

Oracle® Fusion Middleware

Infrastructure Components and Utilities User's Guide for Oracle
Application Integration Architecture Foundation Pack

11g Release 1 (11.1.1.6.3)

E17366-08

August 2012

Describes how to use Composite Application Validation System initiators and simulators to test AIA service integrations. Test initiators simulate service invocations and simulators simulate service endpoints. Describes how to use error handling and logging components, including error notifications and trace and error logs, to support services operating in an AIA ecosystem.

Oracle Fusion Middleware Infrastructure Components and Utilities User's Guide for Oracle Application Integration Architecture Foundation Pack, 11g Release 1 (11.1.1.6.3)

E17366-08

Copyright © 2001, 2012 Oracle and/or its affiliates. All rights reserved.

Primary Author: Ramakanth Kotha

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation shall be subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License (December 2007). Oracle America, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information on content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

Contents

Preface	xi
Audience.....	xi
Oracle AIA Guides	xi
Related Documents	xi
Documentation Accessibility	xii
Conventions	xii
What's New in This Guide for Release 11.1.1.6.x	xiii
1 Introduction to the Composite Application Validation System	
1.1 Describing the Purpose of the Composite Application Validation System	1-1
1.2 Describing Key Components of the CAVS Framework.....	1-2
1.3 Describing the CAVS Design Assumptions and Knowledge Prerequisites.....	1-3
2 Preparing to Use the Composite Application Validation System	
2.1 What Can I Test Using CAVS?.....	2-1
2.2 What Are the Oracle AIA Components That I Need to Test?	2-1
2.3 Which Message Exchange Pattern Is Being Used by the Components Being Tested?	2-2
2.3.1 Describing CAVS Process Flows for Testing the Synchronous Message Exchange Pattern	2-2
2.3.2 Describing CAVS Process Flows for Testing the Asynchronous (Notify) Message Exchange Pattern	2-4
2.3.3 Describing Flows for Testing the Asynchronous Two-Way Message Exchange Pattern	2-5
2.4 Does the Scenario Need to be Unit or Flow Tested?.....	2-8
2.4.1 Describing a Unit Test Configuration.....	2-8
2.4.2 Describing a Flow Test Configuration.....	2-8
2.4.3 Describing a Complex Flow Test Configuration.....	2-9
2.5 Do I Have the Content I Need to Create the Definitions?.....	2-9
2.5.1 How to Obtain Message XML Text from a BPEL Process	2-10
3 Introduction to Defining and Running CAVS Tests Using the CAVS UI	
3.1 Describing the CAVS UI	3-1
3.2 Overview of Defining and Running CAVS Tests.....	3-3

3.3	How to Execute CAVS Definitions as Web Services	3-4
4	Creating and Modifying Test Definitions	
4.1	How to Create a Test Definition	4-1
4.2	How to Modify a Test Definition.....	4-5
4.3	How to Provide Multiple Request and Response Message Sets in a Single Test Definition	4-12
5	Creating and Modifying Simulator Definitions	
5.1	How to Create a Simulator Definition	5-1
5.2	How to Modify a Simulator Definition.....	5-4
5.3	How to Provide Multiple Request and Response Message Sets in a Single Simulator Definition	5-9
5.4	How to Create a Simulator Definition that Supports Chatty Services.....	5-11
5.5	How to Send Dynamic Responses in a Simulator Response	5-12
6	Searching for Test and Simulator Definitions	
6.1	How to Search for and Work with Test and Simulator Definitions	6-1
7	Working with Group Definitions	
7.1	How to Work with Group Definitions.....	7-1
7.2	How to Create and Modify a Group Definition	7-2
8	Defining CAVS Routing Setup IDs	
8.1	Introduction to CAVS Routing Setup IDs	8-1
8.2	How to Create CAVS Routing Setup IDs	8-3
8.3	How to Search for CAVS Routing Setup IDs.....	8-4
8.4	How to Modify Routing Setup IDs	8-6
8.5	How to Set Up CAVS Routing Configurations Without Creating Routing Setup IDs.....	8-7
9	Working with Test and Simulator Instances	
9.1	How to Work with Test and Simulator Instances	9-1
9.2	How to View Test Instance Details	9-4
9.3	How to View Simulator Instance Details	9-8
10	Working with Group Instances	
10.1	How to View Group Instances.....	10-1
10.2	How to View Group Instance Details	10-2
11	Purging CAVS-Related Cross-Reference Entries to Enable Rerunning of Test Scenarios	
11.1	Introduction to Purging CAVS-Related Cross-Reference Entries	11-1
11.2	How to Purge CAVS-Related Cross-Reference Entries to Enable Rerunning of Test Scenarios.....	11-1

12	Exporting and Importing CAVS Definitions and Instances	
12.1	How to Export and Import Definitions	12-1
12.2	How to Export Test and Simulator Instances	12-3
12.3	How to Export Group Instances	12-4
13	Introduction to Oracle AIA Error Handling	
13.1	Introduction to the Error Handling Framework	13-1
13.1.1	Fault Categories	13-3
13.2	Introduction to Error Handling for Business Faults	13-4
13.3	Introduction to Error Handling for BPEL and Mediator System Faults.....	13-4
13.4	Introduction to Error Handling for Oracle B2B Errors.....	13-5
14	Setting Up Error Handling	
14.1	Introduction to Setting Up Error Handling	14-1
14.2	How to Create Error Handling User Roles	14-4
14.3	How to Associate Email Addresses with Error Handling User Roles	14-5
14.4	How to Configure Notification Details.....	14-5
14.5	How to Set Up AIA Error Handling Configuration Details	14-5
14.5.1	What You Need to Know about Setting Up Error Handling Configurations	14-9
15	Using Error Notifications	
15.1	Introduction to Error Notifications	15-1
15.2	Setting Up Error Notification Throttling.....	15-2
15.2.1	Introduction to Error Notification Throttling.....	15-2
15.2.2	How to Enable Error Notification Throttling	15-3
15.2.3	How to Configure Error Notification Throttling Parameters	15-3
15.3	Customizing Error Notification Emails	15-5
15.3.1	Introduction to Error Notification Customization.....	15-5
15.3.1.1	EMAIL Element	15-6
15.3.1.2	FYI_EMAIL Element	15-6
15.3.1.3	URL Element	15-7
15.3.1.4	EXT_URL Element.....	15-7
15.3.2	How to Customize the Subject Line of Error Notification Emails.....	15-8
15.3.3	How to Customize the Body Text of Error Notification Emails	15-9
15.3.4	How to Customize Additional URLs Provided in Error Notification Email Body Text	15-11
15.4	Disabling Error Notifications	15-14
16	Using the Oracle BPM Worklist	
16.1	Introduction to the Oracle BPM Worklist.....	16-1
16.2	How to Enable the Oracle BPM Worklist.....	16-3
16.3	How to Use the Oracle BPM Worklist	16-3
17	Using the Message Resubmission Utility	
17.1	Introduction to the Message Resubmission Utility.....	17-1

17.2	How to Use the Message Resubmission Utility.....	17-1
17.2.1	AQ Store Based Resubmission.....	17-2
17.2.2	WLS JMS based Resubmission.....	17-3
17.2.3	Resequencer Based Resubmission.....	17-5

18 Using Trace and Error Logs

18.1	Introduction to Trace and Error Logging.....	18-1
18.2	How to Enable Trace Logging.....	18-2
18.3	How to Set Trace Log Levels.....	18-2
18.4	How to Access Trace and Error Logs.....	18-3
18.4.1	Accessing Oracle AIA Logs in the Oracle Enterprise Manager Console.....	18-3
18.4.2	Searching for Oracle AIA Log Messages.....	18-4
18.4.3	Accessing Oracle AIA Log XML Files.....	18-5

19 Accessing Oracle B2B Errors

19.1	Accessing B2B Error Reports.....	19-1
------	----------------------------------	------

A XML Structures of Exportable CAVS Definitions and Instances

A.1	Definition.xml.....	A-1
A.2	Instance.xml.....	A-3

Index

List of Examples

4-1	Request Message Format	4-12
4-2	Response Message Format.....	4-13
5-1	Request Message Format	5-10
5-2	Response Message Format.....	5-10
5-3	Request Message Format	5-12
5-4	Request SOAP Message Nodes.....	5-13
5-5	Response SOAP Message.....	5-13
5-6	Response Message Modified by CAVS.....	5-14
15-1	AIAEHNotifications.xml	15-5
15-2	EMAIL Element in AIAEHNotifications.xml	15-6
15-3	FYI_EMAIL Element in AIAEHNotifications.xml	15-6
15-4	URL Element in AIAEHNotification.xml.....	15-7
15-5	EXT_URL Element in AIAEHNotifications.xml.....	15-7
15-6	Customizing the Subject Line of Error Notification Emails	15-8
15-7	Customizing the Body Text of Error Notification Emails	15-10
15-8	Customizing Application Links in Body Text of Error Notification Emails	15-13
17-1	Sample ResubmissionParams.properties for AQ based Resubmission	17-2
17-2	Sample ResubmissionParams.properties for WLS JMS based Resubmission.....	17-4
17-3	Sample ResubmissionParams.properties for Resequencer based Resubmission	17-5
A-1	Definition.xml.....	A-1
A-2	Instance.xml	A-3

List of Figures

1-1	CAVS Test Definition	1-2
1-2	CAVS Simulator	1-3
2-1	Testing a Synchronous MEP Using a CAVS Test Definition.....	2-3
2-2	Testing a Synchronous MEP Using a CAVS Test Definition and Simulator Definition...	2-3
2-3	Testing a Synchronous MEP Using a CAVS Simulator Definition.....	2-4
2-4	Testing an Asynchronous (Notify) MEP Using a CAVS Test Definition.....	2-4
2-5	Testing an Asynchronous (Notify) MEP Using a CAVS Test Definition and Simulator Definition 2-5	
2-6	Testing an Asynchronous (Notify) MEP Using a CAVS Simulator Definition	2-5
2-7	Testing an Asynchronous Two-Way MEP Using a CAVS Test Definition	2-6
2-8	Testing an Asynchronous Two-Way MEP Using a CAVS Test Definition and Simulator Definition 2-7	
2-9	Testing an Asynchronous Two-Way MEP Using a CAVS Simulator Definition	2-7
2-10	Unit Testing a Provider ABCS	2-8
2-11	Flow Testing a Scenario	2-9
2-12	Complex Flow Testing an EBF.....	2-9
3-1	Overview of CAVS Component Relationships	3-2
4-1	Create Test Page (1 of 2).....	4-2
4-2	Create Test Page (2 of 2).....	4-2
4-3	Modify Test Definition Page (1 of 5)	4-5
4-4	Modify Test Definition Page (2 of 5)	4-6
4-5	Modify Test Definition Page (3 of 5)	4-6
4-6	Modify Test Definition Page (4 of 5)	4-6
4-7	Modify Test Definition Page (5 of 5)	4-6
4-8	Providing Multiple Request and Response Message Sets in a Single Test Definition...	4-12
5-1	Create Simulator Page (1 of 2).....	5-2
5-2	Create Simulator Page (2 of 2).....	5-2
5-3	Modify Simulator Definition Page (1 of 5)	5-5
5-4	Modify Simulator Definition Page (2 of 5)	5-5
5-5	Modify Simulator Definition Page (3 of 5)	5-5
5-6	Modify Simulator Definition Page (4 of 5)	5-6
5-7	Modify Simulator Definition Page (5 of 5)	5-6
5-8	Providing Multiple Request and Response Message Sets in a Single Simulator Definition...	5-9
6-1	Definitions Page (1 of 2).....	6-1
6-2	Definitions Page (2 of 2).....	6-2
7-1	Group Definitions Page.....	7-1
7-2	Group Definition Detail Page (New Group Definition).....	7-2
7-3	Group Definition Detail Page (Existing Group Definition)	7-3
8-1	Sample Scenarios for Using CAVS Routing Setup IDs	8-2
8-2	Create Routing Setup Page.....	8-3
8-3	Routing Setup Page	8-5
8-4	Routing Setup Page	8-6
8-5	AIA Configuration Page	8-7
9-1	Instances Page.....	9-2
9-2	Test Instances Detail Page (1 of 4)	9-5
9-3	Test Instances Detail Page (2 of 4)	9-5
9-4	Test Instances Detail Page (3 of 4)	9-5
9-5	Test Instances Detail Page (4 of 4)	9-6
9-6	Simulator Instances Detail Page (1 of 4)	9-9
9-7	Simulator Instances Detail Page (2 of 4)	9-9
9-8	Simulator Instances Detail Page (3 of 4)	9-9
9-9	Simulator Instances Detail Page (4 of 4)	9-10
10-1	Group Instances Page.....	10-1

10-2	Group Instances Detail Page	10-3
12-1	Definitions Page	12-1
12-2	Instances Page.....	12-3
12-3	Group Instances Page	12-4
13-1	Key Features of Error Handling Framework Components	13-2
13-2	Error Handling Framework Support for Capturing B2B Errors.....	13-6
14-1	Error Handling Setup Elements That Enable Error Notification Functionality	14-3
14-2	Error Handling Setup Elements That Enable Oracle Worklist Functionality	14-4
14-3	Error Notification Page (1 of 2)	14-6
14-4	Error Notification Page (2 of 2)	14-6
15-1	Sample Error Notification Email	15-2
15-2	Error Notification Page (1 of 2)	15-4
15-3	Error Notification Page (2 of 2)	15-4
15-4	Sample Error Notification Email Text Providing a Link to Flow Trace Details in the Oracle Enterprise Manager Console	15-12
15-5	Fault Details on the Enterprise Manager Console Flow Trace Page	15-12
18-1	Log Configuration Page	18-2
18-2	View Log File Page	18-4
18-3	Log Messages Page	18-5

List of Tables

4-1	Create Test Page Elements.....	4-2
4-2	Create Test Page - Test Messages Group Box Elements.....	4-4
4-3	Modify Definitions Page - Test Messages Group Box Elements.....	4-7
4-4	Prefix and Namespace Selection Grid Elements	4-9
4-5	XPath Selection Grid Elements	4-10
4-6	Linked Simulator Definition Selection Grid Elements	4-11
4-7	Group Definition Selection Grid Elements	4-11
5-1	Create Simulator Page Elements.....	5-2
5-2	Test Messages Group Box Elements.....	5-3
5-3	Modify Simulator Definition Page Elements	5-6
5-4	Prefix and Namespace Selection Grid Elements	5-7
5-5	XPath Selection Grid Elements	5-8
5-6	Simulator Instance Selection Grid	5-8
5-7	Linked Test Definition Selection Grid.....	5-9
6-1	Search Definitions Group Box Elements	6-2
6-2	Search Result Selection Grid Elements	6-3
7-1	Search Group Definitions Group Box Elements.....	7-2
7-2	Search Result Selection Grid Elements	7-2
7-3	Group Definition Detail Page Elements for New and Existing Definitions.....	7-3
7-4	Test Definition Selection Grid Elements.....	7-4
7-5	Group Instance Selection Grid Elements.....	7-4
8-1	Create Routing Setup Page Elements.....	8-4
8-2	Routing Setup Page Elements	8-5
8-3	Routing Setup Actions Area Elements.....	8-6
8-4	Routing Setup Page Elements	8-7
9-1	Search Instances Group Box Elements.....	9-2
9-2	Search Result Selection Grid Elements	9-3
9-3	Test Instances Detail Page Elements	9-6
9-4	Test Messages Group Box Elements.....	9-7
9-5	Linked Simulator Instance Selection Grid Elements	9-8
9-6	Simulator Instances Detail Page Elements	9-10
9-7	Test Messages Group Box Elements.....	9-11
9-8	Linked Test Instance Selection Page Elements	9-12
10-1	Search Group Instances Group Box Elements	10-2
10-2	Search Result Selection Grid Elements	10-2
10-3	Group Instances Detail Page Elements.....	10-3
13-1	Oracle B2B Error AQ Details	13-5
14-1	Error Notification Page Elements	14-7
18-1	Trace Log Levels of Severity.....	18-3

Preface

Welcome to *Oracle Fusion Middleware Infrastructure Components and Utilities User's Guide for Oracle Application Integration Architecture Foundation Pack*. This document describes how to use Composite Application Validation System initiators and simulators to test AIA service integrations. Test initiators simulate service invocations and simulators simulate service endpoints. It describes how to use error handling and logging components, including error notifications and trace and error logs, to support services operating in an AIA ecosystem.

Audience

This document is intended for users of the components and utilities delivered with Oracle Application Integration Architecture Foundation Pack.

Oracle AIA Guides

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

- *Oracle Fusion Middleware Installation and Upgrade Guide for Oracle Application Integration Architecture Foundation Pack 11g Release 1 (11.1.1.6.3)*
- *Oracle Fusion Middleware Concepts and Technologies Guide for Oracle Application Integration Architecture Foundation Pack 11g Release 1 (11.1.1.6.3)*
- *Oracle Fusion Middleware Developer's Guide for Oracle Application Integration Architecture Foundation Pack 11g Release 1 (11.1.1.6.3)*
- *Oracle Fusion Middleware Reference Process Models User's Guide for Oracle Application Integration Architecture Foundation Pack 11g Release 1 (11.1.1.6.3)*
- *Oracle Fusion Middleware Migration Guide for Oracle Application Integration Architecture 11g Release 1 (11.1.1.6.3)*
- *Oracle Fusion Middleware Product-to-Guide Index for Oracle Application Integration Architecture Foundation Pack 11g Release 1 (11.1.1.6.3)*

Related Documents

For more information, see the following documents in the Oracle SOA Suite and Oracle Business Process Management Suite 11g Release 1 (11.1.1.6.3) documentation set:

- *Oracle Fusion Middleware Installation Guide for Oracle SOA Suite and Oracle Business Process Management Suite 11g Release 1 (11.1.1.6.3)*

- *Oracle Fusion Middleware Administrator's Guide for Oracle SOA Suite and Oracle Business Process Management Suite 11g Release 1 (11.1.1.6.3)*
- *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite 11g Release 1 (11.1.1.6.3)*
- *Oracle Fusion Middleware Security and Administrator's Guide for Web Services 11g Release 1 (11.1.1.6.3)*
- *Oracle Fusion Middleware User Guide for Oracle Enterprise Repository 11g Release 1 (11.1.1.6.3)*

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc>.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info> or visit <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs> if you are hearing impaired.

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
<i>italic</i>	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
monospace	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

What's New in This Guide for Release 11.1.1.6.x

This guide has no updates for the Release 11.1.1.6.x.

For a list of known issues (release notes), see the "Known Issues for Oracle SOA Products and Oracle AIA Foundation Pack" at

<http://www.oracle.com/technetwork/middleware/docs/soa-aiafp-knownissuesindex-364630.html>.

Introduction to the Composite Application Validation System

This chapter describes the purpose and key components of the Composite Application Validation System (CAVS). It also describes design assumptions and knowledge prerequisites.

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.

This chapter includes the following sections:

- [Section 1.1, "Describing the Purpose of the Composite Application Validation System"](#)
- [Section 1.2, "Describing Key Components of the CAVS Framework"](#)
- [Section 1.3, "Describing the CAVS Design Assumptions and Knowledge Prerequisites"](#)

1.1 Describing the Purpose of the Composite Application Validation System

In the context of AIA, where there is a sequence of service invocations; spanning Application Business Connector Services (ABCs), 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.

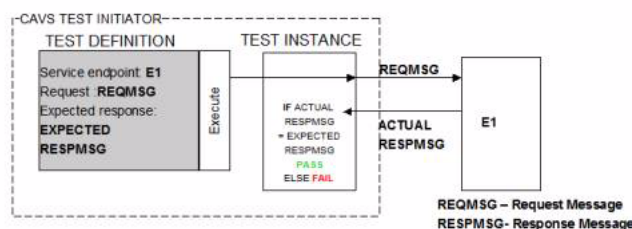
The CAVS user creates a definition using the CAVS user interface (UI) to define the service endpoint URL that needs to be invoked, as well as the request message that will be passed along with metadata about the test definition itself.

For more information about creating test definitions, see [Chapter 4, "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.

Figure 1–1 illustrates the high-level concept of the test initiator.

Figure 1–1 CAVS Test Definition



Simulator Definition

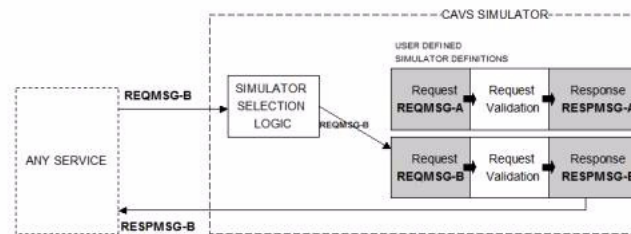
The CAVS simulator is used to simulate a web service. Simulators typically contain predefined responses for a specific request. CAVS users create several simulator definitions, each for a specific set of input.

At run time, 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 [Chapter 5, "Creating and Modifying Simulator Definitions."](#)

Figure 1–2 illustrates the high-level concept of the CAVS simulator:

Figure 1–2 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 after 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 [Chapter 11, "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

Preparing to Use the Composite Application Validation System

This chapter provides a high-level discussion of questions you should answer to help gather requirements for the tests you want to create and run in the Composite Application Validation System (CAVS).

Before you start creating and running tests in the CAVS, take the time to gather your test requirements and plan your approach to using CAVS.

This chapter includes the following sections:

- [Section 2.1, "What Can I Test Using CAVS?"](#)
- [Section 2.2, "What Are the Oracle AIA Components That I Need to Test?"](#)
- [Section 2.3, "Which Message Exchange Pattern Is Being Used by the Components Being Tested?"](#)
- [Section 2.4, "Does the Scenario Need to be Unit or Flow Tested?"](#)
- [Section 2.5, "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?

This section includes the following topics:

- [Section 2.3.1, "Describing CAVS Process Flows for Testing the Synchronous Message Exchange Pattern"](#)
- [Section 2.3.2, "Describing CAVS Process Flows for Testing the Asynchronous \(Notify\) Message Exchange Pattern"](#)
- [Section 2.3.3, "Describing Flows for Testing the Asynchronous Two-Way Message Exchange Pattern"](#)

After 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

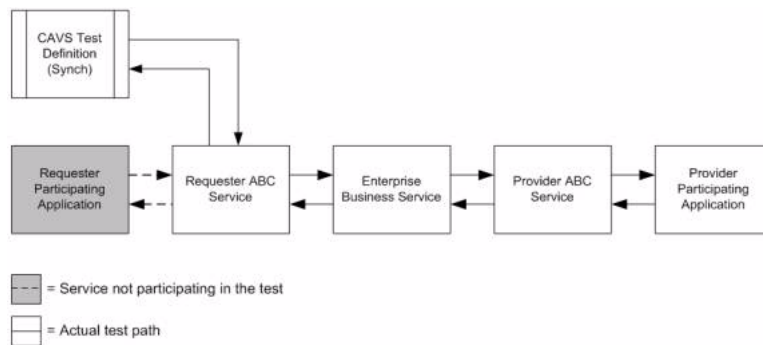
The following diagrams describe CAVS process flows for testing a provider ABCS using a synchronous MEP.

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.

[Figure 2–1](#) illustrates testing a synchronous MEP using a CAVS test definition.

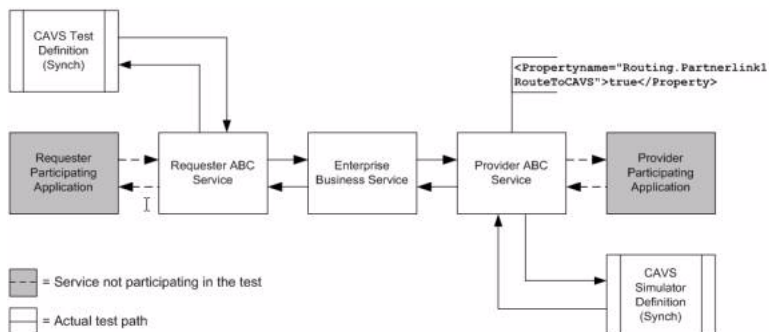
Figure 2–1 Testing a Synchronous MEP Using a CAVS Test Definition**Synchronous MEP Testing Flow Using a Test Definition and Simulator Definition**

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.

Figure 2–2 illustrates testing a synchronous MEP using CAVS test and simulator definitions.

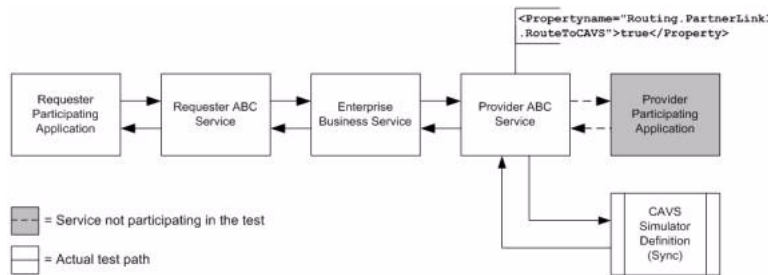
Figure 2–2 Testing a Synchronous MEP Using a CAVS Test Definition and Simulator Definition**Synchronous MEP Testing Flow Using a Simulator Definition**

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.

Figure 2–3 illustrates testing a synchronous MEP using a CAVS simulator definition.

Figure 2–3 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.

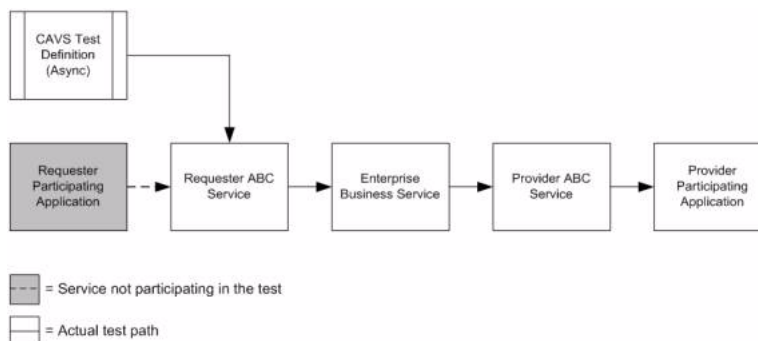
These sample flows can be used as the basis for testing other artifacts as well, such as the requester ABCS, EBF, or the provider service itself.

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.

Figure 2–4 illustrates testing an asynchronous (notify) MEP using a CAVS test definition.

Figure 2–4 Testing an Asynchronous (Notify) MEP Using a CAVS Test Definition



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

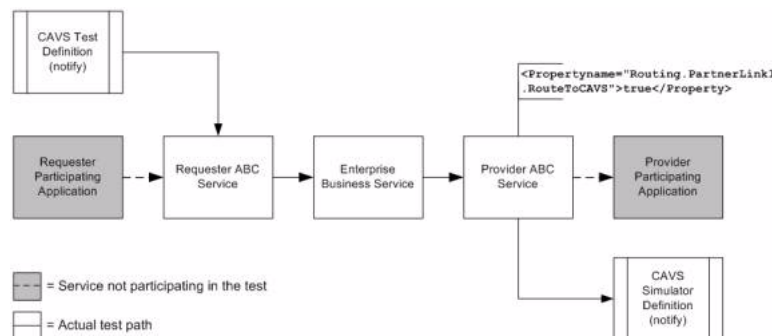
The requester participating application is replaced by the CAVS test definition. The test definition points to the URL of the requester ABCS. It uses a composed request message to invoke the ABCS and does not expect a message in response.

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.

Figure 2–5 illustrates testing an asynchronous (notify) MEP using CAVS test and simulator definitions.

Figure 2–5 Testing an Asynchronous (Notify) MEP Using a CAVS Test Definition and Simulator Definition



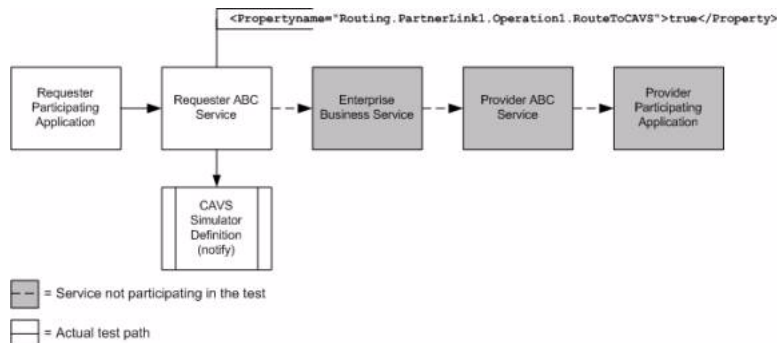
Asynchronous (Notify) MEP Testing Flow Using a Simulator Definition

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.

Figure 2–6 illustrates testing an asynchronous (notify) MEP using a CAVS simulator definition.

Figure 2–6 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.

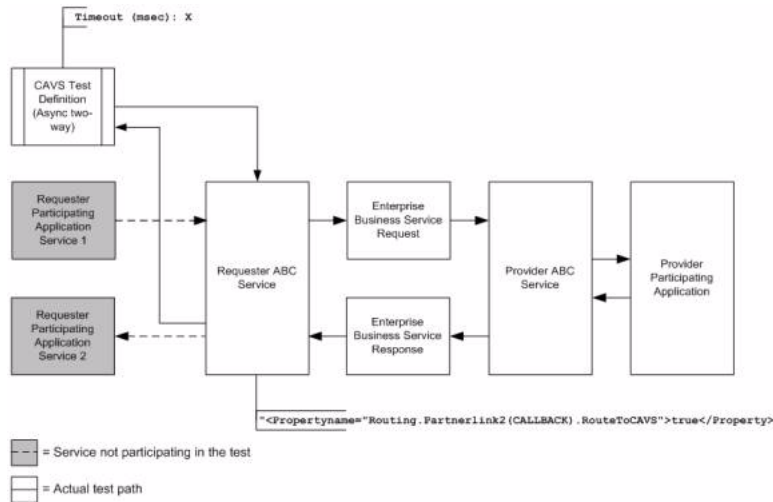
Asynchronous Two-Way MEP Testing Flow Using a Test Definition

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.

Figure 2-7 illustrates testing an asynchronous (two-way) MEP using a CAVS test definition.

Figure 2-7 Testing an Asynchronous Two-Way MEP Using a CAVS Test Definition



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

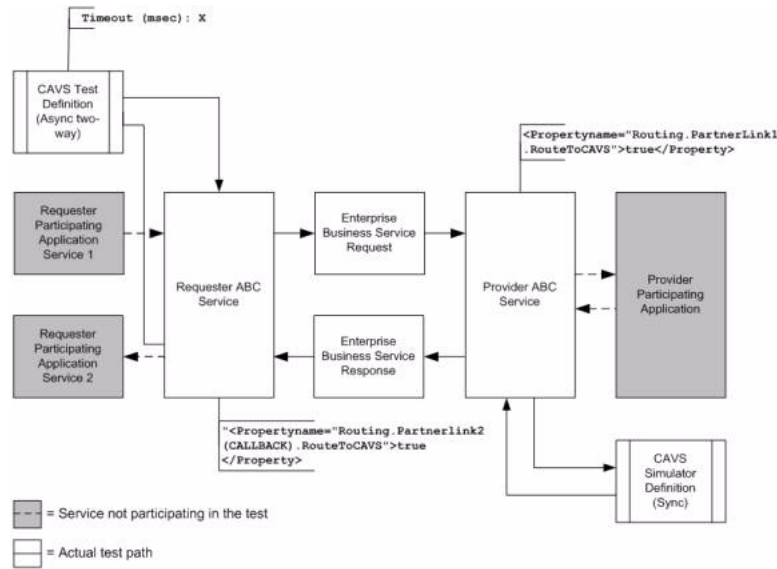
The requester participating application is replaced by the CAVS test definition. The test definition points to the URL of the requester ABCS. It uses a composed request message to invoke the ABCS and expects an eventual message in response. The test definition includes a timeout value. If no response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

Figure 2-8 illustrates testing an asynchronous (two-way) MEP using CAVS test and simulator definitions.

Figure 2–8 Testing an Asynchronous Two-Way MEP Using a CAVS Test Definition and Simulator Definition



Asynchronous Two-Way MEP Testing Flow Using a Simulator Definition

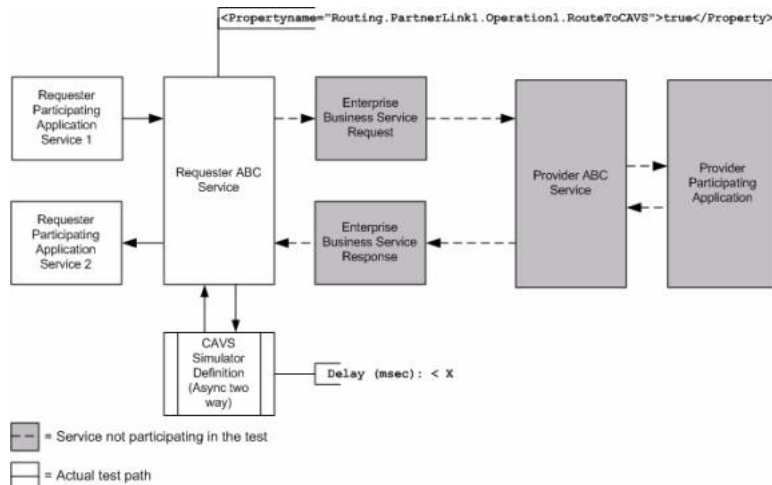
The provider ABCS is replaced by the CAVS simulator definition. The requester ABCS is programmed to route to this simulator instead of having the flow reach the provider ABCS.

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.

Figure 2–9 illustrates testing an asynchronous (two-way) MEP using a CAVS simulator definition.

Figure 2–9 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.

This section includes the following topics:

- [Section 2.4.1, "Describing a Unit Test Configuration"](#)
- [Section 2.4.2, "Describing a Flow Test Configuration"](#)
- [Section 2.4.3, "Describing a Complex Flow Test Configuration"](#)

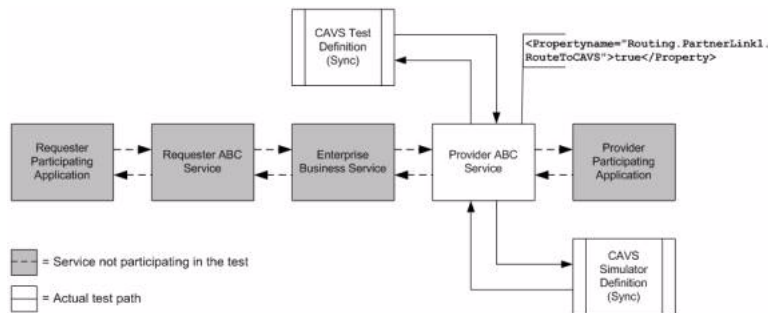
2.4.1 Describing a Unit Test Configuration

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.

[Figure 2–10](#) illustrates how to unit test a provider ABCS.

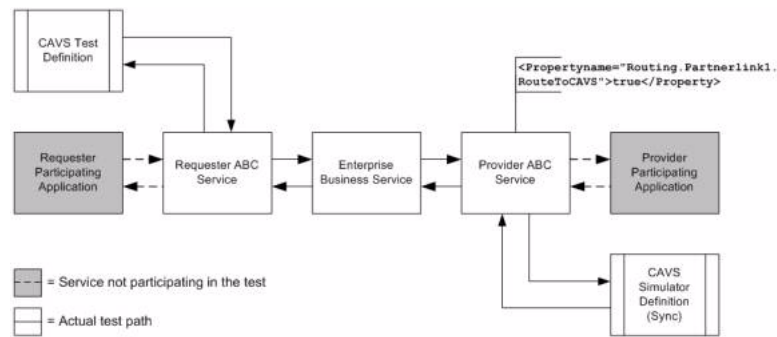
Figure 2–10 Unit Testing a Provider ABCS



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.

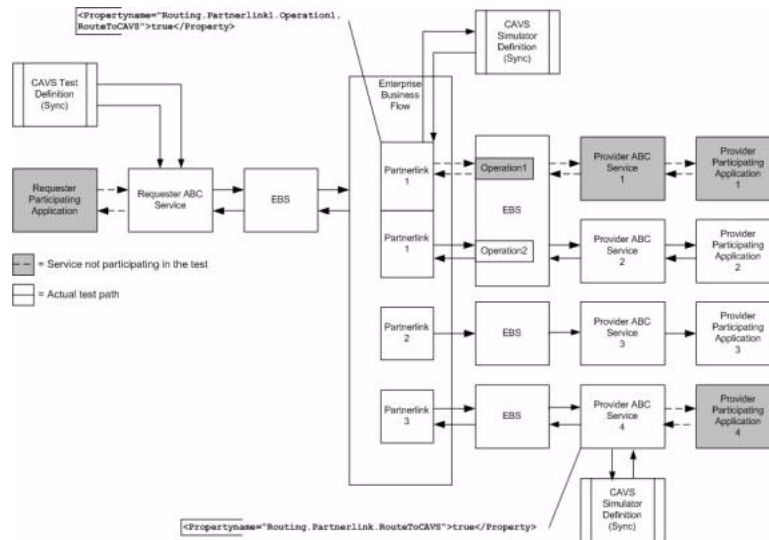
After you have unit tested the components in a scenario, you can flow test the scenario. To flow test a scenario, place a test definition before the requester ABCS at the front of the scenario and a simulator definition after the provider ABCS at the end of the scenario, as shown in [Figure 2–11](#).

Figure 2–11 Flow Testing a Scenario

2.4.3 Describing a Complex Flow Test Configuration

This section uses 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, as shown in Figure 2–12.

Figure 2–12 Complex Flow Testing an EBF

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

This section includes the following topic: [Section 2.5.1, "How to Obtain Message XML Text from a BPEL Process"](#).

After you know what you need to test and which CAVS definitions you need to create, assess whether 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. After 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 **reply 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.

Introduction to Defining and Running CAVS Tests Using the CAVS UI

This chapter describes the Composite Application Validation System (CAVS) user interface and provides an overview of how to define and run tests. It also provides instructions on how to execute CAVS definitions as web services.

This chapter includes the following sections:

- [Section 3.1, "Describing the CAVS UI"](#)
- [Section 3.2, "Overview of Defining and Running CAVS Tests"](#)
- [Section 3.3, "How to Execute CAVS Definitions as Web Services"](#)

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 [Chapter 4, "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 [Chapter 5, "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.

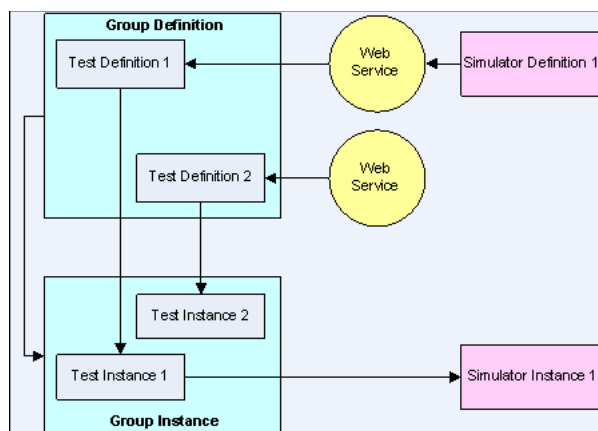
Group Instances

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

Component Overview

[Figure 3–1](#) provides a high-level overview of the relationships among the CAVS components discussed in this section.

Figure 3–1 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.

To define and run tests using the CAVS UI:

1. Assess your test requirements and gather required content.

For more information, see [Chapter 2, "Preparing to Use the Composite Application Validation System."](#)

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

For more information, see [Chapter 4, "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 [Chapter 5, "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 [Chapter 7, "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 [Chapter 8, "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 [Section 8.5, "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 [Chapter 6, "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 [Chapter 7, "Working with Group Definitions."](#)

8. After an individual test has been executed, view the 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 [Chapter 9, "Working with Test and Simulator Instances."](#)

After 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 [Chapter 10, "Working with Group Instances."](#)

9. After 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 [Section 8.5, "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>/AIAValidationSystemService/AIAValidationSystemA
PIServiceSoapHttpPort.`

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>/AIAValidationSystemService/AIAValidationSystemA
PIServiceSoapHttpPort?wsdl.`

Creating and Modifying Test Definitions

This chapter describes how to create and modify test definitions and how to provide multiple request and response message sets in a single test definition.

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 includes the following sections:

- [Section 4.1, "How to Create a Test Definition"](#)
- [Section 4.2, "How to Modify a Test Definition"](#)
- [Section 4.3, "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 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.

The Create Test page displays, as shown in [Figure 4-1](#) and [Figure 4-2](#).

Figure 4–1 Create Test Page (1 of 2)

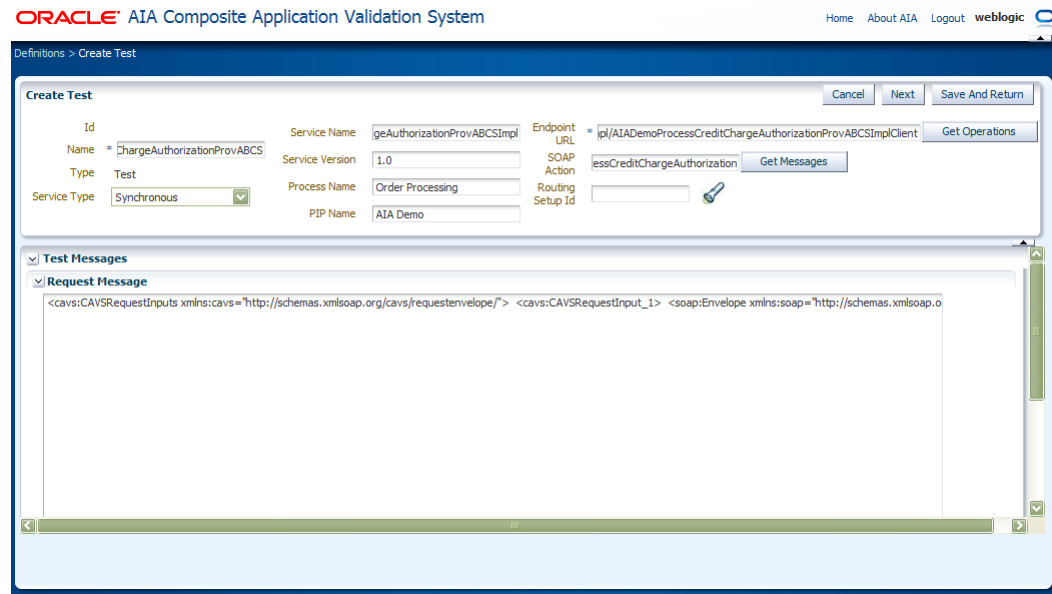
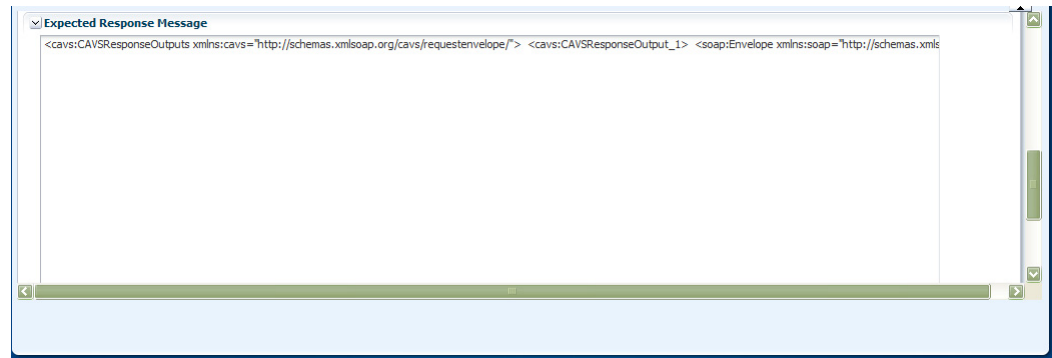


Figure 4–2 Create Test Page (2 of 2)



- On the Create Test page, use the page elements discussed in Table 4–1 to create test definitions.

Table 4–1 Create Test Page Elements

Element	Description
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: Synchronous (request-and-reply) , Notify (asynchronous request-only) , or 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.

Table 4–1 (Cont.) Create Test Page Elements

Element	Description
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	<p>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.</p> <p>Select the operation that you want to test using the test definition. The selected operation displays in the SOAP Action field.</p>
SOAP Action	<p>If you clicked Get Operations to select an operation in the Select WSDL Operations window, the selected operation displays here.</p> <p>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.</p>
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	<p>Select a routing configuration that you want to use for the test.</p> <p>For more information about routing configurations, see Chapter 8, "Defining CAVS Routing Setup IDs."</p>

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.

Elements available in the Test Messages group box are discussed in [Table 4–2](#).

For more information about obtaining request and response XML message text, see [Section 2.5.1, "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 [Section 4.3, "How to Provide Multiple Request and Response Message Sets in a Single Test Definition."](#)

Table 4–2 Create Test Page - Test Messages Group Box Elements

Element	Description
Request Message	<p>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.</p> <p>When you first access the Create Test page, the Request Message text box is populated with a SOAP stub message.</p> <p>You can use the Get Messages button to generate a request stub message based on selected endpoint URL and operation values.</p> <p>If you are manually entering your request message, the <i>Paste your SOAP Message Content here</i> 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.</p> <p>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 page 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.</p>
Expected Response Message	<p>The ability to enter response message XML text is available when the Service Type field value is set to Synchronous or Asynchronous two way.</p> <p>When you first access the Create Test page, the Expected Response Message text box is populated with a SOAP stub message.</p> <p>If you are manually entering your request message, the <i>Paste your SOAP Message Content here</i> 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.</p> <p>For test definitions with the Service Type field set to Synchronous, the response message stub will have been generated when you clicked the Get Messages button during request message generation.</p> <p>When you enter response message XML text on this page, you can click the Generate XPath button 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.</p> <p>The Expected Response Message text box is unavailable when the Service Type field value is set to Notify.</p>
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 and Return	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, as shown in [Figure 4-3](#), [Figure 4-4](#), [Figure 4-5](#), [Figure 4-6](#), and [Figure 4-7](#).

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

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

Figure 4-3 Modify Test Definition Page (1 of 5)

The screenshot displays the 'Modify Test Definition' page in the Oracle AIA Composite Application Validation System. The page header includes the Oracle logo and navigation links for Home, About AIA, Logout, and weblogic. The breadcrumb trail shows 'Definitions > Modify Test Definition'. The main form contains the following fields:

- Id:** 1082
- Service Name:** AIADemoProcessCreditCharge
- Endpoint URL:** http://sdc60025eems.us.oracle.com:8091/soa-infra/services/default
- Service Version:** 1.0
- SOAP Action:** AIADemoProcessCreditCharge
- Process Name:** Order Processing
- PIP Name:** AIA Demo

Below the form, there is a 'Test Messages' section with a 'Get Request Message' button. The text area contains the following XML message text:

```
<cavs:CAVSRquestInputs xmlns:cavs="http://schemas.xmlsoap.org/cavs/requestenvelope/"> <cavs:CAVSRquestInput_1> <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"> <soap:Header/> <soap:Body/> </soap:Envelope> </cavs:CAVSRquestInput_1> </cavs:CAVSRquestInputs>
```

Figure 4–4 Modify Test Definition Page (2 of 5)



Figure 4–5 Modify Test Definition Page (3 of 5)

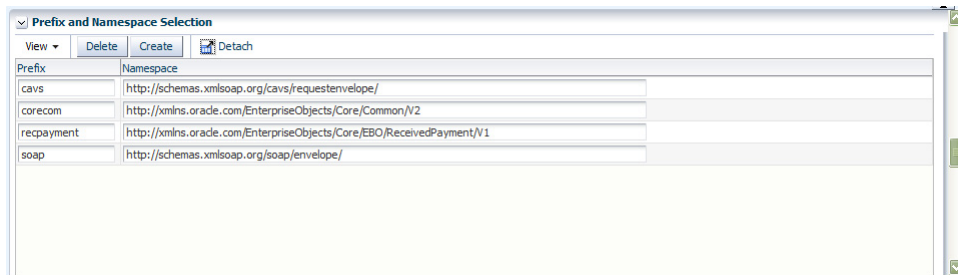


Figure 4–6 Modify Test Definition Page (4 of 5)

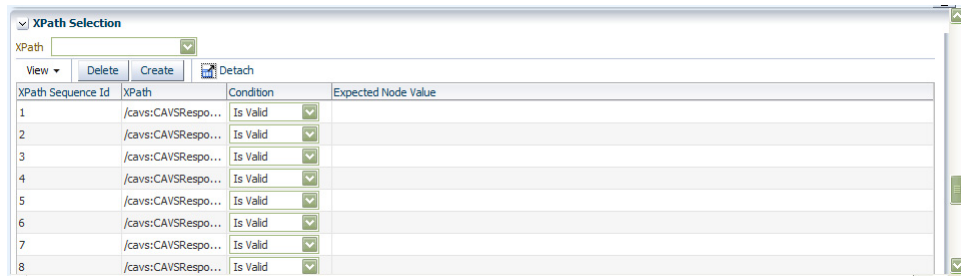
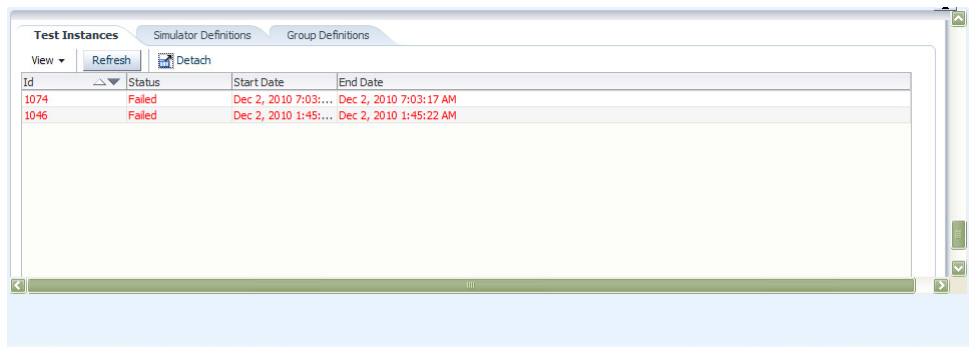


Figure 4–7 Modify Test Definition Page (5 of 5)



2. Use the 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 that have already defined for the test definition. You can modify the values in editable fields.

Most of the elements on this page also appear on the Create Test Definition page and are documented in [Section 4.1, "How to Create a Test Definition."](#) Any additional elements are discussed here.

The Time-out (msec) (in milliseconds) 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 [Section 5.1, "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.

Available Test Message group box elements are discussed in [Table 4-3](#).

For more information about obtaining request and response XML message text, see [Section 2.5.1, "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 [Section 4.3, "How to Provide Multiple Request and Response Message Sets in a Single Test Definition."](#)

Table 4-3 *Modify Definitions Page - Test Messages Group Box Elements*

Element	Description
Request Message	<p>If request message XML text was entered on the Create Test page, it is accessible and editable on this page.</p> <p>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.</p> <p>You can use the Get Messages button to generate a request stub message based on selected endpoint URL and operation values.</p> <p>If you are manually entering your request message, the <i>Paste your SOAP Message Content here</i> 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.</p>

Table 4–3 (Cont.) Modify Definitions Page - Test Messages Group Box Elements

Element	Description
Request CorrelationId Message	<p>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.</p> <p>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.</p> <p>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.</p> <p>Click Lookup to access the Choose Request Correlation Id page, where you can select a correlation ID from XPath variables available in the message.</p>
Expected Response Message	<p>The ability to enter response message XML text is available when the Service Type field value is set to Synchronous or Asynchronous two way.</p> <p>If expected response message XML test was entered on the Create Test page, it is accessible and editable on this page.</p> <p>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.</p> <p>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.</p> <p>When you enter response message XML text on this page, you can click Generate XPath button 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.</p> <p>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.</p> <p>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.</p>
Generate Xpath	<p>Click to generate namespace and XPath values based on available Endpoint URL and Response Message values.</p> <p>After you have generated XPath values, consider deleting any rows that will not be used in the testing effort.</p> <p>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.</p>

Table 4–3 (Cont.) Modify Definitions Page - Test Messages Group Box Elements

Element	Description
Response Message Correlation ID	<p>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.</p> <p>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.</p> <p>Click Lookup to access the Choose Response Correlation Id page, where you can select a correlation ID from XPath variables available in the message.</p>

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. Elements available in the Prefix and Namespace Selection grid are discussed in [Table 4–4](#).

Table 4–4 Prefix and Namespace Selection Grid Elements

Element	Description
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 generalize 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 generalize 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.

Elements available in the XPath Selection grid are discussed in [Table 4-5](#).

Table 4-5 XPath Selection Grid Elements

Element	Description
Xpath	<p>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.</p> <p>For more information about providing multiple data sets in a test definition, see Section 4.3, "How to Provide Multiple Request and Response Message Sets in a Single Test Definition."</p>
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 the Generate Xpath button.
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	<p>Select an available value:</p> <ul style="list-style-type: none"> ▪ 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 <p>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.</p>

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.

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

For more information about the Test Instance Detail page, see [Section 9.2, "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.

Available elements on the Simulator Definition tab are discussed in [Table 4–6](#).

Table 4–6 Linked Simulator Definition Selection Grid Elements

Element	Description
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. After 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. After 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. 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.

Elements available in the Group Definition Selection grid are discussed in [Table 4–7](#).

Table 4–7 Group Definition Selection Grid Elements

Element	Description
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 Chapter 7, "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. <ul style="list-style-type: none"> ■ 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. For more information about the Test Instance Details page, see Chapter 9, "Working with Test and Simulator Instances." ■ Lock: Select to lock the test definition and view the test definition on the View Test Definition page. A locked definition cannot be edited. ■ 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.

Table 4–7 (Cont.) Group Definition Selection Grid Elements

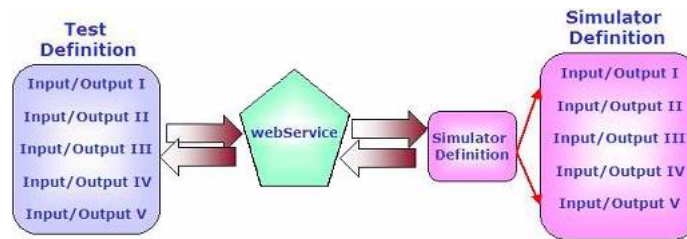
Element	Description
Apply	Click to apply and save any changes you have made to values on the page.
Save	Click to save entries on the page and go to the Definitions page. For more information about the Definitions page, see Chapter 6, "Searching for Test and Simulator Definitions."

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, as shown in [Figure 4–8](#). 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.

Figure 4–8 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 format shown in [Example 4–1](#) 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.

Example 4–1 Request Message Format

```
<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 format shown in [Example 4–2](#) to include multiple sets of response data in the test definition.

Example 4–2 Response Message Format

```

<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 [Section 4.2, "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 [Section 5.3, "How to Provide Multiple Request and Response Message Sets in a Single Simulator Definition."](#)

Creating and Modifying Simulator Definitions

This chapter describes how to create and modify simulator definitions. It also provides instructions for how to provide multiple request and response message sets in a single simulator definition, how to create a simulator definition that supports chatty services, and how to send dynamic responses in a simulator response.

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 includes the following sections:

- [Section 5.1, "How to Create a Simulator Definition"](#)
- [Section 5.2, "How to Modify a Simulator Definition"](#)
- [Section 5.3, "How to Provide Multiple Request and Response Message Sets in a Single Simulator Definition"](#)
- [Section 5.4, "How to Create a Simulator Definition that Supports Chatty Services"](#)
- [Section 5.5, "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 AIA Home Page. In the Composite Application Validation System area, click the **Go** button. Select the **Definitions** tab. Click the **Create Simulator** button. The Create Simulator page displays, as shown in [Figure 5-1](#) and [Figure 5-2](#).

Figure 5–1 Create Simulator Page (1 of 2)

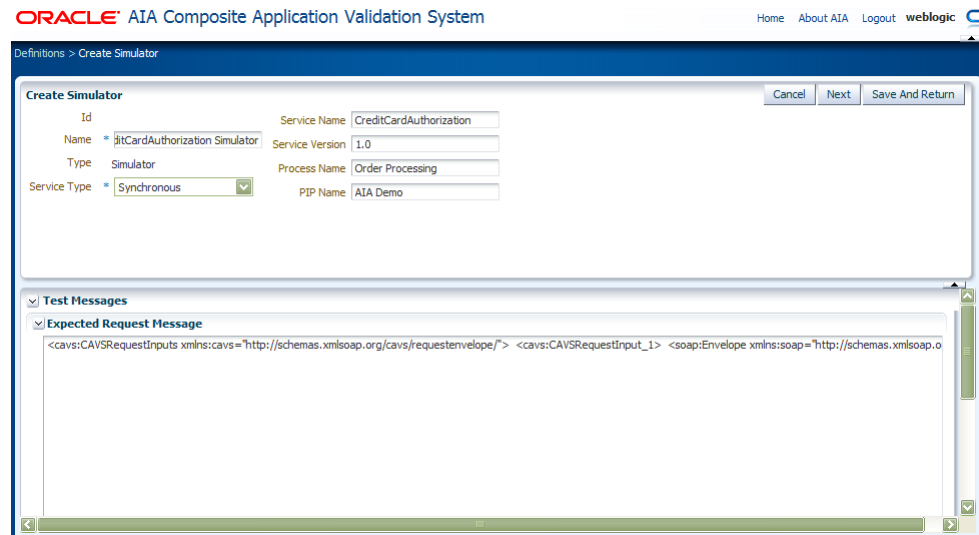
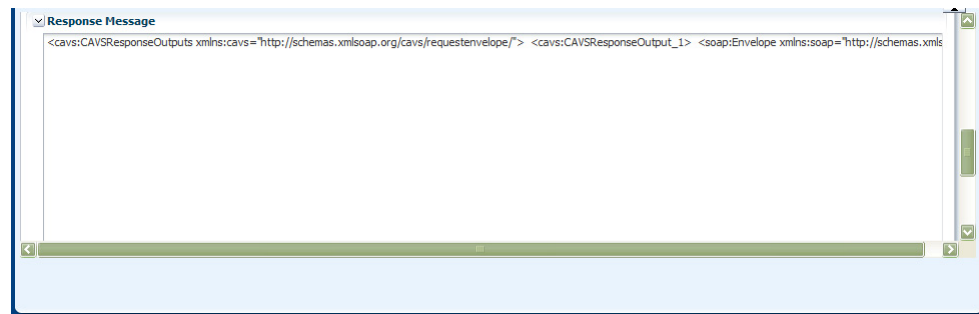


Figure 5–2 Create Simulator Page (2 of 2)



- Use the page elements on the Create Simulator page to create simulator definitions. Available elements are discussed in [Table 5–1](#).

Table 5–1 Create Simulator Page Elements

Element	Description
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. <ul style="list-style-type: none"> Synchronous (request-and-reply) Notify (asynchronous request-only) 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.

Table 5–1 (Cont.) Create Simulator Page Elements

Element	Description
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 [Section 5.3, "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 [Section 5.4, "How to Create a Simulator Definition that Supports Chatty Services."](#)

Available elements in the Test Messages group box are discussed in [Table 5–2](#).

Table 5–2 Test Messages Group Box Elements

Element	Description
Expected Request Message	<p>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.</p> <p>If a simulator has been directly designated for use in the <code>AIAConfigurationProperties.xml</code> via the Routing Configurations page, the simulator definition will be identified directly. However, after 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.</p> <p>For more information about the Routing Configurations page, see Chapter 8, "Defining CAVS Routing Setup IDs."</p> <p>You can enter expected request message XML text on this page and click the Generate XPath button 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.</p> <p>You may choose to copy and paste messages from the BPEL Console, instead of manually entering them.</p> <p>For more information, see Section 2.5.1, "How to Obtain Message XML Text from a BPEL Process."</p>

Table 5–2 (Cont.) Test Messages Group Box Elements

Element	Description
Response Message	<p>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.</p> <p>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.</p> <p>You may choose to copy and paste messages from the BPEL Console, instead of manually entering them.</p> <p>For more information, see Section 2.5.1, "How to Obtain Message XML Text from a BPEL Process."</p>
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 Simulator Definition page, as shown in [Figure 5–3](#), [Figure 5–4](#), [Figure 5–5](#), [Figure 5–6](#), and [Figure 5–7](#).

Access the page using one of the following navigation paths:

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

Figure 5–3 Modify Simulator Definition Page (1 of 5)

ORACLE AIA Composite Application Validation System Home About AIA Logout weblogic

Definitions > Modify Simulator Definition

Modify Simulator Definition Actions Cancel Save Save And Return

Id	1100	Service Name	CreditCardAuthorization
Name	* CreditCardAuthorization Simul	Service Version	1.0
Type	Simulator	Process Name	Order Processing
Service Type	Synchronous	PIP Name	AIA Demo

Test Messages
Use the Test Messages group box to enter and edit XML message text relevant to your simulator. Response message XML text is required for a synchronous service simulator.

Expected Request Message

Generate Xpath

```
<cavs:CAVSRquestInputs xmlns:cavs="http://schemas.xmlsoap.org/cavs/requestenvelope/"> <cavs:CAVSRquestInput_1> <soap:Envelope xmlns:soap="http://schemas.xmlsoap.o
```

Figure 5–4 Modify Simulator Definition Page (2 of 5)

Response Message

```
<cavs:CAVSRResponseOutputs xmlns:cavs="http://schemas.xmlsoap.org/cavs/requestenvelope/"> <cavs:CAVSRResponseOutput_1> <soap:Envelope xmlns:soap="http://schemas.xmls
```

Figure 5–5 Modify Simulator Definition Page (3 of 5)

Prefix and Namespace Selection

View Delete Create Detach

Prefix	Namespace
No rows yet.	

Figure 5–6 Modify Simulator Definition Page (4 of 5)

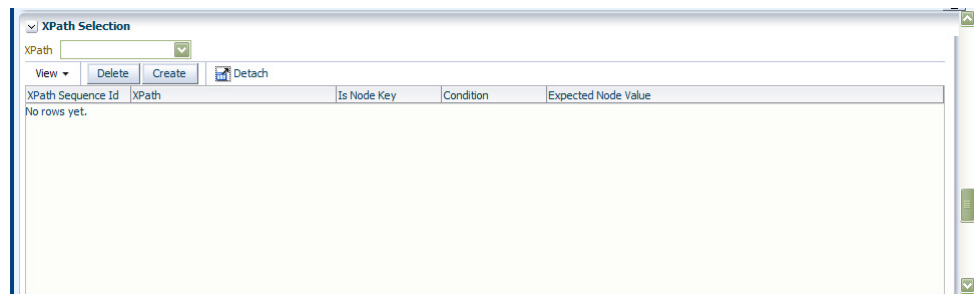
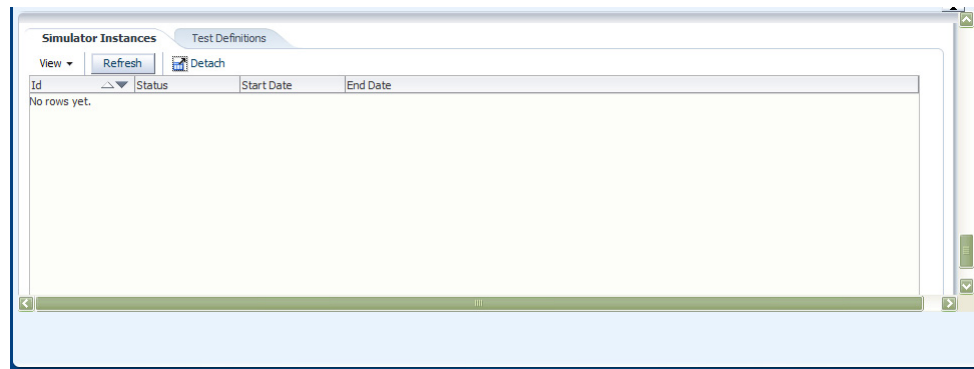


Figure 5–7 Modify Simulator Definition Page (5 of 5)



2. Use the 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.

Most of the elements on this page also appear on the Create Simulator Definition page and are documented in [Section 5.1, "How to Create a Simulator Definition."](#) Any additional elements are discussed here in [Table 5–3.](#)

Table 5–3 Modify Simulator Definition Page Elements

Element	Description
Actions	<p>Select the action you want to take with the simulator definition.</p> <ul style="list-style-type: none"> ▪ 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	<p>Click to save entries on the page and go to the Definitions page.</p> <p>For more information, see Chapter 6, "Searching for Test and Simulator Definitions."</p>

Table 5–3 (Cont.) Modify Simulator Definition Page Elements

Element	Description
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)	<p>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.</p> <p>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 Time-out (msec) value defined for any test definition.</p> <p>For more information about the Time-out (msec) field, see Section 4.2, "How to Modify a Test Definition."</p>

Test Messages

For more information about the elements in the Test Messages group box, see [Section 5.1, "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.

Available elements in the Prefix and Namespace Selection grid are discussed in [Table 5–4](#).

Table 5–4 Prefix and Namespace Selection Grid Elements

Element	Description
Delete	<p>Select one or more namespace rows and click Delete to execute the deletion.</p> <p>This button only appears when namespace rows are present.</p>
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.

Available elements in the XPath Selection grid are discussed in [Table 5–5](#).

Table 5–5 XPath Selection Grid Elements

Element	Description
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 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 the Generate Xpath button.
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	Select the condition you want to use: <ul style="list-style-type: none"> ▪ 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 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 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.

Available elements in the Simulator Instance Selection grid are discussed in [Table 5–6](#).

Table 5–6 Simulator Instance Selection Grid

Element	Description
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 Section 9.3, "How to View Simulator Instance Details."

Table 5–6 (Cont.) Simulator Instance Selection Grid

Element	Description
Status	<p>Displays the status of the simulator instance generated by the simulator definition.</p> <ul style="list-style-type: none"> ■ 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.
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.

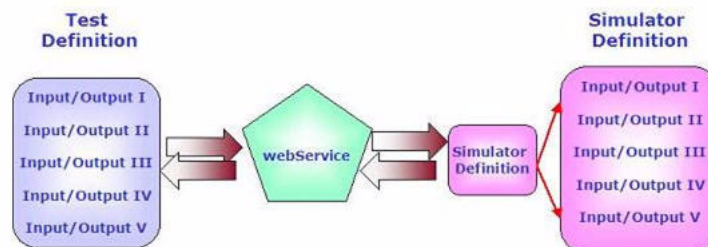
Available elements in the Linked Test Definition Selection grid are discussed in [Table 5–7](#).

Table 5–7 Linked Test Definition Selection Grid

Element	Description
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, as shown in [Figure 5–8](#). This means that simulator definitions only need to be created per usage requirements, not per test data requirements.

Figure 5–8 Providing Multiple Request and Response Message Sets in a Single Simulator Definition

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.

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 format provided in [Example 5–1](#) to include multiple sets of request data in the simulator definition.

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

Example 5–1 Request Message Format

```
<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 format shown in [Example 5–2](#) to include multiple sets of response data in the simulator definition.

Example 5–2 Response Message Format

```
<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 [Section 5.2, "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 [Section 4.3, "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 format shown in [Example 5–3](#) 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.

Example 5–3 Request Message Format

```
<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>
```

After you have provided request and response messages, click the **Generate XPath** button 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 [Section 5.2, "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 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:Envelope/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 nodes shown in [Example 5-4](#):

Example 5-4 Request SOAP Message 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 shown in [Example 5-5](#):

Example 5-5 Response SOAP Message

```
<corecom:PersonName>
  <corecom:FirstName>##XPATH. {/soap:Envelope/soap:Body/corecom:CreateCustomerParty
    ListEBM/ebo:DataArea/ebo:CreateCustomerPartyList/
    corecomx:Contact/corecomx:PersonName/corecomx:FamilyName}##2dot1</corecom:FirstNa
    me>
  <corecom:MiddleName>##XPATH. {/soap:Envelope/soap:Body/corecom:CreateCustomerParty
    ListEBM/ebo:DataArea/ebo:CreateCustomerPartyList/corecomx:Contact/corecomx:Person
    Name/corecomx:MiddleName}##</corecom:MiddleName>
  <corecom:FamilyName>##XPATH. {/soap:Envelope/soap:Body/corecom:CreateCustomerParty
    ListEBM/ebo:DataArea/ebo:CreateCustomerPartyList/corecomx:Contact/corecomx:Person
    Name/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 shown in [Example 5-6](#).

Example 5–6 Response Message Modified by CAVS

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

Searching for Test and Simulator Definitions

This chapter describes how to search for and work with test and simulator definitions.

This chapter includes the following section [Section 6.1, "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 Oracle Application Integration Architecture (AIA) Home Page. In the Composite Application Validation System area, click the **Go** button. Select the **Definitions** tab. The **Definitions** page displays, as shown in [Figure 6-1](#) and [Figure 6-2](#).

Figure 6-1 Definitions Page (1 of 2)

The screenshot displays the Oracle AIA Composite Application Validation System interface. The top navigation bar includes "ORACLE AIA Composite Application Validation System" and links for "Home", "About AIA", "Logout", and "weblogic". The main content area is titled "Definitions" and contains a "Search Definitions" section with input fields for Id, Name, Type, Service Name, Service Version, Process Name, PIP Name, Endpoint URL, SOAP Action, and State. Below the search fields is a "Search" button. The "Search Result Selection" section features a toolbar with buttons for "Execute", "Delete", "Duplicate", "Lock", "Unlock", "Export", "Change URL", "Create Test", "Create Simulator", "Import", and "Detach". A table lists the search results with columns for Id, Name, Type, State, Service Type, Service Name, and Service Version.

Id	Name	Type	State	Service Type	Service Name	Service Version
1000	AIAB2BInterface EDI X12 855 Simulator	Simulator	🔒	Notify	AIAB2BInterface	1.0
1001	AIADemoProcessSalesOrderOrchestrationEBS Simulator	Simulator	🔒	Notify	AIADemoProcessSalesOrderOrchestrationEBS	1.0
1002	StoreFrontService Update Order Simulator	Simulator	🔒	Synchronous	StoreFrontService	1.0
1003	InternalWarehouse Simulator	Simulator	🔒	Synchronous	InternalWarehouse	1.0
1004	Shipment (USPS and Batch) Simulator	Simulator	🔒	Notify	USPSShipmentService / AIADemoBatchJMSAdapter	1.0
1005	PartnerSupplier Simulator	Simulator	🔒	Synchronous	PartnerSupplier	1.0
1006	AIADemoSyncCustomerPartyListCRMPProvDBAdapter Sim...	Simulator	🔒	Notify	AIADemoSyncCustomerPartyListCRMPProvDBAdapter	1.0
1007	AIADemoSyncCustomerPartyListEBS Simulator	Simulator	🔒	Notify	AIADemoSyncCustomerPartyListEBS	1.0
1008	CreditCardAuthorization Simulator	Simulator	🔒	Synchronous	CreditCardAuthorization	1.0
1009	AIADemoProcessSalesOrderShopReqABCS	Test	🔒	Notify	AIADemoProcessSalesOrderShopReqABCSImpl	1.0
1010	AIADemoQueryCustomerPartyCRMPProvABCS	Test	🔒	Synchronous	AIADemoQueryCustomerPartyCRMPProvABCSImpl	1.0

Figure 6–2 Definitions Page (2 of 2)

Process Name	PIP Name	Endpoint URL
B2B Order Processing	AIA Demo	
Order Processing	AIA Demo	
Order Processing	AIA Demo	
Order Processing	AIA Demo	
Order Processing	AIA Demo	
Order Processing	AIA Demo	
Order Processing	AIA Demo	
Customer Synchronization	AIA Demo	
Customer Synchronization	AIA Demo	
Order Processing	AIA Demo	
Order Processing	AIA Demo	http://sic6003zsems.us.oracle.com:8085/soa-infra/s

- Use the 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.

Available elements in the Search Definitions group box are discussed in [Table 6–1](#).

Table 6–1 Search Definitions Group Box Elements

Element	Description
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 for which you are searching: <ul style="list-style-type: none"> ■ <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: <ul style="list-style-type: none"> ■ <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.

Table 6–1 (Cont.) Search Definitions Group Box Elements

Element	Description
State	Select the state of the definition: <ul style="list-style-type: none"> ▪ <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.

Available elements in the Search Result Selection grid are discussed in [Table 6–2](#).

Table 6–2 Search Result Selection Grid Elements

Element	Description
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 Section 9.2, "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 Chapter 12, "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 Section 4.1, "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 Section 5.1, "How to Create a Simulator Definition."

Table 6–2 (Cont.) Search Result Selection Grid Elements

Element	Description
Import	For more information about importing test definitions, see Chapter 12, "Exporting and Importing CAVS Definitions and Instances."
Id	<p>Click for an unlocked test definition to access the Modify Test Definition page.</p> <p>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.</p> <p>For more information, see Section 4.2, "How to Modify a Test Definition."</p> <p>Click for an unlocked simulator definition to access the Modify Simulator Definition page.</p> <p>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.</p> <p>For more information, see Section 5.2, "How to Modify a Simulator Definition."</p>

Working with Group Definitions

This chapter describes how to create, modify, and work with group definitions.

This chapter includes the following sections:

- [Section 7.1, "How to Work with Group Definitions"](#)
- [Section 7.2, "How to Create and Modify a Group Definition"](#)

7.1 How to Work with Group Definitions

To work with group definitions:

1. 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. The Group Definitions page displays as shown in [Figure 7-1](#).

Figure 7-1 Group Definitions Page

The screenshot shows the Oracle AIA Composite Application Validation System interface. The top navigation bar includes 'Home', 'About AIA', 'Logout', and 'weblogic'. The main content area has tabs for 'Definitions', 'Instances', 'Group Definitions', 'Group Instances', and 'Routing Setup'. The 'Group Definitions' tab is active, displaying a 'Search Group Definitions' form with input fields for 'Id', 'Name', 'Process Name', and 'PIP Name', and a 'Search' button. Below the search form is a 'Search Result Selection' section with a table of results.

Id	Name	Process Name	PIP Name
1021	D6789	Order Processing	AIA Demo

2. Use the 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.

Available elements in the Search Group Definitions group box are discussed in [Table 7-1](#).

Table 7–1 Search Group Definitions Group Box Elements

Element	Description
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.

Available elements in the Search Result Selection grid are discussed in [Table 7–2](#).

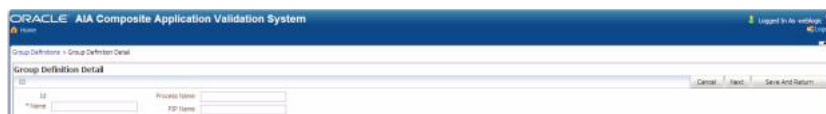
Table 7–2 Search Result Selection Grid Elements

Element	Description
Execute	Select one or more group definitions that you want to run and click Execute to execute the group definition. When a group definition has successfully executed, you can view details of the group instance on the Group Instances Detail page. For more information about the Group Instances Detail page, see Section 10.2, "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. 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 a group definition:

1. 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. Click the **Create** button. The Group Definition Detail page for a new group definition displays as shown in [Figure 7–2](#).

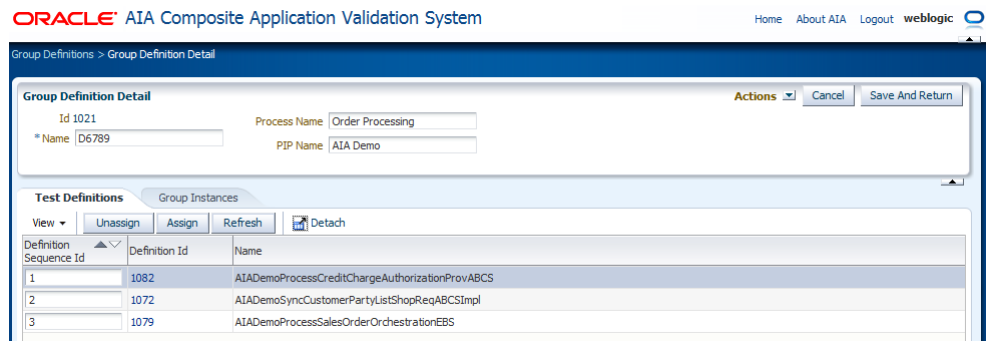
Figure 7–2 Group Definition Detail Page (New Group Definition)

2. Use the page elements on the Group Definition Detail page, as discussed in [Table 7-3](#), to create a group definition that combines one or more tests and executes them in a single-threaded batch sequence.
3. Click **Next** to save entries and display further group definition details elements, discussed in [Table 7-3](#), on the Group Definition Detail page for the newly created and existing group definitions.

To modify a group definition:

1. 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. Select a group definition **Id** link. The Group Definition Detail page for existing group definitions displays as shown in [Figure 7-3](#).

Figure 7-3 Group Definition Detail Page (Existing Group Definition)



2. Use the page elements on the Group Definition Detail page, as discussed in [Table 7-3](#), to modify an existing group definition.

Table 7-3 Group Definition Detail Page Elements for New and Existing Definitions

Element	Description
Actions	Select the action you want to take with the group definition. <ul style="list-style-type: none"> ■ Execute: Select to execute the group definition. When a group definition has successfully executed, you can view details of the test instance on the Group Instances page. ■ 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. 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.

Table 7–3 (Cont.) Group Definition Detail Page Elements for New and Existing

Element	Description
Id	Upon saving a new group definition, a unique key identifier is assigned to the group definition. 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. 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. 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. 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.

Available elements in the Test Definition Selection grid are discussed in [Table 7–4](#).

Table 7–4 Test Definition Selection Grid Elements

Element	Description
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. 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 Section 4.2, "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.

Available elements in the Group Instances Selection grid are discussed in [Table 7–5](#).

Table 7–5 Group Instance Selection Grid Elements

Element	Description
Refresh	Click to refresh the Group Definition Detail page.

Table 7-5 (Cont.) Group Instance Selection Grid Elements

Element	Description
Id	Click to access the Group Instances Detail page.
Start Date	Displays the date and time at which the group instance was initiated.

Defining CAVS Routing Setup IDs

This chapter provides an introduction to CAVS routing setup IDs, and how to create, search, and modify them. It also provides information about setting up routing configurations without creating routing setup IDs.

Composite Application Validation System (CAVS) routing setups are used when CAVS test definitions call services that in turn, call CAVS simulators and when actual applications and services call CAVS simulators instead of calling subsequent actual services.

This chapter includes the following sections:

- [Section 8.1, "Introduction to CAVS Routing Setup IDs"](#)
- [Section 8.2, "How to Create CAVS Routing Setup IDs"](#)
- [Section 8.3, "How to Search for CAVS Routing Setup IDs"](#)
- [Section 8.4, "How to Modify Routing Setup IDs"](#)
- [Section 8.5, "How to Set Up CAVS Routing Configurations Without Creating Routing Setup IDs"](#)

8.1 Introduction to CAVS Routing Setup IDs

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 run time 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.

- 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.

- 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

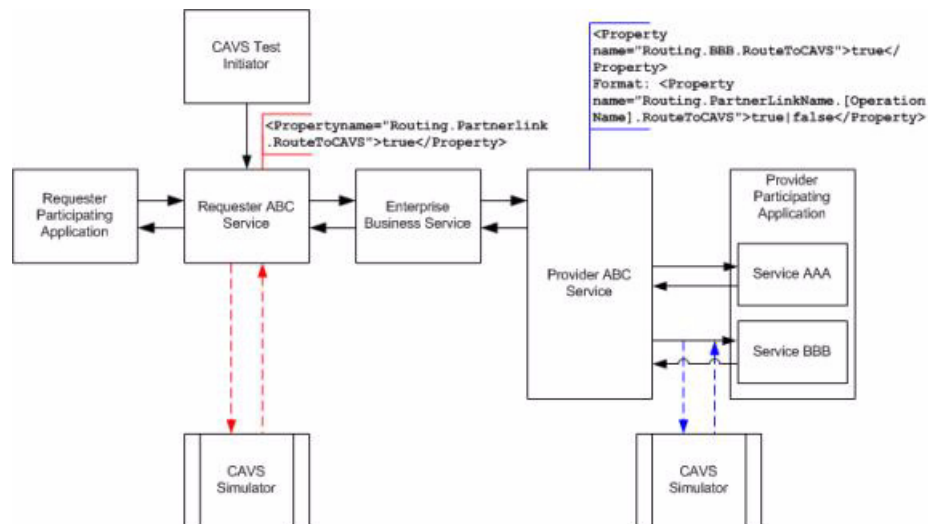
application service. For this scenario, create a routing setup ID to set the `RouteToCAVS` property to `TRUE` on the provider ABCS. This will ensure that the message is routed to the CAVS simulator, as indicated in blue.

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.

- 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.

Figure 8–1 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 enables 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.

Figure 8–1 Sample Scenarios for Using CAVS Routing Setup IDs



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 [Section 4.1, "How to Create a Test Definition."](#)

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

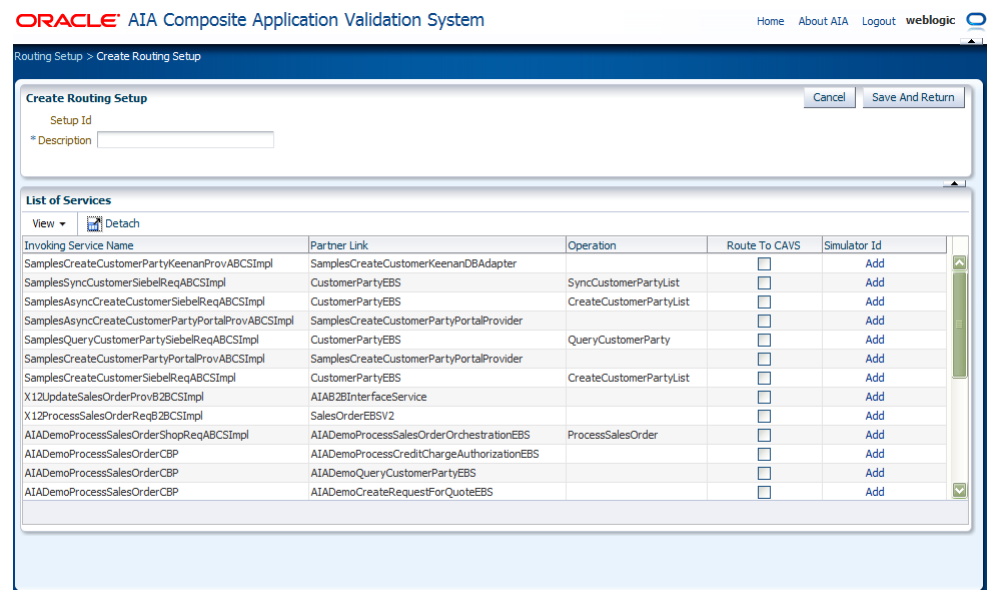
For more information about the Configuration page, see [Section 8.5, "How to Set Up CAVS Routing Configurations Without Creating Routing Setup IDs."](#)

8.2 How to Create CAVS Routing Setup IDs

To create CAVS routing setup IDs:

1. Access the 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. The Create Routing Setup page displays as shown in [Figure 8–2](#).

Figure 8–2 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 [Section 4.1, "How to Create a Test Definition."](#)

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

For more information about how to quickly define a routing configuration that is stored in AIAConfigurationProperties.xml, see [Section 8.5, "How to Set Up CAVS Routing Configurations Without Creating Routing Setup IDs."](#)

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

Available elements on the Create Routing Setup page are discussed in [Table 8-1](#).

Table 8-1 Create Routing Setup Page Elements

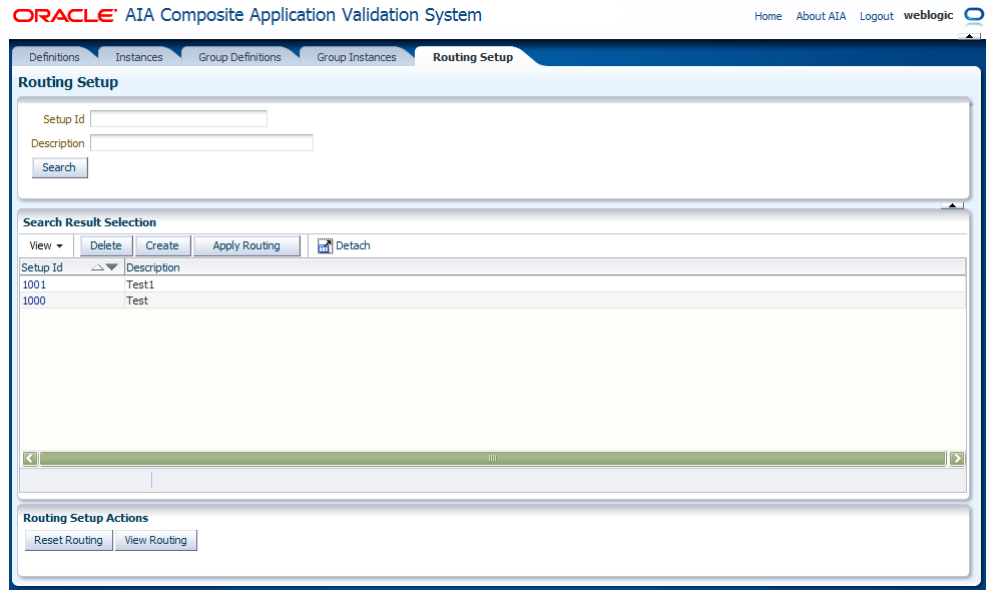
Element	Description
SetupId	Upon saving, a sequentially generated ID is assigned to the routing setup ID.
Description	Enter a description of the routing setup ID you are creating.
InvokingServiceName	Lists all services defined in the AIAConfigurationProperties.xml file in <AIA_HOME>/aia_instances/\$INSTANCE_NAME/AIAMetaData/config.
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 AIA Home Page. In the Composite Application Validation System area, click the **Go** button. Select the **Routing Setup** tab. The Routing Setup page displays, as shown in [Figure 8-3](#).

Figure 8–3 Routing Setup Page



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

Available elements on the Routing Setup page are discussed in [Table 8–2](#).

Table 8–2 Routing Setup Page Elements

Element	Description
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 Section 8.2, "How to Create CAVS Routing Setup IDs."
Apply Routing	After 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 Section 8.4, "How to Modify Routing Setup IDs."

Routing Setup Actions

Available elements in the Routing Setup Actions area are discussed in [Table 8–3](#).

Table 8–3 Routing Setup Actions Area Elements

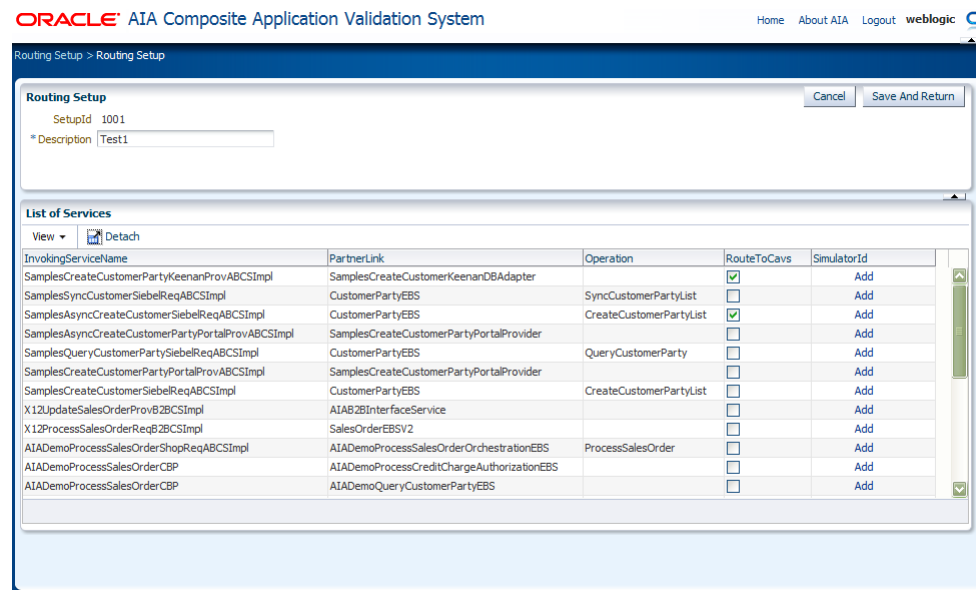
Element	Description
Reset Routing	Click to set all routing configurations to <i>FALSE</i> . This means that all routings to simulators (RouteToCAVS property settings) in the AIAConfigurationProperties.xml file will be set to <i>FALSE</i> , whether you have defined them through the Routing Setup pages 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 AIA Home Page. In the Composite Application Validation System area, click the **Go** button. Select the **Routing Setup** tab. Click a **SetupId** link. The Routing Setup page displays as shown in [Figure 8–4](#).

Figure 8–4 Routing Setup Page



2. Use the page elements on the Routing Setup page to modify existing routing setups. Available elements are discussed in [Table 8–4](#).

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

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 [Section 8.3, "How to Search for CAVS Routing Setup IDs."](#)

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 [Section 8.2, "How to Create CAVS Routing Setup IDs."](#)

Working with Test and Simulator Instances

This chapter describes how to work with test and simulator instances, how to view test instance details, and how to view simulator instance details.

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 includes the following sections:

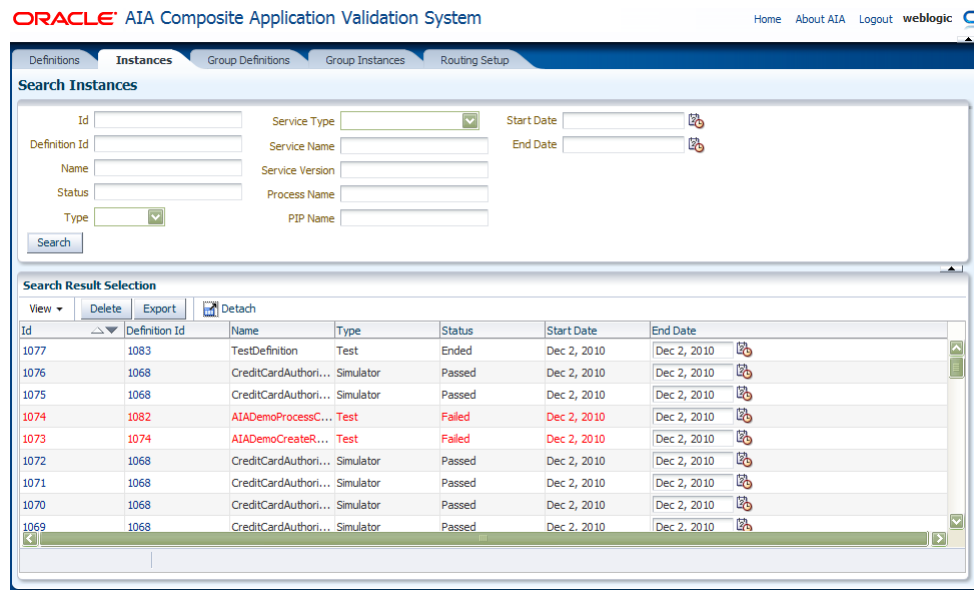
- [Section 9.1, "How to Work with Test and Simulator Instances"](#)
- [Section 9.2, "How to View Test Instance Details"](#)
- [Section 9.3, "How to View Simulator Instance Details"](#)

9.1 How to Work with Test and Simulator Instances

To work with test and simulator instances:

1. Access the Oracle Application Integration Architecture (AIA) Home Page. In the Composite Application Validation System area, click the **Go** button. Select the **Instances** tab. The Instances page displays, as shown in [Figure 9-1](#).

Figure 9–1 Instances Page



- Use the 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.

Available elements in the Search Instances group box are discussed in [Table 9–1](#).

Table 9–1 Search Instances Group Box Elements

Element	Description
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. <ul style="list-style-type: none"> ■ <i>Ended</i>: This status is only applicable to instances that do not involve validations. Indicates that the instance has ended. ■ <i>Faulted</i>: The instance could not execute properly due to exceptions or faults. ■ <i>Failed</i>: The instance did not pass validation. ■ <i>Passed</i>: The instance passed validation. ■ <i>Delayed</i>: For an asynchronous two-way test instance, indicates that the test instance is still active and waiting for an asynchronous reply.

Table 9–1 (Cont.) Search Instances Group Box Elements

Element	Description
Type	Select the type of instance for which you are searching. <ul style="list-style-type: none"> ▪ <Value Not Selected> ▪ Test ▪ Simulator
Service Type	Select the business service pattern of the web service associated with the instance. 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. <ul style="list-style-type: none"> ▪ <Value Not Selected> ▪ Synchronous ▪ Notify ▪ 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.

Available elements in the Search Result Selection grid are discussed in [Table 9–2](#).

Table 9–2 Search Result Selection Grid Elements

Element	Description
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 Chapter 12, "Exporting and Importing CAVS Definitions and Instances."
Id	Click for a test instance to access the Test Instance Detail page.

Table 9–2 (Cont.) Search Result Selection Grid Elements

Element	Description
Definition Id	<p>Click for a simulator instance to access the Simulator Instance Detail page.</p> <p>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.</p> <p>For more information, see Section 4.2, "How to Modify a Test Definition."</p> <p>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.</p> <p>For more information, see Section 5.2, "How to Modify a Simulator Definition."</p>

9.2 How to View Test Instance Details

To view test instance details:

1. Access the Test Instances Detail page, as shown in [Figure 9–2](#) and [Figure 9–3](#).

To access the page, use one of the following navigation paths:

- 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. The Modify Test Definition page displays. Select **Execute** in the **Actions** drop-down list box.
- 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. The Modify Test Definition page displays. Click an instance **Id** link in the **Test Instances** group box.
- 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.

Figure 9–2 Test Instances Detail Page (1 of 4)

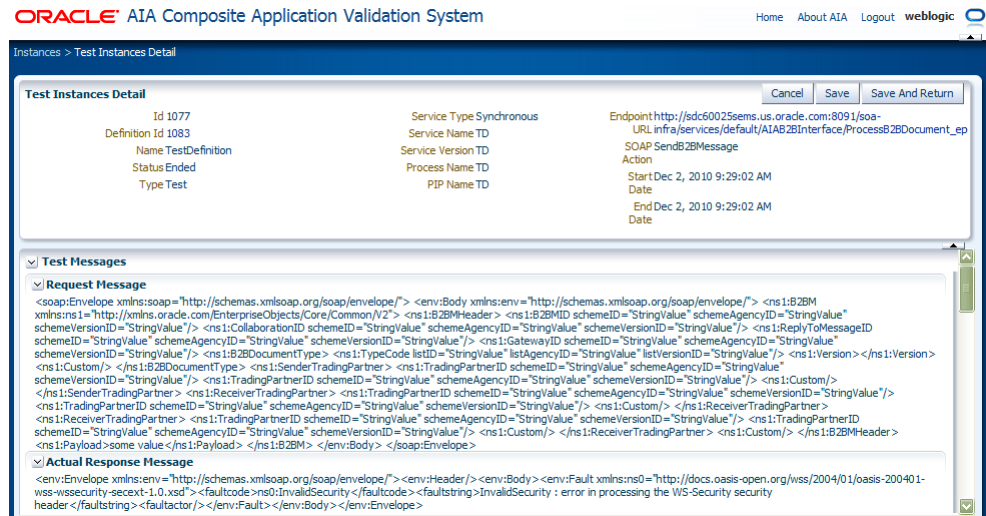


Figure 9–3 Test Instances Detail Page (2 of 4)



Figure 9–4 Test Instances Detail Page (3 of 4)

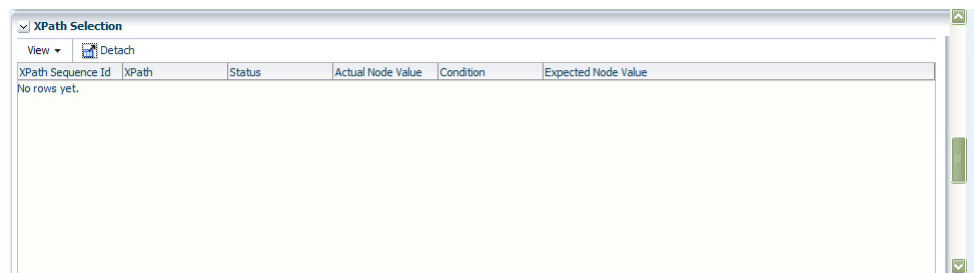
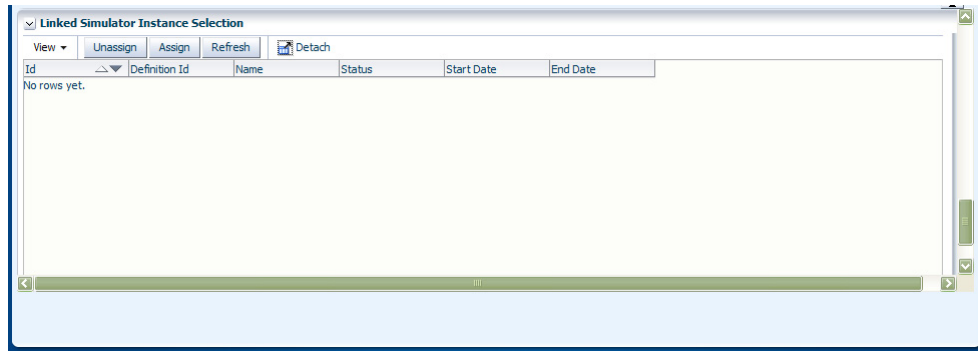


Figure 9–5 Test Instances Detail Page (4 of 4)



2. Use the page elements on the Test Instances Detail page, as discussed in [Table 9–3](#), to view the details of a test instance.

Table 9–3 Test Instances Detail Page Elements

Element	Description
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	<p>Displays the ID of the test definition that generated the test instance.</p> <p>Click for an unlocked test definition to access the Modify Test Definition page.</p> <p>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.</p> <p>For more information, see Section 4.2, "How to Modify a Test Definition."</p>
Name	Displays the descriptive name associated with the test definition that generated the instance.
Status	<p>Displays the status of the test instance.</p> <ul style="list-style-type: none"> ■ Ended: This status is only applicable to test instances that do not involve validations. Indicates that the instance has ended. ■ Faulted: The test instance could not execute properly due to exceptions or faults. ■ Failed: The test instance did not pass validation. ■ Passed: The instance passed validation. ■ 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 .

Table 9–3 (Cont.) Test Instances Detail Page Elements

Element	Description
Service Type	Displays the business service pattern of the web service tested by the test definition that generated the test instance. <ul style="list-style-type: none"> ■ Synchronous ■ Notify ■ 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 elements in the Test Messages group box, as discussed in [Table 9–4](#), to view the request and response XML messages associated with the test definition that generated the instance.

Table 9–4 Test Messages Group Box Elements

Element	Description
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 Section 4.1, "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 Section 4.1, "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 [Section 4.2, "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 [Section 4.2, "How to Modify a Test Definition."](#)

Linked Simulator Instance Selection

Use the elements in Linked Simulator Instance Selection grid, as discussed in [Table 9-5](#), 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.

Table 9-5 *Linked Simulator Instance Selection Grid Elements*

Element	Description
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. After 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 Section 4.2, "How to Modify a Test Definition."

9.3 How to View Simulator Instance Details

To view simulator instance details:

1. Access the AIA Home Page. In the Composite Application Validation System area, click the **Go** button. Select the **Instances** tab. Click the **Instance Id** link for a simulator instance. The Simulator Instances Detail page displays, as shown in [Figure 9-6](#), [Figure 9-7](#), [Figure 9-8](#), and [Figure 9-9](#).

Figure 9–6 Simulator Instances Detail Page (1 of 4)

Simulator Instances Detail

Id 1075 Service Type Synchronous Start Date Dec 2, 2010 7:50:17 AM
 Definition Id 1068 Service Name CreditCardAuthorization End Date Dec 2, 2010 7:50:17 AM
 Name CreditCardAuthorization Simulator Service Version 1.0
 Status Passed Process Name Order Processing
 Type Simulator PIP Name AIA Demo

Test Messages

Actual Request Message

```
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsa="http://www.w3.org/2005/08/addressing" >
<env:Header >
<wsa:To>http://sd60025sems.us.oracle.com:8091/AIAValidationSystemServlet/syncresponsesimulator?
smid=1068</wsa:To>
<wsa:Action>http://www.globalcompany.example.com/ns/CreditAuthorizationService/AuthorizeCredit</wsa:Action>
<wsa:MessageID>urn:DC650120FE2B11DF8F3ABDC805C357A3</wsa:MessageID>
</env:Header>
<env:Body >
<ns:tracking.requestEnvelope xmlns:ns="http://www.globalcompany.example.com/ns/tracking/1.0" >
<ns:tracking.conversationId>urn:DC650120FE2B11DF8F3ABDC805C357A3</ns:tracking.conversationId>
<ns:tracking.parentComponentInstanceId>urn:DC650120FE2B11DF8F3ABDC805C357A3</ns:tracking.parentComponentInstanceId>
<ns:tracking.compositeInstanceCreatedTime>2010-12-01T08:43:19.752-08:00</ns:tracking.compositeInstanceCreatedTime>
<wsa:ReferenceParameters>
<types:CCNumber>123456765432</types:CCNumber>
<types:PurchaseAmount>191.93</types:PurchaseAmount>
</wsa:ReferenceParameters>
</ns:tracking.requestEnvelope>
</env:Body>
</env:Envelope>
```

Response Message

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" >
<soap:Body >
<auth:status xmlns:auth="http://www.globalcompany.example.com/ns/CCAAuthorizationService">APPROVED</auth:status>
</soap:Body>
</soap:Envelope>
```

Prefix and Namespace Selection

Prefix	Namespace
auth	http://www.globalcompany.example.com/ns/CCAAuthorizationService
cavs	http://schemas.xmlsoap.org/cavs/requestenvelope/
soap	http://schemas.xmlsoap.org/soap/envelope/

Figure 9–7 Simulator Instances Detail Page (2 of 4)

Prefix and Namespace Selection

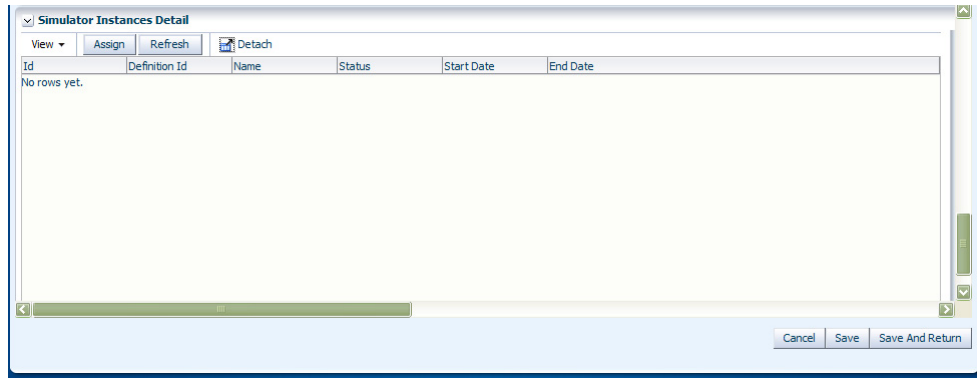
Prefix	Namespace
auth	http://www.globalcompany.example.com/ns/CCAAuthorizationService
cavs	http://schemas.xmlsoap.org/cavs/requestenvelope/
soap	http://schemas.xmlsoap.org/soap/envelope/

Figure 9–8 Simulator Instances Detail Page (3 of 4)

XPath Selection

XPath Sequence Id	XPath	Is Key Node	Status	Actual Node Value	Condition	Expected Node Value
3	/soap:Envelope		Passed		Is Valid	
4	/soap:Envelope/so...		Passed		Is Valid	
5	/soap:Envelope/so... Yes		Passed		Is Valid	
6	/soap:Envelope/so...		Passed		Is Valid	AMEX
7	/soap:Envelope/so...		Passed		Is Valid	54343454334

Figure 9–9 Simulator Instances Detail Page (4 of 4)



2. Use the page elements on the Simulator Instances Detail page, as discussed in Table 9–6, to view the details of a simulator instance.

Table 9–6 Simulator Instances Detail Page Elements

Element	Description
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. <ul style="list-style-type: none"> ■ 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. <ul style="list-style-type: none"> ■ Synchronous ■ Notify ■ Asynchronous two way

Table 9–6 (Cont.) Simulator Instances Detail Page Elements

Element	Description
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 elements in the Test Messages group box, as discussed in [Table 9–7](#), to view the request and response XML messages associated with the simulator definition that generated the instance.

Table 9–7 Test Messages Group Box Elements

Element	Description
Actual Request Message	Displays request message XML defined for the simulator definition that generated the instance. For more information about the Request Message field, see Section 5.1, "How to Create a Simulator Definition."
Response Message	Displays response message XML defined for the simulator definition that generated the instance. For more information about the Response Message field, see Section 5.1, "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 [Section 5.2, "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 [Section 5.2, "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 elements on the page, which are discussed in [Table 9–8](#).

Table 9–8 Linked Test Instance Selection Page Elements

Element	Description
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. After 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. 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. For more information, see Section 4.2, "How to Modify a Test Definition."

Working with Group Instances

This chapter describes how to view group instances and how to view group instance details.

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

This chapter includes the following sections:

- [Section 10.1, "How to View Group Instances"](#)
- [Section 10.2, "How to View Group Instance Details"](#)

10.1 How to View Group Instances

To view group instances:

1. 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. The Group Instances page displays, as shown in [Figure 10–1](#).

Figure 10–1 Group Instances Page

ORACLE AIA Composite Application Validation System

Home About AIA Logout weblogic

Definitions Instances Group Definitions **Group Instances** Routing Setup

Search Group Instances

Id Process Name

Group Definition Id PIP Name

Name Start Date

Search

Search Result Selection

View

Id	Group Definition Id	Name	Process Name	PIP Name	Start Date
1001	1001	Test	Test	Test	Dec 7, 2010
1000	1021	D6789	Order Processing	AIA Demo	Dec 7, 2010

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

Search Group Instances

Use the elements in the Search Group Instances group box, as discussed in [Table 10–1](#), to enter search criteria to find the group instance you are searching for.

Table 10–1 Search Group Instances Group Box Elements

Element	Description
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 elements in the Search Result Selection grid, as discussed in [Table 10–2](#), to work with group instances returned in your search results. Upon accessing this page, the grid is populated by all group instances.

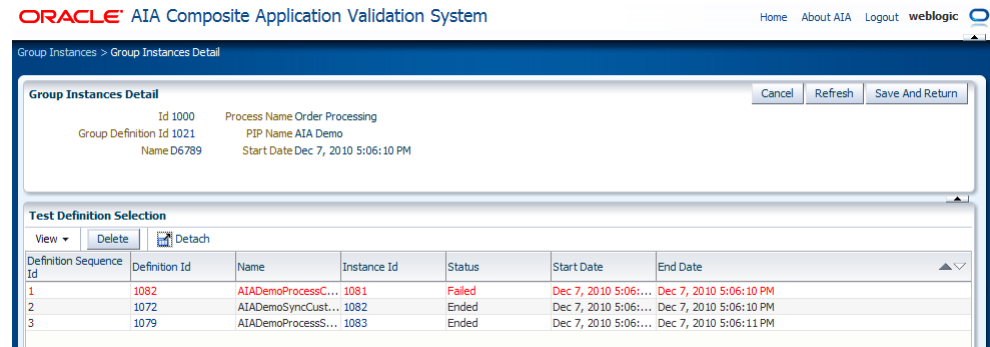
Table 10–2 Search Result Selection Grid Elements

Element	Description
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 Chapter 12, "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 Chapter 7, "Working with Group Definitions."

10.2 How to View Group Instance Details

To view group instance details:

1. Access the AIA Home Page. In the Composite Application Validation System area, click the **Go** button. Select the **Group Instances** tab. Click a group instance **Id** link on the Group Instances page. The Group Instances Detail page displays as shown in [Figure 10–2](#).

Figure 10–2 Group Instances Detail Page

- Use the page elements on the Group Instances Detail page, as discussed in [Table 10–3](#), to view the details of a group instance.

Table 10–3 Group Instances Detail Page Elements

Element	Description
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. For more information about the Modify Test Definition page, see Section 4.2, "How to Modify a Test Definition."
Instance Id	Click to access the Test Instances Detail page. For more information about the Test Instances Detail page, see Section 9.2, "How to View Test Instance Details."
Status	Displays the status of the test instance in the group instance. <ul style="list-style-type: none"> ■ Initiated: The test instance has been initiated. ■ Ended: This status is only applicable to test instances that do not involve validations. Indicates that the instance has ended. ■ Faulted: The test instance could not execute properly due to exceptions or faults. ■ Failed: The test instance did not pass validation. ■ 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.

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

This chapter describes how to purge Composite Application Validation System (CAVS)-related cross-reference entries to allow test scenarios to be rerun.

This chapter includes the following sections:

- [Section 11.1, "Introduction to Purging CAVS-Related Cross-Reference Entries"](#)
- [Section 11.2, "How to Purge CAVS-Related Cross-Reference Entries to Enable Rerunning of Test Scenarios"](#)

11.1 Introduction to Purging CAVS-Related Cross-Reference Entries

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.

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 after the test scenario has been executed. This enables rerunning of the test scenario.

11.2 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 Systems 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 Systems page, see "Building AIA Integration Flows" in *Oracle Fusion Middleware Developer's Guide for Oracle Application Integration Architecture Foundation Pack*.

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 Systems 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 "Building AIA Integration Flows" in *Oracle Fusion Middleware Developer's Guide for Oracle Application Integration Architecture Foundation Pack*.
 - 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.

Exporting and Importing CAVS Definitions and Instances

This chapter describes how to export and import Composite Application Validation System (CAVS) definitions and instances.

This chapter includes the following sections:

- Section 12.1, "How to Export and Import Definitions"
- Section 12.2, "How to Export Test and Simulator Instances"
- Section 12.3, "How to Export Group Instances"

12.1 How to Export and Import Definitions

To export and import CAVS definitions:

1. Access the Oracle Application Integration Architecture (AIA) Home Page. In the Composite Application Validation System area, click the **Go** button. Select the **Definitions** tab. The Definitions page displays, as shown in [Figure 12-1](#).

Figure 12-1 Definitions Page

The screenshot displays the Oracle AIA Composite Application Validation System interface. At the top, there are navigation tabs: Definitions, Instances, Group Definitions, Group Instances, and Routing Setup. The 'Definitions' tab is active. Below the tabs is a search form with the following fields:

- Id**: Text input
- Name**: Text input
- Type**: Dropdown menu
- Service Name**: Text input
- Service Version**: Text input
- Process Name**: Text input
- PIP Name**: Text input
- Endpoint URL**: Text input
- SOAP Action**: Text input
- State**: Dropdown menu

A 'Search' button is located below the search fields. Below the search form is a 'Search Result Selection' section with a table of results. The table has the following columns: Id, Name, Type, State, Service Type, Service Name, and Service Version. The table contains 11 rows of data:

Id	Name	Type	State	Service Type	Service Name	Service Version
1000	AIAB2BInterface EDI X12 855 Simulator	Simulator	Notify	AIAB2BInterface	1.0	
1001	AIADemoProcessSalesOrderOrchestrationEBS Simulator	Simulator	Synchronous	StoreFrontService	1.0	
1002	StoreFrontService Update Order Simulator	Simulator	Synchronous	InternalWarehouse	1.0	
1003	InternalWarehouse Simulator	Simulator	Notify	USPSShipmentService / AIADemoBatchJMSAdapter	1.0	
1004	Shipment (USPS and Batch) Simulator	Simulator	Synchronous	PartnerSupplier	1.0	
1005	PartnerSupplier Simulator	Simulator	Notify	AIADemoSyncCustomerPartyListCRMProvDBAdapter Sim...	1.0	
1006	AIADemoSyncCustomerPartyListCRMProvDBAdapter Sim...	Simulator	Notify	AIADemoSyncCustomerPartyListEBS	1.0	
1007	AIADemoSyncCustomerPartyListEBS Simulator	Simulator	Synchronous	CreditCardAuthorization	1.0	
1008	CreditCardAuthorization Simulator	Test	Notify	AIADemoProcessSalesOrderShopReqABCImpl	1.0	
1009	AIADemoProcessSalesOrderShopReqABCImpl	Test	Synchronous	AIADemoQueryCustomerPartyCRMProvABCImpl	1.0	
1010	AIADemoQueryCustomerPartyCRMProvABCImpl					

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.
2. 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 A, "XML Structures of Exportable CAVS Definitions and Instances."](#)

3. Use the **Import** button to upload a test, simulator, or group definition in the XML flat-file format generated by 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 A, "XML Structures of Exportable CAVS Definitions and Instances."](#)

4. 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:

Click the **Change URL** button on this page to access the Modify Test Definitions page, where you can update the Endpoint URL field value.

For more information about the Endpoint URL field, see [Section 4.1, "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 [Chapter 6, "Searching for Test and Simulator Definitions."](#)

12.2 How to Export Test and Simulator Instances

To export test and simulator instances:

1. Access the AIA Home Page. In the Composite Application Validation System area, click the **Go** button. Select the **Instances** tab. The Instances page displays, as shown in [Figure 12–2](#).

For more information about the Instances page, see [Chapter 9, "Working with Test and Simulator Instances."](#)

Figure 12–2 Instances Page

The screenshot shows the Oracle AIA Composite Application Validation System interface. The top navigation bar includes 'ORACLE AIA Composite Application Validation System' and links for 'Home', 'About AIA', 'Logout', and 'weblogic'. The main content area is titled 'Instances' and contains a 'Search Instances' form with fields for Id, Definition Id, Name, Status, Type, Service Type, Service Name, Service Version, Process Name, PIP Name, Start Date, and End Date. Below the search form is a 'Search Result Selection' section with buttons for 'Delete', 'Export', and 'Detach'. A table displays the search results with columns for Id, Definition Id, Name, Type, Status, Start Date, and End Date.

Id	Definition Id	Name	Type	Status	Start Date	End Date
1077	1083	TestDefinition	Test	Ended	Dec 2, 2010	Dec 2, 2010
1076	1068	CreditCardAuthori...	Simulator	Passed	Dec 2, 2010	Dec 2, 2010
1075	1068	CreditCardAuthori...	Simulator	Passed	Dec 2, 2010	Dec 2, 2010
1074	1082	AIADemoProcessC...	Test	Failed	Dec 2, 2010	Dec 2, 2010
1073	1074	AIADemoCreateR...	Test	Failed	Dec 2, 2010	Dec 2, 2010
1072	1068	CreditCardAuthori...	Simulator	Passed	Dec 2, 2010	Dec 2, 2010
1071	1068	CreditCardAuthori...	Simulator	Passed	Dec 2, 2010	Dec 2, 2010
1070	1068	CreditCardAuthori...	Simulator	Passed	Dec 2, 2010	Dec 2, 2010
1069	1068	CreditCardAuthori...	Simulator	Passed	Dec 2, 2010	Dec 2, 2010

2. Use the Export feature 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.

For more information about the structure of the Definitions.xml file created by the CAVS export instance feature, see [Appendix A, "XML Structures of Exportable CAVS Definitions and Instances."](#)

3. 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.

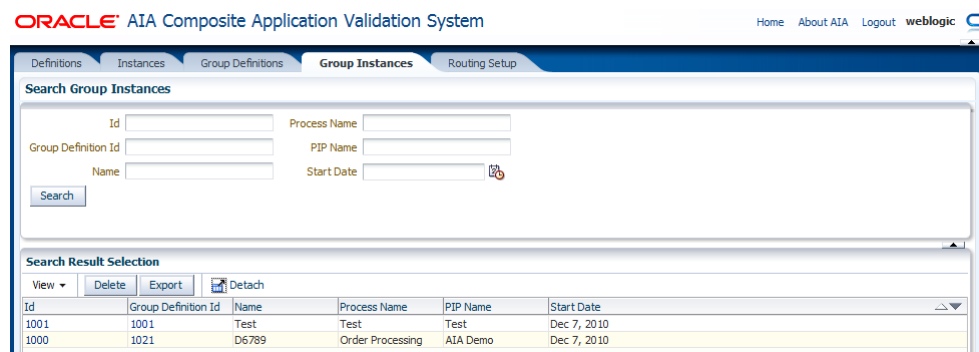
12.3 How to Export Group Instances

To export group instances:

1. Access the AIA Home Page. In the Composite Application Validation System area, click the **Go** button. Select the **Group Instances** tab. The Group Instances page displays, as shown in [Figure 12–3](#).

For more information about the Group Instances page, see [Section 10.1, "How to View Group Instances."](#)

Figure 12–3 Group Instances Page



2. 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 A, "XML Structures of Exportable CAVS Definitions and Instances."](#)

Introduction to Oracle AIA Error Handling

This chapter provides an introduction to the Oracle AIA Error Handling Framework. It also provides overviews about business faults, BPEL and Mediator system faults, and Oracle B2B errors.

The Oracle AIA 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.

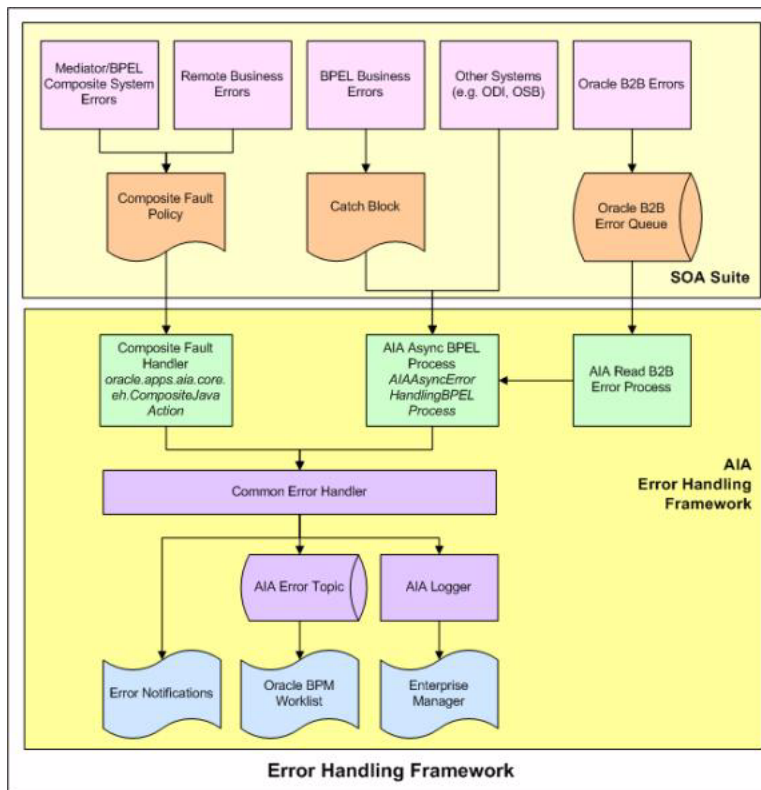
This chapter includes the following sections:

- [Section 13.1, "Introduction to the Error Handling Framework"](#)
- [Section 13.2, "Introduction to Error Handling for Business Faults"](#)
- [Section 13.3, "Introduction to Error Handling for BPEL and Mediator System Faults"](#)
- [Section 13.4, "Introduction to Error Handling for Oracle B2B Errors"](#)

13.1 Introduction to the Error Handling Framework

This section includes the following topic: [Section 13.1.1, "Fault Categories."](#)

[Figure 13-1](#) provides a high-level overview of the Error Handling Framework.

Figure 13–1 Key Features of Error Handling Framework Components

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, run-time, and 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

- Ability to extend error handling capabilities.
- Automated error actions.
- Ability to automatically acts upon the errored object to provide automated retry actions, error notifications, and logging.
- Error actions, including 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 "Configuring Oracle AIA Processes for Error Handling and Trace Logging" in *Oracle Fusion Middleware Developer's Guide for Oracle Application Integration Architecture Foundation Pack*.

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 "Configuring Oracle AIA Processes for Error Handling and Trace Logging" in *Oracle Fusion Middleware Developer's Guide for Oracle Application Integration Architecture Foundation Pack*.

For more information about the Composite Fault Policy framework, see "Using Fault Handling in a BPEL Process" in *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.

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:

- Failure of document 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 acknowledgment messages from trading partner applications containing these failures are treated as independent flows.

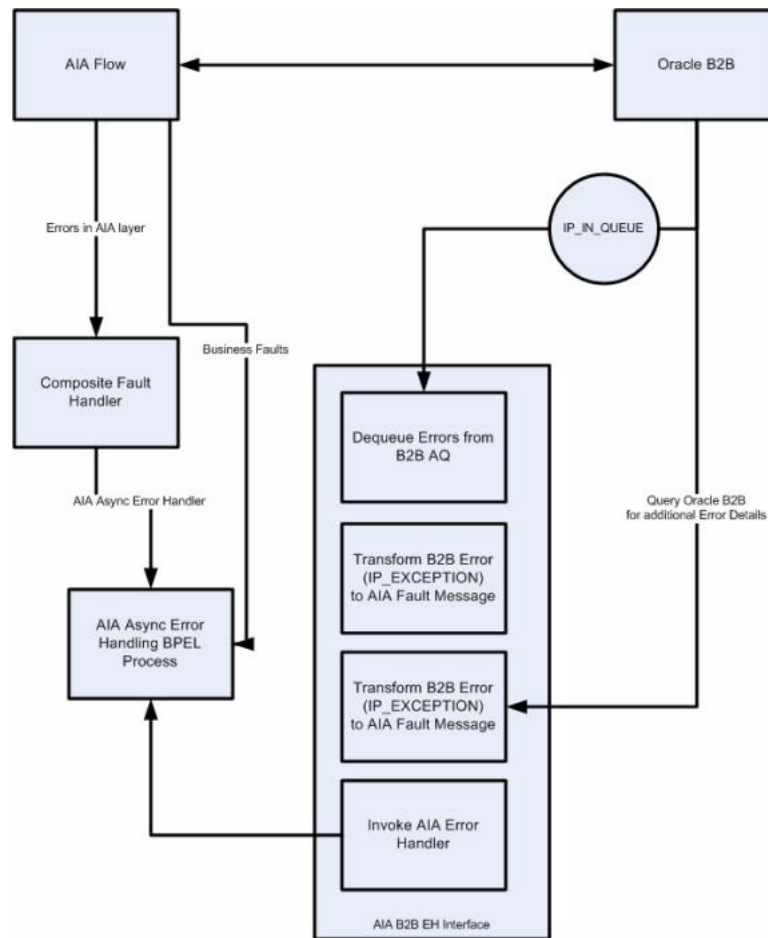
When Oracle B2B encounters these system errors, its default behavior is to publish the error to the Oracle Advanced Queuing (AQ) queue defined in the Oracle B2B infrastructure schema.

The details of the AQ to which Oracle B2B posts errors are covered in [Table 13–1](#).

Table 13–1 Oracle B2B Error AQ Details

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

[Figure 13–2](#) illustrates the way in which the AIA error handling framework captures B2B errors:

Figure 13–2 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 [Chapter 19, "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.

Setting Up Error Handling

This chapter provides an overview and discusses how to create user roles, associate email addresses to user roles, configure notification details, and how to set up error handling configuration details.

This chapter includes the following sections:

- [Section 14.1, "Introduction to Setting Up Error Handling"](#)
- [Section 14.2, "How to Create Error Handling User Roles"](#)
- [Section 14.3, "How to Associate Email Addresses with Error Handling User Roles"](#)
- [Section 14.4, "How to Configure Notification Details"](#)
- [Section 14.5, "How to Set Up AIA Error Handling Configuration Details"](#)

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 [Section 15.4, "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 [Section 16.2, "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 [Section 14.2, "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 [Section 14.3, "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 [Section 14.4, "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 [Section 14.5, "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 "Configuring Oracle AIA Processes for Error Handling and Trace Logging" in *Oracle Fusion Middleware Developer's Guide for Oracle Application Integration Architecture Foundation Pack*.

[Figure 14–1](#) illustrates the way in which error handling setup elements enable Error Notification functionality.

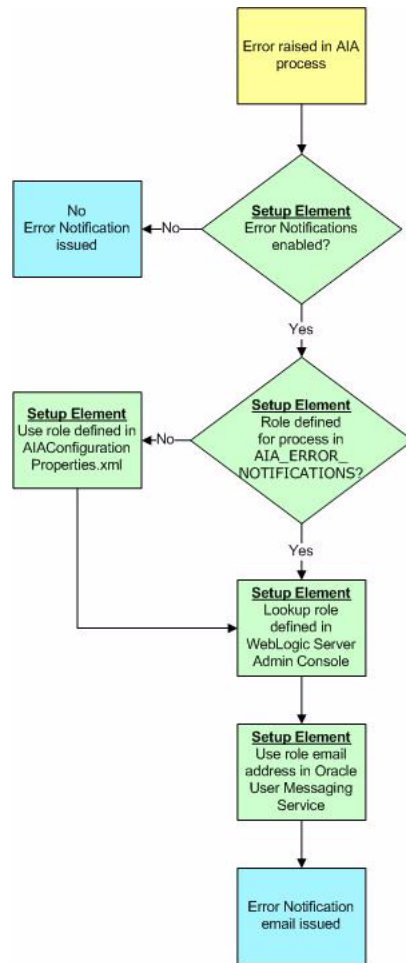
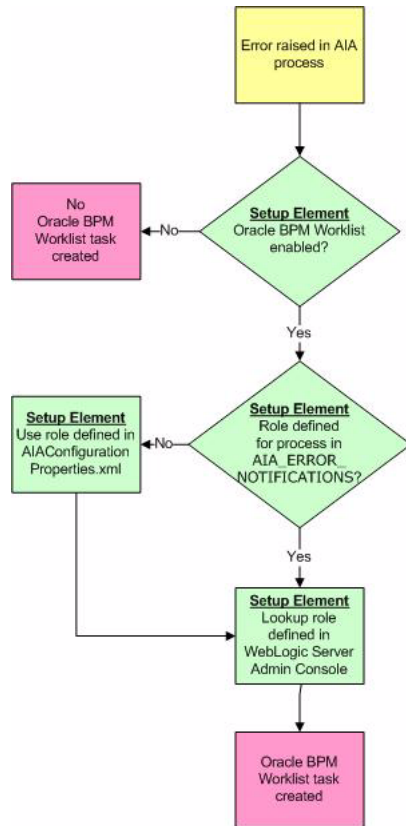
Figure 14–1 Error Handling Setup Elements That Enable Error Notification Functionality

Figure 14–2 illustrates the way in which error handling setup elements enable Oracle Worklist functionality.

Figure 14–2 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 "Using the Administration Console to Manage Users, Groups, and Roles" in *Oracle Fusion Middleware Securing Resources Using Roles and Policies for Oracle WebLogic Server*.

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 "Security Provider Databases" in *Oracle Fusion Middleware Understanding Security for Oracle WebLogic Server*.

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 "How to Manage Messaging Channels" in *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.

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

For more information about creating user roles, see [Section 14.2, "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 [Section 14.2, "How to Create Error Handling User Roles."](#)

14.4 How to Configure Notification Details

To configure notification details:

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

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

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 "How to Set Up AIA Workstation" in *Oracle Fusion Middleware Developer's Guide for Oracle Application Integration Architecture Foundation Pack*.

14.5 How to Set Up AIA Error Handling Configuration Details

This section includes the following topic: [Section 14.5.1, "What You Need to Know about Setting Up Error Handling Configurations."](#)

To set up AIA error handling configuration details:

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 Notification** tab. The Error Notification page displays as shown in [Figure 14-3](#) and [Figure 14-4](#).

Figure 14-3 Error Notification Page (1 of 2)

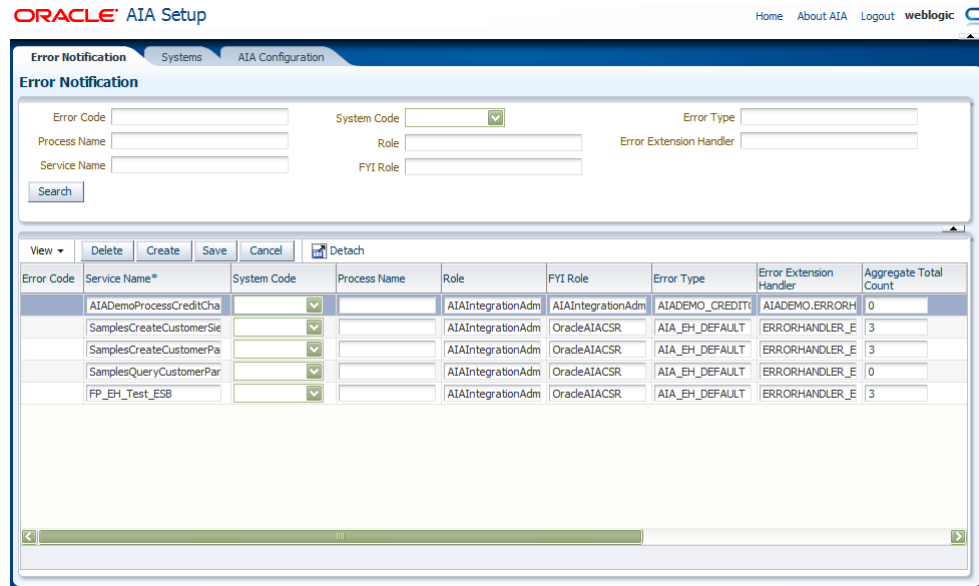


Figure 14-4 Error Notification Page (2 of 2)

Aggregate Current Count	Start Date	End Date
0	Dec 1, 2010 7:28:23 AM	Dec 1, 2010 7:28:23 AM
0	Dec 2, 2010 10:04:13 AM	Dec 2, 2010 10:04:13 AM
0	Dec 2, 2010 10:05:07 AM	Dec 2, 2010 10:05:07 AM
0	Dec 2, 2010 2:37:49 PM	Dec 2, 2010 2:37:49 PM
	Dec 1, 2010 9:03:37 AM	Dec 6, 2010 9:03:37 AM

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 "Configuring Oracle AIA Processes for Error Handling and Trace Logging" in *Oracle Fusion Middleware Developer's Guide for Oracle Application Integration Architecture Foundation Pack*.

Descriptions of key elements on the Error Notification page are discussed in [Table 14-1](#).

Table 14-1 Error Notification Page Elements

Element	Description
ErrorCode	<p>For BPEL and Mediator process system error notifications, this is the fault code.</p> <p>For business errors using catch blocks, this is the business error code you are catching. This is user-defined, for example, <i>OUT_OF_INV</i>.</p> <p>The sample error code for system faults is <code>{http://schemas.oracle.com/bpel/extension}remoteFault</code> and <code>{http://schemas.oracle.com/bpel/extension}bindingFault</code>.</p>
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, <i>SampleBPELProcess</i> .
NotificationRole	<p>If you have enabled Error Notifications, specify the user role that you want to receive Actor error notifications for a process.</p> <p>If you have enabled Oracle Worklist functionality, specify the role to which you want to assign Actor tasks for a process.</p> <p>The Actor role is responsible for taking action to correct the error that generated the notification.</p> <p>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 user store.</p> <p>For more information, see Section 14.2, "How to Create Error Handling User Roles."</p> <p>For Error Notifications functionality, ensure that the user role has an email address defined in the Oracle WebLogic User Messaging Service.</p> <p>For more information, see Section 14.3, "How to Associate Email Addresses with Error Handling User Roles."</p>

Table 14–1 (Cont.) Error Notification Page Elements

Element	Description
FyiNotificationRole	<p>If you have enabled Error Notifications, specify the user role that you want to receive FYI error notifications for a process.</p> <p>If you have enabled Oracle BPM Worklist functionality, specify the role to which you want to assign FYI tasks for a process.</p> <p>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.</p> <p>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.</p> <p>For more information, see Section 14.2, "How to Create Error Handling User Roles."</p> <p>For Error Notifications functionality, ensure that the user role has an email address defined in Oracle User Messaging Service preferences.</p> <p>For more information, see Section 14.3, "How to Associate Email Addresses with Error Handling User Roles."</p>
ErrorType	<p>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.</p> <p>Enter a unique value here if you are using extended error handling functionality.</p> <p>For more information about extending error handling, see "Configuring Oracle AIA Processes for Error Handling and Trace Logging" in <i>Oracle Fusion Middleware Developer's Guide for Oracle Application Integration Architecture Foundation Pack</i>.</p> <p>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, <i>AIA_EH_DEFAULT, ORDER_FO</i>, where <i>AIA_EH_DEFAULT</i> is the default Oracle AIA follow-through action, and <i>ORDER_FO</i> 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 run time.</p>
ErrorExtHandler (error extension handler)	<p>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.</p> <p>For more information about extending fault messages, see "Configuring Oracle AIA Processes for Error Handling and Trace Logging" in <i>Oracle Fusion Middleware Developer's Guide for Oracle Application Integration Architecture Foundation Pack</i>.</p> <p>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.</p>
AggrCountTot (aggregation count total)	<p>Error notification throttling must be enabled before this field value can be used to control the issuance of error notifications.</p> <p>For more information, see Section 15.2.2, "How to Enable Error Notification Throttling."</p> <p>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.</p> <p>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.</p>

Table 14–1 (Cont.) Error Notification Page Elements

Element	Description
StDatetime/EndDate time	<p>Error notification throttling must be enabled before these field values can be used to control the issuance of error notifications.</p> <p>For more information, see Section 15.2.2, "How to Enable Error Notification Throttling."</p> <p>Enter the start and end date-and-time intervals to which you want the count value to apply.</p> <p>For example, if you set the AggrCountTot field value to <i>100</i>, the start date and time to <i>30-Oct-2009 18:00:00</i>, and the end date and time to <i>01-Nov-2009 17:00:00</i>, 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.</p> <p>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.</p>

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

The Error Handling Framework uses run-time 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 run-time 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.

Using Error Notifications

This chapter provides an overview of error notifications and describes how to set up notification throttling, how to customize notification emails, and how to disable error notifications.

This chapter includes the following sections:

- [Section 15.1, "Introduction to Error Notifications"](#)
- [Section 15.2, "Setting Up Error Notification Throttling"](#)
- [Section 15.3, "Customizing Error Notification Emails"](#)
- [Section 15.4, "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 [Chapter 14, "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 [Section 14.2, "How to Create Error Handling User Roles"](#) and [Section 14.3, "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 [Section 14.5, "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 [Section 15.2, "Setting Up Error Notification Throttling."](#)

[Figure 15-1](#) provides a sample error notification email.

Figure 15–1 Sample Error Notification Email

An error has occurred during the processing of AIA Integration Error in AIA SamplesCreateCustomerSiebelReqABCImplProcess 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/SamplesCreateCustomerSiebelReqABCImpl+\[1.0\]&type=oracle\_soa\_composite&soaContext=default/SamplesCreateCustomerSiebelReqABCImpl!1.0/140032
-----

Please access the task in the Worklist Application :
-----
http://sdc68063crm.us.oracle.com:8024/integration/worklistapp/faces/home.jspx
-----

```

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 [Section 16.2, "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 [Section 15.3, "Customizing Error Notification Emails."](#)

15.2 Setting Up Error Notification Throttling

This section includes the following topics:

- [Section 15.2.1, "Introduction to Error Notification Throttling"](#)
- [Section 15.2.2, "How to Enable Error Notification Throttling"](#)
- [Section 15.2.3, "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.NOTIF` 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 "Building AIA Integration Flows" in *Oracle Fusion Middleware Developer's Guide for Oracle Application Integration Architecture Foundation Pack*.

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 [Section 15.2.2, "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 AIA Home Page. In the Setup area, click the **Go** button. Select the **Error Notification** tab. The Error Notification page displays as shown in [Figure 15-2](#) and [Figure 15-3](#).

Figure 15-2 Error Notification Page (1 of 2)

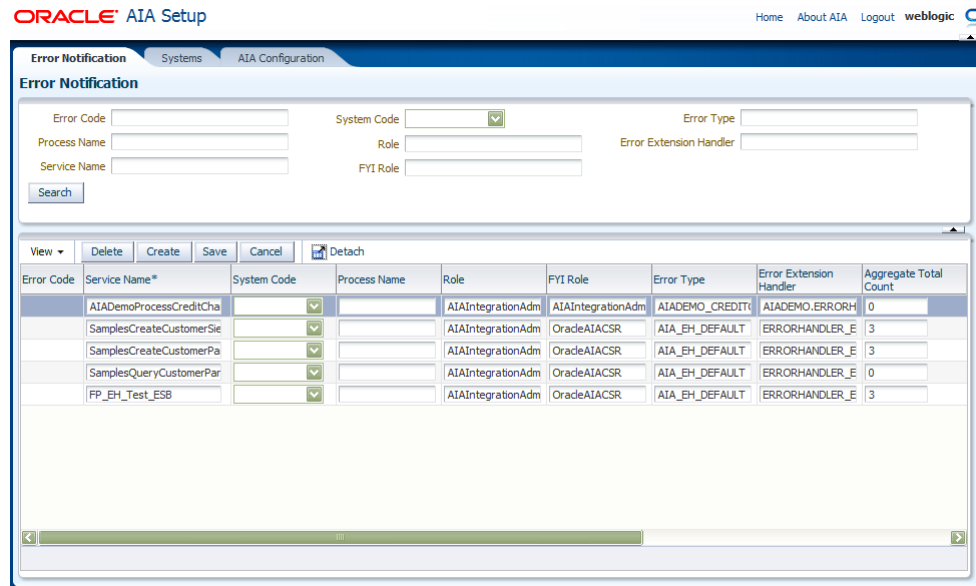


Figure 15-3 Error Notification Page (2 of 2)

Aggregate Current Count	Start Date	End Date
0	Dec 1, 2010 7:28:23 AM	Dec 1, 2010 7:28:23 AM
0	Dec 2, 2010 10:04:13 AM	Dec 2, 2010 10:04:13 AM
0	Dec 2, 2010 10:05:07 AM	Dec 2, 2010 10:05:07 AM
0	Dec 2, 2010 2:37:49 PM	Dec 2, 2010 2:37:49 PM
	Dec 1, 2010 9:03:37 AM	Dec 6, 2010 9:03:37 AM

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 [Section 14.5, "How to Set Up AIA Error Handling Configuration Details."](#)

15.3 Customizing Error Notification Emails

This section includes the following topics:

- [Section 15.3.1, "Introduction to Error Notification Customization"](#)
- [Section 15.3.2, "How to Customize the Subject Line of Error Notification Emails"](#)
- [Section 15.3.3, "How to Customize the Body Text of Error Notification Emails"](#)
- [Section 15.3.4, "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 in [Example 15–1](#).

Example 15–1 AIAEHNotifications.xml

```
<?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/messageFlow?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

[Example 15–2](#) is an example of how you can customize the `EMAIL` element in `AIAEHNotifications.xml` to provide content that appears in error notification emails to Actor roles.

Example 15–2 EMAIL Element in AIAEHNotifications.xml

```

<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:Faulting
    Service/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

[Example 15–3](#) is an example of how you can customize the `FYI_EMAIL` element in `AIAEHNotifications.xml` to provide content that appears in error notification emails to FYI roles.

Example 15–3 FYI_EMAIL Element in AIAEHNotifications.xml

```

<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:Faulting
    Service/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 needs.

`$hostname`, `$adminport`, and `$domain` tokens shown in [Example 15-4](#) are populated with implementation-specific values by the Oracle AIA Installer upon installation of Foundation Pack.

Example 15-4 URL Element in AIAEHNotification.xml

```
<URL>
=====
Please click on the following URL to view instancedetails in the em console:
=====
@ http://$adminHost:$adminPort/em/faces/ai/soa/messageFlow?target=/Farm_
  $domainName/$domainName/$targetServer/##PROPS.{compositeName}
  ###[##PROPS.{composite Revision}##]%26type=oracle_soa_
  composite%26soaContext=##PROPS.{compositeDN}##/##PROPS.{composite
  InstanceID}##
=====
</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 [Example 15-5](#) are populated with implementation-specific values by the Oracle AIA Installer upon installation of Foundation Pack.

Example 15-5 EXT_URL Element in AIAEHNotifications.xml

```
<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 [Section 16.2, "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.

You can customize the subject line text to include multibyte characters for translation to other languages. The encoding used is UTF-8. After customized, the file must be uploaded to Oracle Meta Data Services (MDS).

For more information about uploading content to MDS, see "Updating MDS" in *Oracle Fusion Middleware Developer's Guide for Oracle Application Integration Architecture Foundation Pack*.

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.

[Example 15-6](#) is an example of how to customize the subject line of error notification emails.

Example 15-6 Customizing the Subject Line of 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 instancedetails in the em console :
=====
@ http://$adminHost:$adminPort/em/faces/ai/soa/messageFlow?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 "Configuring Oracle AIA Processes for Error Handling and Trace Logging" in *Oracle Fusion Middleware Developer's Guide for Oracle Application Integration Architecture Foundation Pack*.

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 "Configuring Oracle AIA Processes for Error Handling and Trace Logging" in *Oracle Fusion Middleware Developer's Guide for Oracle Application Integration Architecture Foundation Pack*.

5. Reload updates to the AIAEHNotifications.xml file.

For more information about reloading updates to AIAEHNotifications.xml, see "Building AIA Integration Flows" in *Oracle Fusion Middleware Developer's Guide for Oracle Application Integration Architecture Foundation Pack*.

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.

You can customize the body text to include multibyte characters for translation to other languages. The encoding used is UTF-8. After customized, the file must be uploaded to Oracle Meta Data Services (MDS).

For more information about uploading content to MDS, see "Updating MDS" in *Oracle Fusion Middleware Developer's Guide for Oracle Application Integration Architecture Foundation Pack*.

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.

Example 15-7 is an example of how to customize the body text of an error notification email.

Example 15-7 Customizing the Body Text of 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}
      ##ProcessFYI</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 instancedetails in the em console :
    =====
    @ http://$adminHost:$adminPort/em/faces/ai/soa/messageFlow?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 "Configuring Oracle AIA Processes for Error Handling and Trace Logging" in *Oracle Fusion Middleware Developer's Guide for Oracle Application Integration Architecture Foundation Pack*.

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 "Configuring Oracle AIA Processes for Error Handling and Trace Logging" in *Oracle Fusion Middleware Developer's Guide for Oracle Application Integration Architecture Foundation Pack*.

5. Reload updates to the AIAEHNotifications.xml file.

For more information about reloading updates to AIAEHNotifications.xml, see "Building AIA Integration Flows" in *Oracle Fusion Middleware Developer's Guide for Oracle Application Integration Architecture Foundation Pack*.

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 [Section 16.2, "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 "Viewing the Audit Trail and Process Flow of a BPEL Process Service Component" in *Oracle Fusion Middleware Administrator's Guide for Oracle SOA Suite and Oracle Business Process Management Suite*.

Figure 15–4 shows a sample error notification email text, which provides a link to flow trace details in the Oracle Enterprise Manager console.

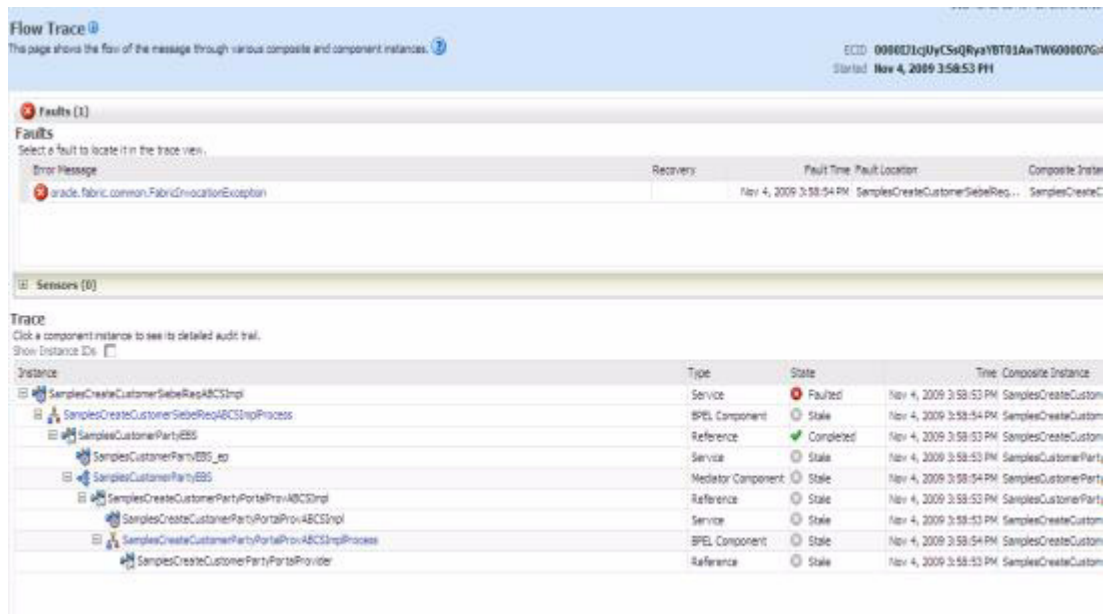
Figure 15–4 Sample Error Notification Email Text Providing a Link to Flow Trace Details in the Oracle Enterprise Manager Console

```

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/SamplesCreateCustomerSiebelReqABCSImpl1.0/140032
=====
Please access the task in the Worklist Application :
=====
http://sdc68063crm.us.oracle.com:8024/integration/worklistapp/faces/home.jspx
=====
    
```

Figure 15–5 is an example of fault details being displayed on the Oracle Enterprise Console Flow Trace page.

Figure 15–5 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 "Configuring Oracle AIA Processes for Error Handling and Trace Logging" in *Oracle Fusion Middleware Developer's Guide for Oracle Application Integration Architecture Foundation Pack*.

4. As delivered, the <EXT_URL> element provides a link to the Oracle BPM Worklist, as shown in [Example 15-8](#).

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.

Example 15-8 Customizing Application Links in Body Text of 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 AIAIntegration 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/messageFlow?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 "Building AIA Integration Flows" in *Oracle Fusion Middleware Developer's Guide for Oracle Application Integration Architecture Foundation Pack*.

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 AIAEHNotifications.xml, see "Building AIA Integration Flows" in *Oracle Fusion Middleware Developer's Guide for Oracle Application Integration Architecture Foundation Pack*.

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.

Using the Oracle BPM Worklist

This chapter provides an overview of the Oracle BPM Worklist and describes how to enable and use the 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.

This chapter includes the following sections:

- [Section 16.1, "Introduction to the Oracle BPM Worklist"](#)
- [Section 16.2, "How to Enable the Oracle BPM Worklist"](#)
- [Section 16.3, "How to Use the Oracle BPM Worklist"](#)

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 [Section 16.2, "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 [Chapter 15, "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
- EBName
- 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 "Building AIA Integration Flows" in *Oracle Fusion Middleware Developer's Guide for Oracle Application Integration Architecture Foundation Pack*.

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 [Chapter 15, "Using Error Notifications."](#)

16.3 How to Use the Oracle BPM Worklist

After 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 "Using Oracle BPM Worklist" in *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.

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 [Chapter 17, "Using the Message Resubmission Utility."](#)

Using the Message Resubmission Utility

This chapter provides an overview and describes how to use the message resubmission utility.

This chapter includes the following sections:

- [Section 17.1, "Introduction to the Message Resubmission Utility"](#)
- [Section 17.2, "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 "Configuring Oracle AIA Processes for Error Handling and Trace Logging" in *Oracle Fusion Middleware Developer's Guide for Oracle Application Integration Architecture Foundation Pack*.

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 a Queue or topic or Resequencer. 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 [Chapter 16, "Using the Oracle BPM Worklist."](#)

For more information about error notifications, see [Chapter 15, "Using Error Notifications."](#)

Once notified, the most natural course of action is for the administrator to bring up the failed service or component. After 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.

It can be done in three ways:

- Oracle AQ store based or
- WLS JMS store based (can be configured with file system or db).
- Resequencer based

17.2.1 AQ Store Based Resubmission

1. For message resubmission scenarios that involve Oracle Advanced Queue, Topic, internally the MSG_RESUBMIT stored procedure is used. This procedure assumes that the message type is SYS.AQ\$_JMS_MESSAGE.

If the message type being used is not SYS.AQ\$_JMS_MESSAGE, 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 "Configure AQ JMS Foreign Server Destinations" in *Oracle Fusion Middleware Configuring and Managing JMS for Oracle WebLogic Server*.

2. 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 "Configuring Oracle AIA Processes for Error Handling and Trace Logging" in *Oracle Fusion Middleware Developer's Guide for Oracle Application Integration Architecture Foundation Pack*.

Alternatively, you can also look up the aia-error.log in the Oracle Enterprise Manager.

- a. Under **WebLogic Domain**, <domain name>, right-click the manage server entry (usually `soa_server1`).
- b. 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 Enterprise Manager, see [Chapter 18, "Using Trace and Error Logs."](#)

3. AIAResubmissionUtility is available under \$AIA_HOME/util, so browse to \$AIA_HOME/util/AIAResubmissionUtility. Set all the required values in ResubmissionParams.properties file.

Note: Since it is AQ based resubmission, `jms.aq` property in the properties file should be set to true in the properties file. The `messageID` should be in normal format, and should not start with ID : < >

Example 17–1 Sample ResubmissionParams.properties for AQ based Resubmission

```

jms.app.hostName=hostservername.domain.com
jms.app.admin.port=port
jms.app.soa.port=8097
jms.app.userName=weblogic
jms.app.password=password
jms.aq=true
jms.moduleName=AIAJMSModule
jms.resourceCFJndi=
jms.errorResourceCFJndi=
#QUEUE/TOPIC/RESEQUENCER - 1/2/3
resourceType=1
#queueName/topicName/routingServiceName
resourceName=AIA_SiebelCustomerJMSQueue
#messageID/groupID
messageID= 7109EDC5FFD9BA25E04014908FC62C90
#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

```

4. For Windows, execute `$AIA_INSTANCE\bin\aiainv.bat`.
For Linux, source `$AIA_INSTANCE/bin/aiainv.sh`.
5. Navigate to `$AIA_HOME/util/AIAMessageResubmissionUtil` and execute the following:

```

ant -f MessageResubmit.xml -logfile $AIA_
HOME/util/AIAResubmissionUtility/MessageResubmit.log

```

The `MessageResubmit.xml` script references the edited `ResubmissionParams.properties` file. After it is run, the script resets the message status back to a ready state so that the transaction can resume its flow.

17.2.2 WLS JMS based Resubmission

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 "Configuring Oracle AIA Processes for Error Handling and Trace Logging" in *Oracle Fusion Middleware Developer's Guide for Oracle Application Integration Architecture Foundation Pack*.

Alternatively, you can also look up the `aia-error.log` in the Oracle Enterprise Manager.

- a. Under **WebLogic Domain**, `<domain name>`, right-click the manage server entry (usually `soa_server1`).
- b. 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 Enterprise Manager, see [Chapter 18, "Using Trace and Error Logs."](#)

2. You may optionally define `jms.resourceCFJndi=` and `jms.errorResourceCFJndi=` property values in the `ResubmissionParams.properties` file. The default values for the error jndi are fetched based on the `resourceName` that is provided in the properties file by using the mbean infrastructure. The connection factories are derived based on the naming standards. See the second list item below.
 - The `jms.resourceCFJndi=` property defines a resource-specific `ConnectionFactory` that will be used to connect to the resource error queue. In this context, a resource is a JMS queue or topic. This property cannot have multiple values, even if you have multiple connection factories. You must specify one `ConnectionFactory` to be used by the resubmission script.
 - If the resource name is `AIASamples_Queue` and the JNDI is `jndi/aia/AIASamples_Queue`, the `ConnectionFactory` property value would be `jndi/aia/AIASamples_QueueCF`.
 - The `jms.errorResourceCFJndi=` property defines a generic `ConnectionFactory` that will be used to connect to all resource error queues that are not explicitly defined using the `jms.resourceCFJndi=` property. If you do not define this value, it will be automatically generated during Foundation Pack installation as `jms.errorResourceCFJndi=jms/aia/AIAErrorCF`.
3. `AIAResubmissionUtility` is available under `$AIA_HOME/util`, so browse to `$AIA_HOME/util/AIAResubmissionUtility`. Set all the required values in `ResubmissionParams.properties` file.

Note: Since it is WLS JMS based resubmission, `jms.aq` property in the properties file should be set to false in the properties file. The `messageID` should be in the format `ID : < >`

Example 17-2 Sample ResubmissionParams.properties for WLS JMS based Resubmission

```
jms.app.hostName=hostname.domain.com
jms.app.admin.port=port
jms.app.soa.port=8097
jms.app.userName=weblogic
jms.app.password= password
jms.aq=false
jms.moduleName=AIAJMSModule
jms.resourceCFJndi=jms/aia/AIAEbizCF
jms.errorResourceCFJndi=jms/aia/AIAErrorCF
#QUEUE/TOPIC/RESEQUENCER - 1/2/3
resourceType=3
#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
```

4. For Windows, execute `$AIA_INSTANCE\bin\aiacnv.bat`.
For Linux, source `$AIA_INSTANCE/bin/aiaenv.sh`.
5. Navigate to `$AIA_HOME/util/AIAMessageResubmissionUtil` and execute the following:

```
ant -f MessageResubmit.xml -logfile $AIA_
HOME/util/AIAResubmissionUtility/MessageResubmit.log
```

The `MessageResubmit.xml` script references the edited `ResubmissionParams.properties` file. After it is run, the script resets the message status back to a ready state so that the transaction can resume its flow.

17.2.3 Resequencer Based Resubmission

1. Faults/rejected messages which are marked as Recovery Needed in the EM Console can only be resubmitted using `AIAResubmissionUtility`. To get the list of faulted/rejected messages in the EM Console, navigate to SOA, click on corresponding domain and navigate to Faults and Rejected messages tab in the middle pane.
2. Make a list of all faulted instances (which are marked as Recovery Needed).
3. Get the composite name and messageID/GroupID (xpath of the messageID is defined at the design time of the resequencer) of the faulted instances.
4. `AIAResubmissionUtility` is available under `$AIA_HOME/util`. Navigate to `$AIA_HOME/util/AIAResubmissionUtility` and set all the required values in `ResubmissionParams.properties` file.

Note: The `resourceName` should be in the format `default/<compositeName>!<version>`. Multiple messageID's of a particular composite can be set for `messageID` property with comma `"/` as a delimiter. `7109EDC5FFD9BA25E0401,9EDC5FFD9BA25E04014908F`.

Example 17-3 Sample ResubmissionParams.properties for Resequencer based Resubmission

```
jms.app.hostName=serverhostname.domain.com
jms.app.admin.port=adminport
jms.app.soa.port=soapport
jms.app.userName=weblogic
jms.app.password=password
jms.aq=false
jms.moduleName=AIAJMSModule
jms.resourceCFJndi=
jms.errorResourceCFJndi=
#QUEUE/TOPIC/RESEQUENCER - 1/2/3
resourceType=3
#queueName/topicName/routingServiceName
resourceName=default/JMSConsumer!1.0
#messageID/groupID
messageID=7109EDC5FFD9BA25E0
#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
```

5. For Windows, execute `$AIA_INSTANCE\bin\aiacnv.bat`.
For Linux, source `$AIA_INSTANCE/bin/aiaenv.sh`.
6. Navigate to `$AIA_HOME/util/AIAMessageResubmissionUtil` and execute the following:

```
ant -f MessageResubmit.xml -logfile $AIA_
HOME/util/AIAResubmissionUtility/MessageResubmit.log
```

The `MessageResubmit.xml` script references the edited `ResubmissionParams.properties` file. After it is run, the script resets the message status back to a ready state so that the message can resume its execution flow.

Using Trace and Error Logs

This chapter provides an overview and describes how to enable trace logging, how to set trace log levels, and how to access trace and error logs.

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.

This chapter includes the following sections:

- [Section 18.1, "Introduction to Trace and Error Logging"](#)
- [Section 18.2, "How to Enable Trace Logging"](#)
- [Section 18.3, "How to Set Trace Log Levels"](#)
- [Section 18.4, "How to Access Trace and Error Logs"](#)

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 "Configuring Oracle AIA Processes for Error Handling and Trace Logging" in *Oracle Fusion Middleware Developer's Guide for Oracle Application Integration Architecture Foundation Pack*.

- 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.

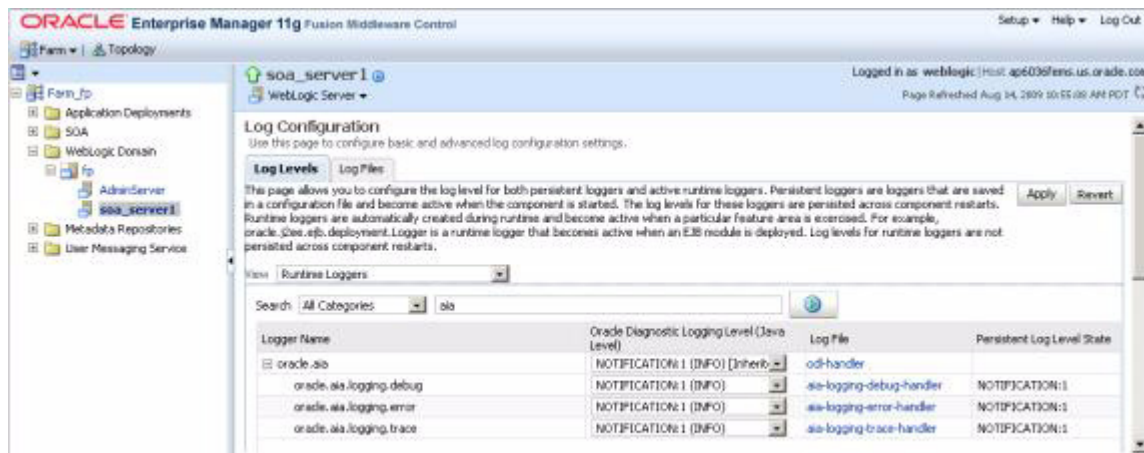
For more information about reloading updates to AIAConfigurationProperties.xml, see "Building AIA Integration Flows" in *Oracle Fusion Middleware Developer's Guide for Oracle Application Integration Architecture Foundation Pack*.

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, as shown in [Figure 18–1](#).

Figure 18–1 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 values listed in [Table 18–1](#), ordered from highest to lowest severity. The lower the severity level, the more information is written to the log file.

Table 18–1 Trace Log Levels of Severity

Severity Level	Description
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

This section includes the following topics:

- [Section 18.4.1, "Accessing Oracle AIA Logs in the Oracle Enterprise Manager Console"](#)
- [Section 18.4.2, "Searching for Oracle AIA Log Messages"](#)
- [Section 18.4.3, "Accessing Oracle AIA Log XML Files"](#)

18.4.1 Accessing Oracle AIA Logs in the Oracle Enterprise Manager Console

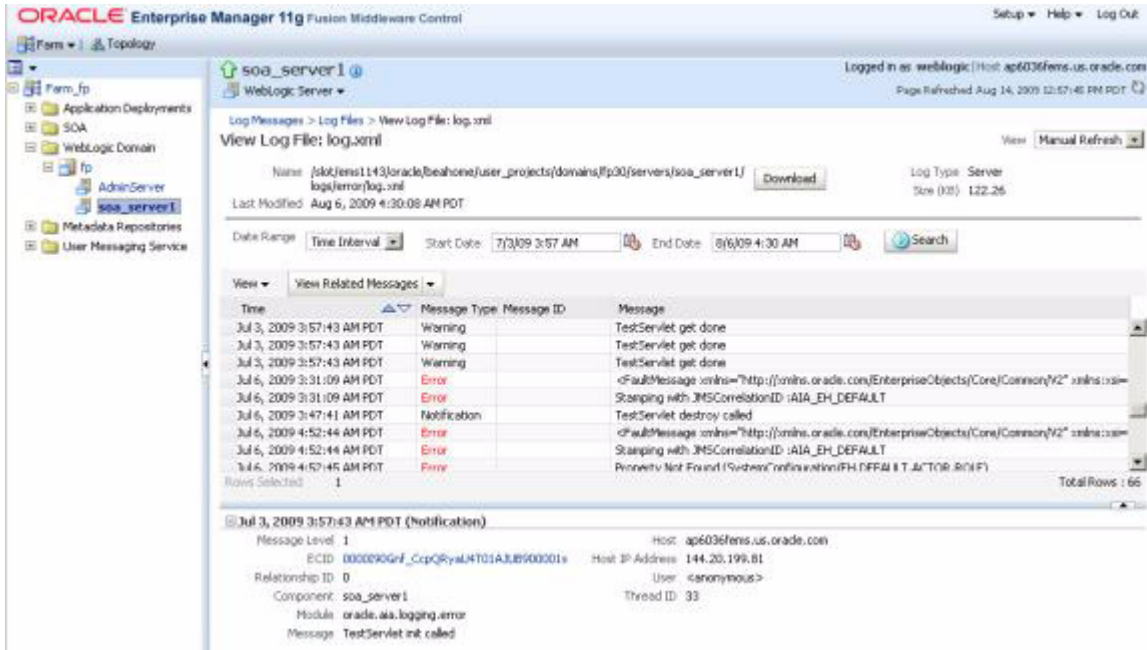
Log files can be accessed using the Oracle Enterprise Manager user interface, in much the same way that standard log files generated by various components of the Oracle SOA Suite can be handled in Oracle Enterprise Manager. Using Oracle Enterprise Manager as the user interface for the logs enables searching, 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 $\{\text{domain.home}\}/\text{servers}/\{\text{weblogic.Name}\}/\text{logs}$. The trace log file, aia-trace.log, can be found under $\{\text{domain.home}\}/\text{servers}/\{\text{weblogic.Name}\}/\text{logs}$.
4. To view a log file, select the file row and click the **View Log File** button, as shown in [Figure 18–2](#).

Figure 18–2 View Log File Page



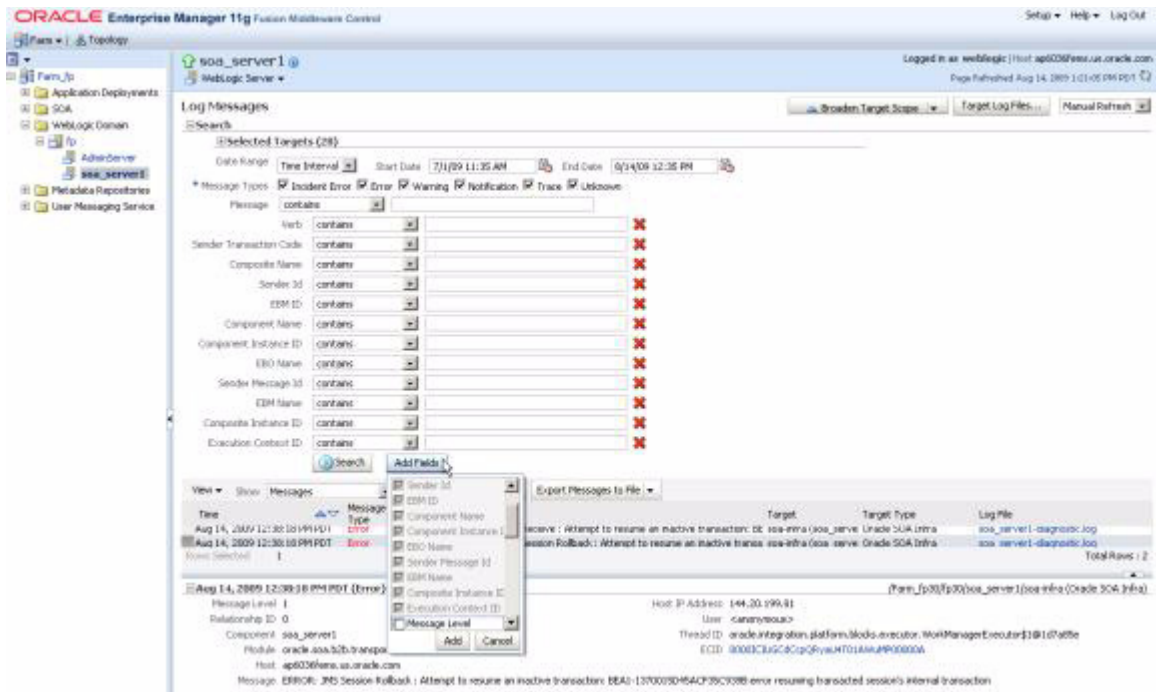
5. To download a log file, select the file row and click the **Download** button.

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, as shown in [Figure 18–3](#).

Figure 18–3 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`

Accessing Oracle B2B Errors

This chapter describes how to access Oracle B2B error reports.

This chapter includes the following section: [Section 19.1, "Accessing B2B Error Reports."](#)

19.1 Accessing B2B Error Reports

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**. after 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 acknowledgment 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" in *Oracle Fusion Middleware Developer's Guide for Oracle Application Integration Architecture Foundation Pack*.

XML Structures of Exportable CAVS Definitions and Instances

This appendix provides XML structures of exportable Composite Application Validation System (CAVS) definitions and instances.

This appendix includes the following sections:

- [Section A.1, "Definition.xml"](#)
- [Section A.2, "Instance.xml"](#)

A.1 Definition.xml

The structure of the Definitions.xml file created by the CAVS definition export feature is shown in [Example A-1](#).

This export feature should be used to migrate definitions between instances running on the same version of Oracle Application Integration Architecture (AIA) Foundation Pack.

Use this structure as a reference if you receive a validation error when importing definitions.

Edit this structure to create new definitions for importing to an Foundation Pack instance.

For more information about the definition export and import feature, see [Chapter 12, "Exporting and Importing CAVS Definitions and Instances."](#)

Example A-1 Definition.xml

```
<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>

```

A.2 Instance.xml

The structure of the Instance.xml file created by the CAVS instance export feature is shown in [Example A-2](#).

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 [Chapter 12, "Exporting and Importing CAVS Definitions and Instances."](#)

Example A-2 Instance.xml

```

<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>
    </InstanceXpathsViewRORow>
  </InstanceXpathsViewRO>
</InstancesViewRORow>
</InstancesList>

```

```
        <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>
```

A

- asynchronous (notify) MEP testing flow
 - simulator definition, 2-5
 - test definition, 2-4
- asynchronous (two-way) MEP testing flow
 - simulator definition, 2-7
 - test definition, 2-5

B

- B2B errors, accessing, 19-1
- BPEL process, obtaining message XML, 2-10

C

- CAVS
 - asynchronous (notify) testing flows, 2-4
 - asynchronous (two-way) testing flows, 2-5
 - complex flow testing, 2-8
 - creating routing setup IDs, 8-3
 - creating simulator definitions, 5-1
 - creating test definitions, 4-1
 - definitions export structure, A-1
 - design assumptions, 1-3
 - exporting definitions, 12-1
 - exporting group instances, 12-4
 - exporting simulator instances, 12-3
 - exporting test instances, 12-3
 - flow testing, 2-8
 - group definitions, 3-1, 7-1
 - group instances, 3-1
 - importing definitions, 12-1
 - instances export structure, A-1
 - key components, 1-2
 - knowledge prerequisites, 1-3
 - modifying routing setup IDs, 8-6
 - modifying simulator definitions, 5-4
 - modifying test definitions, 4-5
 - obtaining message XML, 2-10
 - overview of defining tests, 3-3
 - overview of running tests, 3-3
 - process flows, 2-2
 - purging cross-reference entries, 11-1
 - routing setup IDs, 8-1
 - searching for routing setup IDs, 8-4

- setting up routing configurations, 8-7
- simulator definition key component, 1-3
- simulator definitions, 3-1
- simulator instances, 3-1
- synchronous testing flows, 2-2
- test definition key component, 1-2
- test definitions, 3-1
- test instances, 3-1
- test requirements, 2-1
- test scenarios, 2-1
- unit testing, 2-8
- web services, 3-4
- working with group instances, 10-1
- working with simulator instances, 9-1
- working with test instances, 9-1
- complex flow testing
 - using CAVS, 2-8
- Composite Application Validation System
 - See CAVS
- cross-references, purging CAVS-related, 11-1

D

- definition export structure, A-1
- dynamic responses, sending in simulator
 - response, 5-12

E

- error handling
 - accessing B2B errors, 19-1
 - associating email addresses with user roles, 14-5
 - BPEL system faults, 13-4
 - configuring, 14-5
 - configuring notification details, 14-5
 - configuring notification throttling
 - parameters, 15-3
 - creating user roles, 14-4
 - customizing notification email URLs, 15-11
 - customizing notification emails, 15-5
 - disabling notifications, 15-14
 - enabling notification throttling, 15-3
 - fault categories, 13-3
 - for B2B faults, 13-5
 - for business faults, 13-4
 - key features, 13-2

- mediator system faults, 13-4
- notification throttling overview, 15-2
- notifications, 15-1
- setting up, 14-1
- setting up notification throttling, 15-2
- using the Message Resubmission Utility, 17-1
- using the Oracle BPM Worklist, 16-1

Error Handling Framework
See also error handling

Error Handling Framework overview, 13-1

error logging

- accessing logs, 18-3
- enabling, 18-2
- overview, 18-1

error notifications

- configuring, 14-5
- configuring throttling parameters, 15-3
- customizing body text of emails, 15-9
- customizing email subject line, 15-8
- customizing emails, 15-5
- customizing URLs, 15-11
- disabling, 15-14
- enabling throttling, 15-3
- overview, 15-1
- setting up throttling, 15-2
- throttling overview, 15-2

F

faults

- B2B, 13-5
- BPEL system, 13-4
- business, 13-3
- mediator system, 13-4
- system, 13-3

flow testing using CAVS, 2-8

formats

- request message, 4-12
- response message, 4-13

G

gathering test requirements, 2-1

group definitions, 3-1

- creating, 7-2
- group instance selection, 7-4
- modifying, 7-2
- search group, 7-1
- test definition selection, 7-4
- working with, 7-1

group instances, 3-1

- exporting, 12-4
- viewing, 10-1
- viewing details, 10-2

I

instances, export structure, A-1

K

key components

- simulator definition, 1-3
- test definition, 1-2

L

logging

- accessing trace and error logs, 18-3
- searching for messages, 18-4
- using trace and error logs, 18-1

M

message exchange pattern

- asynchronous (notify) process flow, 2-4
- process flows, 2-2

Message Resubmission Utility

- overview, 17-1
- using, 17-1

message sets

- multiple requests and responses, 4-12
- provide multiple requests and responses in simulator definition, 5-9
- request message format, 4-12
- response message formats, 4-13

message XML, obtaining from a BPEL process, 2-10

N

notification details, configuring for error handling, 14-5

O

Oracle BPM Worklist

- enabling, 16-3
- overview, 16-1
- using, 16-3

P

process flows, testing the asynchronous (notify) MEP, 2-4

R

routing setup IDs

- creating, 8-3
- defining, 8-1
- modifying, 8-6
- routing setup actions, 8-6
- searching for, 8-4

S

simulator definitions, 1-3, 3-1

- asynchronous (notify) MEP testing flow, 2-4, 2-5
- asynchronous (two-way) MEP testing flow, 2-6, 2-7
- creating, 5-1

- modifying, 5-4
- providing multiple request and response message sets, 5-9
- searching for, 6-1
- supporting chatty services, 5-11
- synchronous MEP testing flow, 2-3
- working with, 6-1
- simulator instances, 3-1
 - exporting, 12-3
 - viewing details, 9-8
 - working with, 9-1
- simulator responses, sending dynamic responses, 5-12
- synchronous MEP testing flow
 - simulator definitions, 2-3
 - test definitions, 2-3

T

- test definitions, 1-2, 3-1
 - asynchronous (notify) MEP testing flow, 2-4
 - asynchronous (two-way) MEP testing flow, 2-5, 2-6
 - creating, 4-1
 - exporting, 12-1
 - importing, 12-1
 - modifying, 4-5
 - provide multiple request and response message sets, 4-12
 - searching for, 6-1
 - synchronous MEP testing flow, 2-2, 2-3
 - working with, 6-1
- test instances, 3-1
 - exporting, 12-3
 - viewing details, 9-4
 - working with, 9-1
- test requirements, gathering, 2-1
- testing flows
 - asynchronous (notify) testing flows, 2-4
 - asynchronous (two-way), 2-5
 - synchronous, 2-2
- tests
 - defining in CAVS, 3-3
 - running in CAVS, 3-3
- trace logging
 - accessing logs, 18-3
 - overview, 18-1
 - setting levels, 18-2

U

- unit testing using CAVS, 2-8
- user roles
 - associating email addresses, 14-5
 - creating for error handling, 14-4

W

- web service for CAVS, 3-4

