and content governance during design-time, you can provide hosted mapping and technical compliance reports in a central location.

**Note.** At this time, we provide Linux scripts only.

The $AIA_Home/DeveloperTools/bin/devtoolsenv.sh script is used to provide these reports. This file will be used to set up environment variable values for executing DeveloperTools. The following table provides descriptions of each variable.

| Variable | Description |
|---|---|
| VERSION | Specifies the release number. For example, 2.3, 2.5, and so forth. |
| INPUTDIR | Specifies the path of the directory root where the AIA source is available. For example, /scratch/aia/AIASource/RV2.5/aia. |
| REPORT_STAGING_HOME | Specifies the directory location where reports will be generated by the tools. For example, /scratch/aia/AIAReports_stage. |
| REPORT_PRODUC_HOME | Specifies the directory where reports are copied to be viewed by report viewers. For example, /scratch/aia/AIAReports. |
| AIA_HOME | Specifies the directory where the Developer Tools are installed. For example, /scratch/aia/. |

To generate reports and HTML pages for your hosted solution:

1. Change the environment variables in $AIA_HOME/DeveloperTools/bin/devtoolsenv.sh using information provided in the table above.
2. Access a command line and invoke $AIA_Home/DeveloperTools/bin/aia_runreports<release number>.sh.

   Use the script specific to the release for which you want to run reports. Running this script refreshes the source code from source control, executes XSL Mapping Analyzer (XMAN), Process Integration Pack (PIP) Auditor, PIP Shared Artifact Analyzer, and XSD Flattener tools, and generates reports in the stage directory that has been set in the script. After executing these tools, this script executes commands to produce Enterprise Business Message (EBM)-level consolidated comma-separated values (CSV) files and creates the Mapping Report Dashboard page and the technical Compliance Report Dashboard page in the stage directory.
3. Verify the presence and accuracy of the reports in the stage directory.
4. Invoke $AIA_Home/DeveloperTools/bin/aia_movestagetoprod.sh <release number>.

   Running this script removes all generated report and page content from the $STAGEDIR directory and places it in the $PRODDIR directory, as defined in the script.

5. If you want to revert content back to the version of your production site that existed before the most recent run of $AIA_Home/DeveloperTools/bin/aia_movestagetoprod.sh, invoke $AIA_Home/DeveloperTools/bin/aia_restorebaktoprod.sh <release number>.

   Running this script moves reports from the $PRODDIR directory back to the $STAGEDIR directory.

# 26. Appendix: XML Structures of Exportable CAVS Definitions and Instances

This appendix provides the following XML structures of exportable Composite Application Validation System (CAVS) definitions and instances:

- Definition.xml
- Instance.xml

## 26.1. Definition.xml

This is the structure of the Definitions.xml file created by the CAVS definition export feature.

This export feature should be used to migrate definitions between instances running on the same version of Foundation Pack.

Use this structure as a reference if you a receiving a validation error when importing definitions.

Edit this structure to create new definitions for importing to an Oracle Application Integration Architecture (AIA) Foundation Pack instance.

**For more information** about the definition export and import feature, see Exporting and Importing CAVS Definitions and Instances.

```
<DefinitionsList>

<!-- The section below is for one test/simulator definition. This
includes all definition details as well as XPATH conditions set by
the user. For each definition the section below will be repeated -->

<DefinitionsViewRORow>
   <DefinitionId>[Definition ID that was set in the previous
environment. During import, the target system will generate a new ID
for this field]</DefinitionId>
   <Type>[Test|Simulator]</Type>
   <Description>[String. Description of the test or
simulator]</Description>
   <State>[Locked|Unlocked]</State>
   <ServiceType>[Synchronous|Notify|Asynchronous two
way]</ServiceType>
   <UrlEndpoint>[URL]</UrlEndpoint>
   <SoapAction>[String. Valid soap action from the wsdl of the above
URL]</SoapAction>
   <SoapTransportType>[HTTP]</SoapTransportType>
   <MessageRequest>[SOAP Message. Request message along with CAVS
SOAP envelopes]</MessageRequest>
   <MessageResponse>[SOAP Message. Response message along with CAVS
SOAP envelopes]</MessageResponse>
```

```
   <Delay>[Integer greater than -1. Only in the case of ServiceType
Asynchronous two way]</Delay>
   <ServiceName>[String]</ServiceName>
   <ServiceVersion>[String]</ServiceVersion>
   <ProcessName>[String]</ProcessName>
   <PipName>[String]</PipName>
   <AuditedOn>[YYYY-MM-DD HH:MM:SS.M]</AuditedOn>
   <AuditedBy>[oc4jadmin]</AuditedBy>
   <!-- Namespace details from the request/response message. There
can be more than one occurrence of the section below -->
   <NsXpathsForDefinitionsRO>
      <DefinitionNsXpathsViewRORow>
         <DefinitionId>[Definition ID mentioned
above]</DefinitionId>
         <NamespaceAlias>[String. namespace alias]</NamespaceAlias>
         <Namespace>[valid namespace URL]</Namespace>
      </DefinitionNsXpathsViewRORow>
   </NsXpathsForDefinitionsRO>
   <!-- XPATH variables and values. There can be more than one
occurrence of the section below -->
   <XpathsForDefinitionsRO>
      <DefinitionXpathsViewRORow>
         <DefinitionId>[Definition ID mentioned
above]</DefinitionId>
         <XpathSeqId>[Non negative Integer]</XpathSeqId>
         <Xpath>[XPATH expression]</Xpath>
         <IsNodeText>[0|1.Applicable only for Simulator
Definitions]</IsNodeText>
         <IsNodeKey>[0|1. Applicable only for Simulator
Definitions]</IsNodeKey>
         <Condition>[OK|EQ|NE|LT|GE|LE|Not Null]</Condition>
         <IsSystemGenerated>[0|1]</IsSystemGenerated>
      </DefinitionXpathsViewRORow>
   </XpathsForDefinitionsRO>
</DefinitionsViewRORow>


<!-- The section below is for one group test definition. This
includes all definition details as well as references to Test
definitions that are mentioned above. For each such group definition
the section below will be repeated -->

<GroupDefinitions>
   <!-- There can be more than one occurrences of the section below
-->
<GroupDefinitionsViewRORow>
   <GroupDefinitionId>[Group Definition ID that was set in the
previous environment. During import the target system will generate
a new ID for this field]</GroupDefinitionId>
   <Description>[String]</Description>
   <ProcessName>[String]</ProcessName>
   <PipName>[String]</PipName>
   <GDDefinitionsViewRO>
      <!-- There can be more than one occurrences of the section
below -->
      <GDDefinitionsViewRORow>
         <GroupDefinitionId>[Group Definition ID set
```

```
above]</GroupDefinitionId>
         <SeqId>[Non negative Integer]</SeqId>
         <DefinitionId>[One of the Definition ID set in the
DefinitionsViewRORow section]</DefinitionId>
         <DefinitionSeqId>[Non negative Integer]</DefinitionSeqId>
         <ServiceType>[Synchronous|Notify|Asynchronous two
way]</ServiceType>
         <SoapTransportType>[HTTP]</SoapTransportType>
      </GDDefinitionsViewRORow>
   </GDDefinitionsViewRO>
</GroupDefinitionsViewRORow>
</GroupDefinitions>

</DefinitionsList>
```

# 26.2. Instance.xml

This is the structure of the Instance.xml file created by the CAVS instance export feature.

This export feature can be used to export a test or group instance in XML format that can be used with XML reporting tools to generate reports of test executions.

**For more information** about the instance export feature, see Exporting and Importing CAVS Definitions and Instances.

```
<InstancesList><?xml version = '1.0' encoding = 'UTF-8'?>
<InstancesViewRORow>
<!-- There would be more occurrences of this if more instances are
exported→
   <InstanceId>[Instance ID that was assigned by the environment in
which the instance was run]</InstanceId>
   <Type>[Test|Simulator|Group</Type>
   <Status>[Status of the instances being exported] </Status>
   <StartedOn>[Date and time at which the instance
started]</StartedOn>
   <EndedOn>[Date and time at which the instance ended]</EndedOn>
   <IsStaled>[0|1]</IsStaled>
   <DefinitionId>[Definition ID of the definition that generated the
instance]</DefinitionId>
   <Description>[Description of the definition ID that generated the
instance]</Description>
   <ServiceType>Synchronous|Asynchronous two-way|Asynchronous
(notify)</ServiceType>
   <SoapAction>[String. Valid SOAP action for the WSDL defined for
the definition ID]</SoapAction>
   <SoapTransportType>HTTP</SoapTransportType>
   <MessageRequest>actual request message</MessageRequest>
   <MessageResponse>actual response message</MessageResponse>
   <DefinitionsViewRO>
      <DefinitionsViewRORow>
         <DefinitionId>[Definition ID mentioned
above]</DefinitionId>
         <Type>[Type mentioned above] </Type>
         <Description>[Description mentioned above]</Description>
         <State>[Locked|Unlocked]</State>
```

```
            <ServiceType>[Service Type mentioned above] </ServiceType>
            <SoapAction>[SOAP Action mentioned above] </SoapAction>
            <SoapTransportType>HTTP</SoapTransportType>
            <MessageRequest>[Request message defined in the
corresponding Test or Simulator definition]</MessageRequest>
            <MessageResponse>[Response message defined in the
corresponding Test or Simulator definition]</MessageResponse>
            <AuditedOn>[YYYY-MM-DD HH:MM:SS.M]</AuditedOn>
            <AuditedBy>[oc4jadmin]</AuditedBy>
         </DefinitionsViewRORow>
      </DefinitionsViewRO>
      <InstanceXpathsViewRO>
         <InstanceXpathsViewRORow>
            <InstanceId>[Instance ID assigned to the
instance}</InstanceId>
            <XpathSeqId>[Non-negative integer] </XpathSeqId>
            <Status>[Status of the instance] </Status>
            <Xpath>/soap:Envelope</Xpath>
            <IsNodeKey>[0|1. Applicable only for Simulator
Definitions]</IsNodeKey>
            <Condition>[OK|EQ|NE|LT|GE|LE|Not Null]</Condition>
         </InstanceXpathsViewRORow>
      </InstanceXpathsViewRO>
      <InstanceNsXpathsViewRO>
         <InstanceNsXpathsViewRORow>
            <InstanceId>[Instance ID assigned to the instance]
</InstanceId>
            <NamespaceAlias>[String]</NamespaceAlias>
            <Namespace>[Valid namespace URL]</Namespace>
         </InstanceNsXpathsViewRORow>
      </InstanceNsXpathsViewRO>
</InstancesViewRORow></InstancesList>
```

# 27. Appendix: Understanding GenerateScriptInput.xml

This file is used to compile an inventory of Process Integration Pack (PIP) artifacts. All folder paths expressed below are relative to AIA_HOME, which should be provided to Oracle Application Integration Architecture (AIA) Developer Tools as the -inputDir parameter.

An example of a GenerateScriptInput.xml file can be accessed here: $AIA_HOME/aia/PIPS/Core/Setup/Fleet/OrderManagement/Install/GenerateScriptInput.xml.

## 27.1. Describing GenerateScriptInput.xml Elements

Here is a view of a sample GenerateScriptInput.xml file:

```
<?xml version="1.0" encoding="UTF-8"?>
<ScriptGenerator>
    <Application Name="Logistics" Version="6.0">
        <Service Location="PIPS/Core/Logistics/RequestorABCS/UpdateTransportationSalesOrderListLogisticsReqABCSImpl"/>
        <Service Location="PIPS/Core/Logistics/InboundAdapterServices/UpdatePlannedShipmentLogisticsAQConsumer"/>
        <Service Location="PIPS/Core/Logistics/ProviderABCS/QueryTransportationSalesOrderItineraryListLogisticsProvABCSImpl"/>
    </Application>
    <Application Name="Ebiz" Version="R12.1">
        <Service Location="PIPS/Core/Ebiz/InboundAdapterServices/QueryResponsibilityEbizAdapter"/>
        <Service Location="PIPS/Core/Ebiz/ProviderABCS/SyncCustomerPartyListEbizProvABCSImpl"/>
        <Service Location="PIPS/Core/Ebiz/InboundAdapterServices/SyncCustomerPartyEbizJMSConsumerV1"/>
    </Application>

    <Application Name="Siebel" Version="SIA810">
        <Service Location="PIPS/Core/Siebel/ProviderABCS/UpdateTransportationSalesOrderListSiebelProvABCSImpl"/>
        <Service Location="PIPS/Core/Siebel/RequestorABCS/SyncTransportationSalesOrderListSiebelReqABCSImpl"/>
        <Service Location="PIPS/Core/Siebel/JMSAdapterServices/TransportationSalesOrderListSiebelJMSConsumer"/>
    </Application>
    <Application Name="EBS" Version="">
        <Service Location="PIPS/Core/EBS/TransportationSalesOrder/TransportationSalesOrderEBS"/>
        <Service Location="PIPS/Core/EBS/TransportationSalesOrder/TransportationSalesOrderResponseEBS"/>
    </Application>

    <Application Name="EBSRoutingRules" Version="">
        <Service Location="PIPS/Core/Setup/Fleet/OrderManagement/EBS/TransportationSalesOrder/TransportationSalesOrderEBS"/>
        <Service Location="PIPS/Core/Setup/Fleet/OrderManagement/EBS/TransportationSalesOrder/TransportationSalesOrderResponseEBS"/>
    </Application>

    <PIPName>FleetOrderManagement</PIPName>
</ScriptGenerator>
```

Sample GenerateScriptInput.xml file

### 27.1.1. Application

This element can appear any number of times depending on the number of services that we need to group. Services are grouped under a particular application name.

For participating application services, the Application Name attribute can be set to E-Business Suite, Logistics, or Siebel, for example.

It is mandatory to create a ServiceGroup as the first element, even if one of the services in that group is an Enterprise Business Service (EBS), such as an inbound or outbound adapter. Therefore, for predefined AIA-based services, such as EBSs or Enterprise Business Flows (EBFs). The Application Name attributes should only use the following attribute values:

- EBS
- EBF

#### 27.1.1.1. Service

This element is contained within the Application element. A Service element must be the first element in the Application element. The Service element includes the Location attribute, which is used to express the location of the Service that needs to be deployed.

The Location path should start from the PIPs folder, PIPS/Core/Logistics/RequestorABCS/CreatePayableInvoiceListLogisticsReqABCSImpl, for example.

**Note**: When running any of the AIA Developer Tools, this is the path relative to the –inputDir option. The concatenation of the path given at –inputDir and the path given at Application/Service/@Location is the absolute path for any service for a PIP.

```
<Application Name="Logistics">
      <Service Location="PIPS/Core/Logistics/ServiceGroups/LogisticsServiceGroup"/>
      <Service Location="PIPS/Core/Logistics/RequestorABCS/CreatePayableInvoiceListLogisticsReqABCSImpl"/>
      <Service Location="PIPS/Core/Logistics/OutboundAdapterServices/CreatePayableInvoiceListLogisticsAQConsumer"/>
      <Service Location="PIPS/Core/Logistics/ProviderABCS/SyncCurrencyExchangeListLogisticsProvABCSImpl"/>
      <Service Location="PIPS/Core/Logistics/RequestorABCS/CreateAccountingEntryListLogisticsReqABCSImpl"/>
      <Service Location="PIPS/Core/Logistics/OutboundAdapterServices/CreateAccountingEntryListLogisticsAQConsumer"/>
      <Service Location="PIPS/Core/Logistics/RequestorABCS/CreateInvoiceListLogisticsReqABCSImpl"/>
      <Service Location="PIPS/Core/Logistics/OutboundAdapterServices/CreateInvoiceListLogisticsAQConsumer"/>
      <Service Location="PIPS/Core/Logistics/ProviderABCS/SyncSupplierPartyListLogisticsProvABCSImpl"/>
</Application>
```

Application and Service elements in GenerateScriptInput.xml

### 27.1.2. EBSRoutingRules

This element is used for listing EBSs that do not have associated routing rules. This is an optional element. The Service element is not required within this element because the service information should have already been defined as a part of the EBS Application Name attribute.

### 27.1.3. EBF

This element is used if there is an EBF in one of PIPs that need to be deployed. This is an optional element.

### 27.1.4. CommonSeedData

This element is used to specify common seed data. This is an optional element. Possible attributes are:

- DBObject

- DVMName
- XrefName
- IntegrationScenarios
- FaultPolicy
- ConnectorFactory

### 27.1.5. CompatiblePIP

This element is used to specify one or more PIPs with which the PIP is known to be compatible. This element may appear more than once.

### 27.1.6. IncompatiblePIP

This element is used to specify one or more PIPs with which the PIP is known to be incompatible. This element may appear more than once.

### 27.1.7. PIPName

This element is used to specify the name of the PIP. This should be the name of the PIP without spaces.

# 28. Appendix: Delivered PIP Auditor Rule Executors

This appendix describes available Process Integration Pack (PIP) Auditor rule executors, as well as:

- XPathExecutor Rule Parameters
- FSExecutor Rule Parameters
- Available Operations for XPathExecutor
- Available Operations for FSExecutor

Available PIP Auditor rule executors include:

| Executor Name | Description |
|---|---|
| XPathExecutor | Contains all XPath-related operations. For example, "XpathExists", "XpathNodeCountEqual". |
| FSExecutor | Contains all file system-related operations. For example, "FileExist", "FilesMatchPattern". |

## 28.1. XPathExecutor Rule Parameters

This section provides:

- Mandatory Params
- Optional Params

### 28.1.1. Mandatory Params

| Param Name | Description | Default Value | Example | Failure Situation |
|---|---|---|---|---|
| Xpath | XPath to be executed on a particular document | No default | //xsl:variable/@name | XPath execution failure exception |
| prefixes | List of delimited values of namespace prefixes that are used for a particular XPath expression execution. | No default | 'bpel="http://schemas.xmlsoap.org/ws/2003/03/business-process/"; xsl="http://www.w3.org/1999/XSL/Transform"; aiacfg="http://xmlns.oracle.com/aia/core/ | XPath execution failure exception |

| Param Name | Description | Default Value | Example | Failure Situation |
|---|---|---|---|---|
| | | | config/V1"; wsdl="http://schemas.xmlsoap.org/wsdl/"; xsd="http://www.w3.org/2001/XMLSchema"; xsd="http://www.w3.org/2001/XMLSchema"; | |
| assertCondition | Any one of the Operations supported by the XPathExecutor.<br>For more information, see Available Operations for XPathExecutor. | No default | XpathValuesPatternMatch | Unsupported Operation exception |
| assertValue | Assertion value. | No default | [a-zA-Z_0-9]*<br>Above regular expression says that variable name can contain only alphanumeric characters with underscores. | Error depending on the comparison type |

## 28.1.2. Optional Params

| Param Name | Description | Default Value | Example | Failure Situation |
|---|---|---|---|---|
| noNodeFlag | Every XPath operation assumes that the comparison between XPath execution result and assertValue can only be made if the output NodeList returned after evaluating XPath contains at least one node (default behavior except for 'XpathExists' and 'XpathNotExists' operations). Default | false | We have a test saying "all compensate activities in BPEL should start with a prefix of compensate". Now if we do not have any compensate activities in a bpel file, PIPAuditor reports a non-compliance. If we specify noNodeFlag="true", then the test would be considered as a | None |

| Param Name | Description | Default Value | Example | Failure Situation |
|---|---|---|---|---|
| | behavior is non-compliance is reported if the XPath does not return any nodes.) | | success by PIPAuditor. This is where this flag comes into picture. | |
| compareXML | There are situations when assertValue is more than just a mere String. The value can be an entire XML fragment, which contains regEx patterns. In this case a compare file argument specifies the file and Xpath specifies the fragment within the file. | No default | Used to compare standard code snippets.<br><Param name="compareXML" value="${faultXML}" default="AIAStdCode.xml"/> | None |
| compareXPATH | CompareXPATH is always used with compareXML. As mentioned above, this XPath helps us identify the XML fragment for comparison. | No default | <Param name="compareXPATH" value="//EBMHeaderPopulation/corecom:EBMHeader/P4/corecom:MessageProcessingInstruction"/><br>Now this is used with the compareXML param above.<br>When we apply the XPath (CompareXPATH) on the file (CompareXML) we get a XML fragment for comparison. In the above example, MessageProcessingInstruction returned by evaluating XPath on selected files is evaluated against MessageProcessingInstruction returned by evaluating CompareXPATH on the file CompareXML. | None |

| Param Name | Description | Default Value | Example | Failure Situation |
|---|---|---|---|---|
| compareType | Special operations, if any, to be executed to derive the assertValue. Otherwise, default behavior is executed, which is stated in the default section. | String<br>It converts the NodeList from Resultant XPath to string (ConvertNodeListToStirng) | Only supported type is length. By default, when nothing is specified, it converts the resultant NodeList to String. | None |

# 28.2. FSExecutor Rule Parameters

This section provides:

- Mandatory Params
- Optional Params

## 28.2.1. Mandatory Params

| Param Name | Description | Default Value | Example | Failure Situation |
|---|---|---|---|---|
| assertCondition | Any one of the Operations supported by the FSExecutor.<br>For more information, see Available Operations for FSExecutor. | No default | FileNotExist | Unsupported Operation exception |
| filePattern | RegularExpression for selection of matching files. | No default | [a-zA-Z_0-9_/]*(EBF)((V)[0-9]*)??.wsdl | Invalid Regular expression |

## 28.2.2. Optional Params

| Param Name | Description | Default Value | Example | Failure Situation |
|---|---|---|---|---|
| assertValue | Assertion value (String) | No default | InputFilePattern_(c\|C)ustom.xsl | None |
| fileExcludeContentPattern | Exclude of file for which content matched with this pattern (Regular Expression) | No default | .*(c\|C)ustom.xsl | None |

| Param Name | Description | Default Value | Example | Failure Situation |
|---|---|---|---|---|
| fileExcludePattern | Exclude of file for which filename matched with this pattern (Regular Expression) | No default | .*(JMSProducer\|OutboundHeader).*.wsdl | None |

## 28.3. Available Operations for XPathExecutor

| **XpathExists**<br>Checks for the existence of at least one node in the given Xpath. | | |
|---|---|---|
| Xpath | Xpath where at least one node is expected | |
| Prefixes | | |
| resolveImport | Optional param when set true enables the PIPAuditor to check for xsl files imported from local directory. | If this is set to true then even the imported files are checked for xpath, This would be helpful when a modular programming approach is used. I.e. the actual content we want to check might be in imported XLSs. Note that this parameter only resolves local file imports from same directory. |
| resolveServerImport | Optional param when set true enables the PIPAuditor to check for xsl files imported from server directory. | If set true helps resolve the imported server files. It tries to resolve the http import. |
| serverDir | Name of the server directory where the files exist. | For example, AIAComponents. PIPAuditor will then look for a directory called AIAComponents within inputDir. Once it finds it, it replaces http by the location of the dir AIAComponents. The remaining part of the file location is taken from the http server import. |

| **XpathListExist**<br>Checks for the existence of the nodes in the given Xpath List. Every xpath in the list should have at least a node. This is very similar to the XpathExist operation except that we can check for multiple xpaths. | | |
|---|---|---|
| Xpath | Xpath where zero nodes are expected | |

| **XpathListExist**<br>Checks for the existence of the nodes in the given Xpath List. Every xpath in the list should have at least a node. This is very similar to the XpathExist operation except that we can check for multiple xpaths. | | |
|---|---|---|
| Prefixes | | |
| xpath1 | | /A/B |
| xpath2 | | And so on... We can check for 'n' xpaths in this manner (xpathn) |
| resolveImport | | |
| resolveServerImport | | |
| serverDir | | |

| **XpathNotExists**<br>Checks for the non-existence of any node in the given Xpath. | | |
|---|---|---|
| Xpath | Xpath where zero nodes are expected | |
| Prefixes | | |
| resolveImport | | Same as the one in XpathExists. |
| resolveServerImport | | |
| serverDir | | |

| **XpathNodeCountLessThan**<br>Checks if the number of nodes found at the xpath is less than the assert value. | | |
|---|---|---|
| Xpath | Xpath for which node count is checked. | |
| Prefixes | | |
| assertValue | Maximum number of nodes that can be present for the xpath. Note that if your intended value is 'n' then the assert value is its 'n+1'. | All BPEL processes, which follow SYNC Request Response pattern, should not have more than 6 extension points.<br>So assert value would be '7'. |

| **XpathNodeCountGreaterThan**<br>Checks if the number of nodes found at the xpath is less than the assert value. | | |
|---|---|---|
| Xpath | Xpath for which node count is checked. | |

| **XpathNodeCountGreaterThan**<br>Checks if the number of nodes found at the xpath is less than the assert value. | | |
|---|---|---|
| Prefixes | | |
| assertValue | Minimum number of nodes that can be present for the Xpath. Note that if your intended value is 'n' then the assert value is its 'n-1'. | All BPEL processes, which follow SYNC Request Response pattern, should have minimum of 4 extension points.<br>So assert value would be '3'. |

| **XpathValuesLessThan**<br>Checks if the value in the xpath is less than the assert value. | | |
|---|---|---|
| xpath | Xpath for which node count is checked. | |
| prefixes | | |
| assertValue | Maximum number that the value from xpath can have. Note that if your intended value is 'n' then the assert value is its 'n+1'. | All BPEL processes, which follow SYNC Request Response pattern, should not have more than 6 extension points.<br>So assert value would be '7'. |

| **XpathValuesLessThanEqual**<br>Checks if the value in the xpath is less than or equal to the assert value. | | |
|---|---|---|
| xpath | Xpath for which node count is checked. | |
| prefixes | | |
| assertValue | Maximum number that the value from xpath can have. | |

| **XpathValuesGreaterThan**<br>Checks if the value in the xpath is greater than the assert value. | | |
|---|---|---|
| xpath | Xpath for which node count is checked. | |
| prefixes | | |

| **XpathValuesGreaterThan**<br>Checks if the value in the xpath is greater than the assert value. | | |
|---|---|---|
| assertValue | Minimum number of nodes that can be present for the xpath. Note that if your intended value is 'n' then the assert value is its 'n-1'. | All BPEL processes, which follow SYNC Request Response pattern, should have minimum of 4 extension points.<br>So assert value would be '3'. |

| **XpathValuesGreaterThanEqual**<br>Checks if the value in the xpath is greater than or equal to the assert value. | | |
|---|---|---|
| xpath | Xpath for which node count is checked. | |
| prefixes | | |
| assertValue | Minimum number of nodes that can be present for the xpath. | |

| **compareNodeWithRegExXML**<br>Checks if the node returned by the xpath matches the XML snippet from a file. Note that it is a regular expression comparison and the snippet can contain regular expressions. | | |
|---|---|---|
| xpath | Xpath for the node, which has to be checked. | |
| prefixes | | |
| compareXML | The XML file where the snippet for comparison lies. | ABCS WSDL should be documented as per AIA Documentation standards." The file AIAStdCode.xml for example contains all the XML snippets. So this file is the compareXML |
| compareXPATH | Xpath to derive the XML snippet from the compareXML XML file. | //ABCSwsdlDoc/wsdl:documentation<br>This Xpath will separate out just the documentation snippet from the XML. |

### CompareNodeListWithRegExXML

Checks if every node from the NodeList returned by the xpath matches the XML snippet from a file. Note that it is a regular expression comparison and the snippet can contain regular expressions. This can be used when multiple nodes from a file have to be checked against the same XML snippet.

| | | |
|---|---|---|
| xpath | Xpath for the NodeList, which has to be checked. | Every node from NodeList returned from this Xpath should be compliant to the XML snippet derived using compareXML and compareXpath. |
| prefixes | | |
| compareXML | The XML file where the snippet for comparison lies. | Catch blocks are defined as per AIA Error Handling Guidelines. The file AIAStdCode.xml for example contains all the XML snippets. So this file is the compareXML |
| compareXPATH | Xpath to derive the XML snippet from the compareXML XML file. | //catch<br>This Xpath will separate out just the documentation snippet from the XML. |

### XpathValuesEqual

Checks if the string value of every node from the NodeList returned by the xpath matches the string value specified in the assert value.

| | | |
|---|---|---|
| xpath | Xpath for the NodeList, which has to be checked. | Every node from NodeList returned from this Xpath should be compliant to the assert value. |
| prefixes | | |
| assertValue | String value to check against. | |
| noNodeFlag | | |

### XpathValuesNotEqual

Checks if the string value of every node from the NodeList returned by the xpath does not match the string value specified in the assert value.

| | | |
|---|---|---|
| xpath | Xpath for the NodeList, which has to be checked. | Every node from NodeList returned from this Xpath should be compliant to the assert value. |
| prefixes | | |
| assertValue | String value to check against. | |
| noNodeFlag | | |

| **XpathValuesPatternMatch**<br>Checks if the string value of every node from the NodeList returned by the xpath matches the regular expression pattern specified in the assert value. | | |
|---|---|---|
| xpath | Xpath for the NodeList, which has to be checked. | Every node from NodeList returned from this Xpath should be compliant to the assert value. |
| prefixes | | |
| assertValue | Regular expression pattern to check against. | "All Assign activities in a BPEL process should start with a prefix of Assign followed by activity name". The pattern would look like: (Assign){1}(_)??(([a-zA-Z])([a-zA-Z_0-9])*) |
| noNodeFlag | | |

| **XpathValuesNotMatchPattern**<br>Checks if the string value of every node from the NodeList returned by the xpath does not match the regular expression pattern specified in the assert value. This does the exact opposite check of XpathValuesPatternMatch. | | |
|---|---|---|
| xpath | Xpath for the NodeList, which has to be checked. | Every node from NodeList returned from this Xpath should be compliant to the assert value. |
| prefixes | | |
| assertValue | Regular expression pattern to check against. | "Target node should not be populated during ABM to EBM transformation in Requester ABCSImpl."<br>The following pattern would ensure that no hard coding of target id is present. :([a-zA-Z_0-9\s]*) |
| noNodeFlag | | |

| **XpathValueNotContains**<br>Checks if the string value of every node from the NodeList returned by the xpath does not contain the string specified in the assert value. | | |
|---|---|---|
| xpath | Xpath for the NodeList, which has to be checked. | Every node from NodeList returned from this Xpath should be compliant to the assert value. |
| prefixes | | |

### XpathValueNotContains

Checks if the string value of every node from the NodeList returned by the xpath does not contain the string specified in the assert value.

| | | |
|---|---|---|
| assertValue | String value to check against. | "DVMs stores should have no credentials stored.."<br>The following pattern would ensure that no tokens that are generally used to store credentials are used in DVMs.:UserName;Password;uname;pwd;username;password |
| noNodeFlag | | |

### XpathValueContains

Checks if the string value of every node from the NodeList returned by the xpath contains the string specified in the assert value.

| | | |
|---|---|---|
| xpath | Xpath for the NodeList, which has to be checked. | Every node from NodeList returned from this Xpath should be compliant to the assert value. |
| prefixes | | |
| assertValue | String value to check against. | |
| noNodeFlag | | |

### ExistsRegExXML

Iterates through children of node specified by compareXML and compareXPATH. Checks if every node in NodeList returned by executing compareXPATH on compareXML, exists in the NodeList returned by executing xpath on the test file. Note that CompareNodeWithRegExXML checks against the compareXML. The behavior is reverse in the operation. This operation iterates through all the children of the node from compareXML and makes sure each one of them is present in the NodeList from Xpath.

| | | |
|---|---|---|
| Xpath | Xpath for the NodeList, which has to be checked. | Every node from NodeList returned from this Xpath should contain all the children of the node derived from compareXML and compareXPATH. |
| Prefixes | | |
| compareXML | String value to check against. | "Ensure MessageProcessingInstruction is populated fully in ReqABM_to_EBM xsl". |
| compareXPATH | | |

### ExistsRegExXML

Iterates through children of node specified by compareXML and compareXPATH. Checks if every node in NodeList returned by executing compareXPATH on compareXML, exists in the NodeList returned by executing xpath on the test file. Note that CompareNodeWithRegExXML checks against the compareXML. The behavior is reverse in the operation. This operation iterates through all the children of the node from compareXML and makes sure each one of them is present in the NodeList from Xpath.

| | | |
|---|---|---|
| matchMode | | 1. (NODE_MUST) This is the default option. All the elements are considered for comparison. Any missing elements are reported for non-compliance.<br>2. (NODE_IGNORE). If matchMode is specified as 2, then missing nodes are not considered for comparison. E.g. Consider a XML structure with A as a parent and B and C as children (<A><B/><C/></A>). When matchMode=1 and if node B or C is absent all together, then non-compliance is reported. If we want to change this default behavior to report a compliance, then we should specify matchMode=2. Note that if a node is present then it should conform to the regular expression specified.<br>3. (NODE_OPTIONAL). This lets us pick and choose what differences we would want to ignore. We can add an attribute minoccurs="0" in any element that we would want to skip comparison when not found. E.g. Consider an XML structure where A is a parent element and has 2 children B and C (<A><B/><C/></A>). If we want a scenario where missing B's should be reported as compliance where as missing C's should be reported as non-compliance then this is how we can achieve it through matchMode: <A><B minoccurs="0"/><C/></A> |
| noNodeFlag | | |

| **NotExistsRegExXML**<br>Iterates through children of node specified by compareXML and compareXPATH. Checks if every node in NodeList returned by executing compareXPATH on compareXML, does not exist in the NodeList returned by executing xpath on the test file. Note that this does the exact reverse of ExistsRegExXML. | | |
|---|---|---|
| xpath | Xpath for the NodeList, which has to be checked. | Every node from NodeList returned from this Xpath should contain all the children of the node derived from compareXML and compareXPATH. |
| prefixes | | |
| compareXML | String value to check against. | |
| compareXPATH | | Example of this would be say we want to make sure double notifications are not sent as part of error handling. So we could check for the non-existence of certain error handling code snippets in some of the catch blocks. |
| matchMode | | Please refer to documentation of ExistsRegExXML for more details regarding this. |
| errorPath | | This would be helpful if we would want to show the user the node, which is not supposed to exist. E.g. we would want to show the user the catch block that contains the redundant call, this is how we can do it:<br><Param name="errorPath" default="@faultName"/> |
| noNodeFlag | | |

# 28.4. Available Operations for FSExecutor

| **FileExist**<br>Checks if a file of particular pattern exists in the selected directory. | | |
|---|---|---|
| filePattern | Pattern of the file to be selected. | Suppose we want to check for the existence of a config file AIAConfigurations.xml in every ABCS project, then we could select FileType="*" and context="ABCS" and then provide "AIAConfigurations.xml" in this param value. |

| **FileNotExist**<br>Checks if a file of particular pattern does not exist in the selected directory. | | |
|---|---|---|
| filePattern | Pattern of the file to be selected. | Suppose we want to check for the non-existence of a local schemas in every ABCS project, then we could select FileType="*" and context="ABCS" and then provide "*.xsd" in this param value. |

| **FilesMatchPattern**<br>Checks if the selected file name matches a particular pattern. | | |
|---|---|---|
| filePattern | Pattern of the file to be selected. | Suppose we want to check for the existence of a extension WSDL in every ABCS project, then we could select FileType="*" and context="ABCS" and then provide ".*(ABCSImpl)((V)[0-9]*)??.wsdl" in this param value. |
| assertValue | The pattern the file name has to be checked against. | We want to assert that the extension file selected matches a particular naming pattern e.g. ".*(ABCSImpl)Extension.wsdl" |

# Index