

JD Edwards EnterpriseOne Tools
Orchestrator Guide
Release 9.2.x
E65704-24

January 2019

Describes how to use the JD Edwards EnterpriseOne Orchestrator to support EnterpriseOne business process automation and support various integrations including integrations to Cloud services, third-party applications, custom programs, and many more.

Copyright © 2015, 2019, Oracle and/or its affiliates. All rights reserved.

Primary Author: Darcy Dahley

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, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

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 about 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 unless otherwise set forth in an applicable agreement between you and Oracle. 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, except as set forth in an applicable agreement between you and Oracle.

Contents

Preface	xv
Audience.....	xv
Documentation Accessibility	xv
Related Information.....	xv
Conventions.....	xvi
1 Understanding the JD Edwards EnterpriseOne Orchestrator	
1.1 JD Edwards EnterpriseOne Orchestrator Overview	1-1
1.1.1 Invoking Orchestrations and Notifications from an EnterpriseOne Interactive or Batch Application	1-1
1.1.2 Understanding an Internet of Things Integration.....	1-2
1.2 How It Works.....	1-2
1.3 EnterpriseOne Architecture for Orchestrator.....	1-3
2 Implementing the Orchestrator Studio	
2.1 Downloading and Installing Orchestrator Studio	2-1
2.2 Setting Up the Orchestrator Studio.....	2-2
2.2.1 Before You Begin	2-2
2.2.2 Define the Path to the AIS Server in the Orchestrator Studio	2-2
2.2.3 Test the EnterpriseOne Orchestrator Implementation	2-3
2.2.4 Set Up a Temporary Directory on the AIS Server for File Transfers.....	2-5
2.2.5 Upgrade Orchestration Components.....	2-5
3 Designing an Orchestration	
3.1 Understanding the Orchestration Design Process.....	3-1
3.2 Identifying the Problem and Solution	3-2
3.3 Identifying the Data for the Orchestration	3-2
3.4 Identifying the Rules for the Orchestration	3-3
3.5 Identifying the Cross Reference and White List Information for the Orchestration	3-3
3.6 Identifying the Service Request Information for the Orchestration.....	3-4
4 Creating Orchestrations with Orchestrator Studio 7.x.x.x	
4.1 Understanding the Orchestrator Studio and Orchestrations.....	4-2
4.1.1 Invoking an Orchestration	4-3

4.1.2	Reusable Orchestration Components.....	4-3
4.1.3	Orchestrations as User Defined Objects	4-3
4.2	Accessing the Orchestrator Studio.....	4-3
4.3	Navigating the Orchestrator Studio	4-4
4.3.1	Orchestrator Studio Design Page Features	4-5
4.3.2	User Defined Object (UDO) Features in the Orchestrator Studio	4-6
4.3.3	Working with the Graphical Representation of an Orchestration	4-7
4.4	Creating Service Requests	4-8
4.4.1	Understanding Service Requests	4-9
4.4.2	Creating a Service Request.....	4-10
4.4.3	Configuring a Form Request in the Orchestrator Studio	4-11
4.4.3.1	Loading EnterpriseOne Application Form Fields and Controls	4-11
4.4.3.2	Configuring Form Request Actions.....	4-12
4.4.3.3	Configuring the Order of Execution.....	4-16
4.4.3.4	Populating Multiple Grids with Repeating Inputs (Optional)	4-17
4.4.4	Creating a Form Request with the JD Edwards EnterpriseOne Orchestrator Process Recorder (Release 9.2.2.4)	4-17
4.4.4.1	Rules for Creating a Form Request in the Process Recorder	4-18
4.4.4.2	Prerequisites	4-18
4.4.4.3	Using the Process Recorder to Create a Form Request.....	4-18
4.4.5	Configuring a Data Request.....	4-19
4.4.5.1	Configuring a Data Request to Return Field Data.....	4-20
4.4.5.2	Configuring a Data Request with Data Aggregation.....	4-21
4.4.5.3	Viewing and Copying JSON Code of a Data Request	4-23
4.4.6	Configuring a Message Request	4-23
4.4.7	Configuring a Connector Service Request	4-25
4.4.7.1	Understanding Connector Service Requests	4-25
4.4.7.2	Before You Begin.....	4-26
4.4.7.3	Configuring a Connector Service Request to Invoke an Orchestration or Notification	4-26
4.4.7.4	Configuring a REST Connector to Invoke a REST Service.....	4-26
4.4.7.5	Configuring a REST Connector to Transfer Files to a REST Service	4-28
4.4.7.6	Configuring a Database Connector	4-29
4.4.7.7	Configuring a Connector to Transfer Files Using File Transfer Protocol (FTP)	4-30
4.4.8	Configuring a Watchlist Service Request	4-33
4.4.9	Configuring a Report Service Request	4-33
4.4.10	Configuring Form Request and Data Request Processing.....	4-35
4.4.11	Configuring a Custom Service Request with Java (Advanced).....	4-36
4.4.12	Configuring a Custom Service Request with Groovy (Advanced)	4-36
4.5	Creating Rules.....	4-38
4.5.1	Understanding Rules.....	4-38
4.5.2	Creating a Rule	4-38
4.5.3	Creating a Custom Rule with Java (Advanced)	4-39
4.5.4	Creating a Custom Rule with Groovy (Advanced)	4-40
4.6	Creating Cross References.....	4-41
4.6.1	Understanding Cross References	4-41
4.6.2	Creating a Cross Reference	4-42
4.7	Creating White Lists	4-43

4.7.1	Understanding White Lists	4-43
4.7.2	Creating a White List.....	4-43
4.8	Creating Orchestrations	4-44
4.8.1	Understanding Orchestrations	4-44
4.8.2	Creating an Orchestration	4-45
4.8.3	Adding Inputs to an Orchestration.....	4-46
4.8.4	Adding Steps to an Orchestration	4-48
4.8.4.1	Adding the Initial Step to an Orchestration	4-49
4.8.4.2	Adding Additional Steps to an Orchestration.....	4-49
4.8.4.3	Removing a Step from an Orchestration	4-49
4.8.4.4	Defining the Actions Between a Rule and Dependent Components	4-50
4.8.4.5	Defining Error Handling for Orchestration Steps (Orchestrator Studio 7.0.0.0)	4-51
4.8.5	Mapping Orchestration Inputs	4-52
4.8.6	Retrieving and Passing Data Sets in an Orchestration.....	4-53
4.8.6.1	Configuring a Data Request to Retrieve a Data Set	4-54
4.8.6.2	Configuring a Form Request to Retrieve a Data Set	4-54
4.8.6.3	Passing a Data Set to a Subsequent Orchestration Step	4-55
4.8.7	Working with Orchestration Output	4-55
4.9	Creating Schedules for Orchestrations	4-56
4.9.1	Understanding Schedules.....	4-56
4.9.2	Creating a Schedule	4-57
4.9.3	Adding a Schedule to an Orchestration	4-58
4.10	Updating Version 1 Orchestrations to Version 2 Orchestrations.....	4-58
4.11	Reloading Orchestrations and Orchestration Components	4-59
4.12	Supported Input Message Formats.....	4-59
4.12.1	Additional Supported Input Message Formats for the Generic Input Format	4-60
4.12.2	Differences Between Input Message Formats	4-60
4.13	Orchestration Security Considerations	4-61
4.13.1	Restricting Access to Exposed Orchestrations	4-62
4.13.2	How to Maintain a Single Session for Multiple Calls to an Orchestration	4-62
4.14	Exporting Orchestration Components from the Orchestrator Studio	4-62
4.15	Importing Orchestration Files in the Orchestrator Studio.....	4-63

5 Setting Up Cross References and White Lists in EnterpriseOne (P952000)

6 Orchestrator Health and Exception Monitoring (Release 9.2.3)

6.1	Understanding the EnterpriseOne Orchestrator Monitor	6-1
6.2	Prerequisites.....	6-2
6.3	Accessing the Orchestrator Monitor	6-2
6.4	Monitoring UDOs Based on Different EnterpriseOne User, Environment, or Role	6-2
6.5	Refreshing the Data Displayed in the Orchestrator Monitor	6-2
6.6	Resetting the Data Displayed in the Orchestrator Monitor.....	6-3
6.7	Monitoring Orchestrator Health	6-3
6.7.1	Filtering the UDOs Displayed in the Health Tab	6-3
6.7.2	Understanding Orchestrator Health Data	6-3
6.8	Monitoring Orchestrator Exceptions.....	6-5

6.8.1	Possible Causes of Orchestrator Exceptions.....	6-5
6.8.2	Viewing and Changing Exceptions Displayed in the List View	6-6
6.8.3	Reviewing General Exception Information in the List View	6-6
6.8.4	Reviewing Detailed Exception Information in the List View	6-7
6.8.5	Understanding Orchestration Step Trace Details	6-8
6.8.6	Viewing Exceptions in the Chart View	6-8
6.8.7	Changing How Exceptions Are Displayed in the Chart View	6-9
6.8.8	Accessing Detailed Information from the Chart.....	6-10
6.9	Managing Orchestrator Health and Exception Records in EnterpriseOne	6-10
6.9.1	Managing Health Records in EnterpriseOne	6-10
6.9.2	Managing Exception Records in EnterpriseOne.....	6-11

7 Creating Custom Java for Orchestration

7.1	Understanding Custom Java for Orchestration	7-1
7.2	Creating Custom Java	7-1
7.2.1	Using Custom Java for the Orchestration Service Request	7-2
7.2.2	Using Custom Java for the Orchestration Rule	7-2
7.3	Deploying Custom Java	7-2
7.3.1	Deploying Custom Java on AIS Server on Oracle WebLogic Server.....	7-2
7.3.2	Deploying Custom Java on AIS Server on IBM WebSphere Application Server	7-3

8 Using Apache Groovy for Custom Service Requests, Rules, and Manipulating Output (Orchestrator Studio 5.1.0 and Higher)

8.1	Understanding Groovy for Orchestration Components	8-1
8.2	Groovy Template for a Custom Service Request	8-2
8.3	Groovy Template for a Custom Rule.....	8-2
8.4	Groovy Template for Manipulating Output from a REST Connector Response.....	8-3
8.5	Groovy Template for Manipulating Output from an Orchestration Response	8-4
8.6	Additional Attributes and Methods Available in the Groovy Script Templates	8-5

9 Testing Orchestrations in the EnterpriseOne Orchestrator Client

9.1	Understanding the Orchestrator Client	9-1
9.1.1	Using cURL to Simulate Testing from a Third-Party Application or IoT Device	9-2
9.2	Testing an Orchestration in the EnterpriseOne Orchestrator Client	9-2

10 Administering the Orchestrator Studio and Orchestrations

10.1	Understanding Orchestration Life Cycle Management (Release 9.2.1)	10-1
10.2	Setting Up User Access to the Orchestrator Studio	10-2
10.2.1	Setting Up Allowed Users	10-2
10.2.2	Setting Up UDO Security for Orchestrator Studio Users (Release 9.2.1).....	10-2
10.3	Upgrading Orchestration Components to User Defined Objects (Release 9.2.1).....	10-4
10.4	Clearing Orchestration Cache on the AIS Server	10-4
10.4.1	Activating the AIS Administration Service	10-5
10.4.2	Using the AIS Administration Service to Clear Cache	10-5
10.5	Creating Connection Soft Coding Records for Connector Service Requests	10-6
10.5.1	Understanding Connection Soft Coding Records.....	10-6

10.5.2	Creating a Soft Coding Record for an Orchestrator Connection	10-7
10.5.3	Creating a Soft Coding Record for a REST Connection	10-8
10.5.4	Creating a Soft Coding Record for a Database Connection	10-9
10.5.5	Creating a Soft Coding Record for an FTP Server Connection (Orchestrator Studio 6.1.0)	10-10
10.6	Setting Up Orchestration Error and Exception Tracking (Release 9.2.2.4).....	10-10
10.6.1	Understanding Orchestration Error Files.....	10-11
10.7	Setting Up Orchestrator Health and Exception Monitoring (Release 9.2.3)	10-12
10.7.1	Downloading and Installing the Orchestrator Monitor.....	10-12
10.7.2	Enabling Orchestrator Health and Exception Tracking in Server Manager	10-13
10.7.3	Set Up User Access to the Orchestrator Monitor and Orchestrator Health and Exceptions Program	10-14
10.7.4	Enable UDO View Security to Monitor Orchestrations, Notifications, and Schedules	10-14

11 Understanding the AIS Server Discovery Service

A Creating Orchestrations with Orchestrator Studio 6.x.x

A.1	Understanding the Orchestrator Studio and Components.....	A-2
A.1.1	Invoking an Orchestration	A-3
A.1.2	Reusable Orchestration Components.....	A-3
A.1.3	Orchestrations as User Defined Objects	A-3
A.2	Accessing the Orchestrator Studio.....	A-3
A.3	Navigating the Orchestrator Studio	A-4
A.3.1	Orchestrator Studio Design Page Features	A-5
A.3.2	User Defined Object (UDO) Features in the Orchestrator Studio.....	A-6
A.3.3	Working with the Graphical Representation of an Orchestration	A-7
A.4	Creating Service Requests	A-8
A.4.1	Understanding Service Requests	A-9
A.4.1.1	Understanding the Application Stack Option in a Form Request	A-10
A.4.2	Creating a Service Request.....	A-10
A.4.3	Configuring a Form Request in the Orchestrator Studio	A-11
A.4.3.1	Load the EnterpriseOne Application Form Fields and Controls	A-11
A.4.3.2	Configure Form Request Actions	A-12
A.4.3.3	Configure the Order of Execution	A-16
A.4.3.4	Populating Multiple Grids with Repeating Inputs.....	A-17
A.4.4	Creating a Form Request with the JD Edwards EnterpriseOne Process Recorder (Tools 9.2.2.4)	A-17
A.4.4.1	Rule for Creating a Form Request in the Process Recorder.....	A-18
A.4.4.2	Prerequisites	A-18
A.4.4.3	Using the Process Recorder to Create a Form Request	A-18
A.4.5	Configuring a Data Request.....	A-19
A.4.5.1	Configuring a Data Request to Return Field Data.....	A-19
A.4.5.2	Configuring a Data Request with Data Aggregation.....	A-21
A.4.5.3	Viewing and Copying JSON Code of a Data Request	A-23
A.4.6	Configuring a Message Request	A-23
A.4.7	Configuring a Connector Service Request	A-24

A.4.7.1	Understanding Connector Service Requests	A-25
A.4.7.2	Before You Begin.....	A-25
A.4.7.3	Configuring a Connector Service Request to Invoke an Orchestration or Notification	A-25
A.4.7.4	Configuring a REST Connector to Invoke a REST Service.....	A-26
A.4.7.5	Configuring a REST Connector to Transfer Files to a REST Service (Orchestrator Studio 6.1.0).....	A-28
A.4.7.6	Configuring a Database Connector (Orchestrator Studio 6.0.1)	A-29
A.4.7.7	Configuring a Connector to Transfer Files Using File Transfer Protocol (FTP) (Orchestrator Studio 6.1.0).....	A-29
A.4.8	Configuring a Watchlist Service Request (Orchestrator Studio 6.0.1.0).	A-31
A.4.9	Configuring a Report Service Request (Orchestrator Studio 6.1.0.0)	A-32
A.4.10	Configuring Form Request and Data Request Processing.....	A-33
A.4.11	Configuring a Custom Service Request with Java (Advanced).....	A-34
A.4.12	Configuring a Custom Service Request with Groovy (Advanced)	A-34
A.5	Creating Rules.....	A-36
A.5.1	Understanding Rules.....	A-36
A.5.2	Creating a Rule	A-37
A.5.3	Creating a Custom Rule with Java (Advanced).....	A-38
A.5.4	Creating a Custom Rule with Groovy (Advanced)	A-38
A.6	Creating Cross References.....	A-40
A.6.1	Understanding Cross References	A-40
A.6.2	Creating a Cross Reference	A-40
A.7	Creating White Lists	A-41
A.7.1	Understanding White List.....	A-41
A.7.2	Creating a White List.....	A-41
A.8	Creating Orchestrations	A-42
A.8.1	Understanding Orchestrations	A-42
A.8.2	Creating an Orchestration	A-43
A.8.3	Adding Inputs to an Orchestration.....	A-44
A.8.4	Adding Steps to an Orchestration	A-46
A.8.4.1	Defining the Actions Between a Rule and Dependent Components	A-48
A.8.5	Mapping Orchestration Inputs	A-49
A.8.6	Retrieving and Passing Data Sets in an Orchestration (Orchestrator Studio 6.0.x)	A-50
A.8.6.1	Configuring a Data Request to Retrieve a Data Set	A-51
A.8.6.2	Configuring a Form Request to Retrieve a Data Set	A-51
A.8.6.3	Passing a Data Set to a Subsequent Orchestration Step	A-52
A.8.7	Working with Orchestration Output	A-52
A.9	Creating Schedules for Orchestrations (Orchestrator Studio 6.0.x)	A-53
A.9.1	Understanding Schedules.....	A-53
A.9.2	Creating a Schedule	A-54
A.9.3	Adding a Schedule to an Orchestration	A-55
A.10	Updating Version 1 Orchestrations to Version 2 Orchestrations.....	A-55
A.11	Reloading Orchestrations and Orchestration Components	A-56
A.12	Supported Input Message Formats.....	A-56
A.12.1	Additional Supported Input Message Formats for the Generic Input Format	A-57
A.12.2	Differences Between Input Message Formats	A-57

A.13	Orchestration Security Considerations	A-58
A.13.1	Restricting Access to Exposed Orchestrations	A-59
A.13.2	How to Maintain a Single Session for Multiple Calls to an Orchestration	A-59
A.14	Exporting Orchestration Components from the Orchestrator Studio	A-59
A.15	Importing Orchestration Files in the Orchestrator Studio.....	A-60

B Creating Orchestrations with Orchestrator Studio 5.1.0

B.1	Understanding the Orchestrator Studio and Orchestrations.....	B-2
B.1.1	Reusable Orchestration Components.....	B-3
B.1.2	Orchestrations as User Defined Objects	B-3
B.2	Accessing the Orchestrator Studio.....	B-3
B.3	Navigating the Orchestrator Studio	B-4
B.4	Working with Orchestrator Studio Design Pages	B-4
B.4.1	Design Page Features.....	B-5
B.4.2	User Defined Object (UDO) Features	B-5
B.4.3	Working with the Graphical Representation of an Orchestration	B-7
B.5	Creating Service Requests	B-8
B.5.1	Understanding Service Requests	B-8
B.5.1.1	Understanding the Application Stack Option in a Form Request	B-9
B.5.2	Creating a Service Request.....	B-9
B.5.3	Configuring a Form Request	B-10
B.5.3.1	Populating Multiple Grids with Repeating Inputs.....	B-15
B.5.4	Configuring a Custom Service Request with Java (Advanced).....	B-15
B.5.5	Configuring a Custom Service Request with Groovy (Advanced) (Studio 5.1.0)	B-15
B.5.6	Configuring a Data Request.....	B-17
B.5.6.1	Viewing and Copying JSON Code of a Data Request	B-20
B.5.7	Configuring a Message Request	B-20
B.5.8	Configuring a Connector	B-21
B.5.8.1	Configuring a Connector to Invoke an Orchestration	B-22
B.5.8.2	Configuring a Connector to Invoke a REST Service (Studio 5.1.0)	B-22
B.5.9	Configuring Form Request and Data Request Processing.....	B-24
B.6	Creating Rules.....	B-25
B.6.1	Understanding Rules.....	B-25
B.6.2	Creating a Rule	B-25
B.6.3	Creating a Custom Rule with Java (Advanced)	B-26
B.6.4	Creating a Custom Rule with Groovy (Advanced) (Studio 5.1.0)	B-27
B.7	Creating Cross References	B-28
B.7.1	Understanding Cross References	B-28
B.7.2	Creating a Cross Reference	B-29
B.8	Creating White Lists	B-30
B.8.1	Understanding White Lists	B-30
B.8.2	Creating a White List.....	B-30
B.9	Creating Orchestrations	B-31
B.9.1	Understanding Orchestrations	B-31
B.9.2	Creating an Orchestration	B-32
B.9.3	Adding Inputs to an Orchestration.....	B-33
B.9.4	Adding Steps to an Orchestration	B-33

B.9.4.1	Defining the Actions Between a Rule and Dependent Components	B-35
B.9.5	Mapping Inputs in the Transformations Area.....	B-36
B.9.6	Updating Version 1 Orchestrations to Version 2 Orchestrations	B-38
B.9.7	Working with Orchestration Output	B-38
B.10	Reloading Orchestrations and Orchestration Components	B-40
B.11	Supported Input Message Formats.....	B-40
B.11.1	Additional Supported Input Message Formats for the Generic Input Format	B-41
B.11.2	Differences Between Input Message Formats	B-41
B.12	Setting Up Orchestration Security	B-42
B.12.1	Restricting Access to Exposed Orchestrations	B-43
B.12.2	How to Maintain a Single Session for Multiple Calls to an Orchestration	B-43
B.13	Exporting Orchestration Components from the Orchestrator Studio	B-43
B.14	Importing Orchestration Files in the Orchestrator Studio.....	B-44

C Creating Orchestrations with Orchestrator Studio 3.0.1 (Release 9.2.1)

C.1	Understanding the Orchestrator Studio and Orchestrations.....	C-2
C.1.1	Reusable Orchestration Components.....	C-3
C.1.2	Orchestrations as User Defined Objects	C-3
C.2	Accessing the Orchestrator Studio.....	C-3
C.3	Navigating the Orchestrator Studio	C-3
C.4	Working with the Orchestrator Studio Design Pages	C-4
C.4.1	Design Page Features.....	C-4
C.4.2	User Defined Object (UDO) Features	C-5
C.4.3	Working with the Graphical Representation of an Orchestration	C-6
C.5	Creating Service Requests	C-8
C.5.1	Understanding Service Requests	C-8
C.5.1.1	Understanding the Application Stack Option.....	C-8
C.5.2	Understanding the Service Request Design Page	C-9
C.5.3	Creating a Service Request.....	C-11
C.5.4	Configuring the Actions in the Service Request.....	C-11
C.5.5	Defining the Order of Execution in the Service Request	C-14
C.5.6	Creating a Custom Java Service Request	C-14
C.6	Creating Rules.....	C-15
C.6.1	Understanding Rules.....	C-15
C.6.2	Creating a Rule	C-15
C.6.3	Adding a Custom Java Rule	C-16
C.7	Creating Cross References	C-17
C.7.1	Understanding Cross References	C-17
C.7.2	Creating a Cross Reference	C-17
C.8	Creating White Lists	C-18
C.8.1	Understanding White Lists	C-18
C.8.2	Creating a White List.....	C-18
C.9	Creating Orchestrations	C-19
C.9.1	Understanding Orchestrations	C-19
C.9.2	Creating an Orchestration	C-20
C.9.3	Adding Inputs to an Orchestration.....	C-21
C.9.4	Adding Steps to an Orchestration	C-21

C.9.4.1	Defining the Actions Between a Rule and Dependent Components.....	C-23
C.9.5	Configuring Transformations	C-24
C.10	Reloading Orchestrations and Orchestration Components.....	C-25
C.11	Supported Input Message Formats.....	C-26
C.11.1	Additional Supported Input Message Formats for the Generic Input Format.....	C-26
C.11.2	Differences Between Input Message Formats	C-27
C.12	Setting Up Orchestration Security	C-28
C.12.1	Restricting Access to Exposed Orchestrations	C-28
C.13	Exporting Orchestration Components from the Orchestrator Studio	C-29
C.14	Importing Orchestration Files in the Orchestrator Studio.....	C-29

D Creating Orchestrations with the Orchestrator Studio 2.0 (Release 9.2.0.5)

D.1	Understanding the Orchestrator Studio and Orchestrations.....	D-1
D.1.1	Reusable Orchestration Components.....	D-2
D.2	Accessing the Orchestrator Studio.....	D-3
D.3	Navigating the Orchestrator Studio	D-3
D.3.1	Navigating Orchestrator Studio Design Pages	D-4
D.3.2	Working with the Graphical Representation of an Orchestration	D-5
D.4	Creating Service Requests	D-7
D.4.1	Understanding Service Requests	D-7
D.4.1.1	Understanding the Application Stack Option.....	D-7
D.4.2	Understanding the Service Request Design Page	D-8
D.4.3	Creating a Service Request.....	D-10
D.4.4	Configuring the Actions in the Service Request.....	D-10
D.4.5	Defining the Order of Execution in a Service Request.....	D-13
D.4.6	Creating a Custom Java Service Request	D-13
D.5	Creating Rules.....	D-14
D.5.1	Understanding Rules.....	D-14
D.5.2	Creating a Rule	D-14
D.5.3	Adding a Custom Java Rule	D-15
D.6	Creating Cross References	D-16
D.6.1	Understanding Cross References	D-16
D.6.2	Creating a Cross Reference	D-16
D.7	Creating White Lists	D-17
D.7.1	Understanding White Lists	D-17
D.7.2	Creating a White List.....	D-17
D.8	Creating Orchestrations	D-18
D.8.1	Understanding Orchestrations	D-18
D.8.2	Creating an Orchestration	D-19
D.8.3	Adding Inputs to an Orchestration.....	D-19
D.8.4	Adding Steps to an Orchestration	D-20
D.8.4.1	Defining the Actions Between a Rule and Dependent Components	D-22
D.8.5	Configuring Transformations	D-23
D.9	Reloading Orchestrations and Orchestration Components	D-24
D.10	Setting up Orchestration Security	D-25
D.11	Exporting Files from the Orchestrator Studio.....	D-25
D.12	Importing Orchestration Files in the Orchestrator Studio.....	D-25

E Creating Orchestration with the Orchestrator Studio (Release 9.2.0.2)

E.1	Understanding the Orchestrator Studio and Orchestration.....	E-1
E.1.1	Reusing or Copying Orchestration Components	E-3
E.2	Accessing the Orchestrator Studio.....	E-3
E.3	Creating Orchestration	E-4
E.3.1	Understanding Orchestration	E-4
E.3.2	Creating an Orchestration	E-5
E.3.3	Adding Inputs to an Orchestration.....	E-5
E.3.4	Adding Steps to an Orchestration	E-6
E.3.5	Completing an Orchestration.....	E-8
E.4	Creating Service Requests	E-8
E.4.1	Understanding Service Requests	E-9
E.4.1.1	Understanding the Application Stack Option.....	E-9
E.4.2	Understanding the Service Request Design Panel.....	E-9
E.4.3	Creating a Service Request.....	E-12
E.4.4	Configuring the Actions in the Service Request.....	E-12
E.4.5	Defining the Order of Execution in the Service Request	E-14
E.4.6	Adding an Existing Service Request to an Orchestration.....	E-14
E.4.7	Adding a Custom Java Service Request	E-15
E.5	Creating Rules.....	E-16
E.5.1	Understanding Rules.....	E-16
E.5.1.1	Understanding Nested Rules (and Nested Service Requests)	E-16
E.5.2	Creating a Rule	E-16
E.5.3	Adding an Existing Rule to an Orchestration	E-17
E.5.4	Adding a Custom Java Rule	E-18
E.6	Creating Cross References	E-18
E.6.1	Understanding Cross References	E-19
E.6.2	Creating a Cross Reference	E-19
E.6.3	Adding an Existing Cross Reference to an Orchestration	E-20
E.7	Creating White Lists	E-20
E.7.1	Understanding White Lists	E-20
E.7.2	Creating a White List.....	E-21
E.7.3	Adding an Existing White List to an Orchestration.....	E-21
E.8	Reloading Orchestration and Orchestration Components	E-22

F Sample Orchestration

F.1	Prerequisite	F-1
F.2	Running the Sample Orchestration.....	F-1
F.3	Add Conditioned Based Maintenance Alert Sample Orchestration	F-2
F.3.1	Sample Input	F-3
F.4	Update Equipment Location Sample Orchestration	F-3
F.4.1	Sample Input	F-4
F.5	Update Meter Reading Sample Orchestration	F-4
F.5.1	Sample Input	F-4

G Troubleshooting

G.1	Enable Debugging on the AIS Server.....	G-1
G.2	Troubleshooting Orchestration Runtime Issues.....	G-1

Preface

Welcome to the *JD Edwards EnterpriseOne Tools Orchestrator Guide*. The main chapters in this guide contain instructions for Orchestrator Studio 7.1.x.x, the latest version of the Orchestrator Studio supported with a minimum release of EnterpriseOne Tools 9.2.3.2.

Instructions for prior releases of the Orchestrator Studio have been moved to the appendices in this guide.

The IoT Orchestrator has been renamed to "Orchestrator." Although the Orchestrator is an integral part of an IoT configuration with JD Edwards EnterpriseOne, the Orchestrator functionality has expanded to support various integrations including integrations to Cloud services, third-party applications, custom programs, and many more. Keep using it for integrating JD Edwards EnterpriseOne with IoT devices. But start using it for all your REST-based integrations to EnterpriseOne.

Audience

This guide is intended for business analysts or project managers responsible for configuring orchestrations for data integration with JD Edwards EnterpriseOne. It is also intended for an administrator responsible for setting up the Orchestrator and Orchestrator Studio as described in the [Chapter 2, "Implementing the Orchestrator Studio"](#) in this guide.

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 that have purchased support 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.

Related Information

For additional information about JD Edwards EnterpriseOne applications, features, content, and training, visit the JD Edwards EnterpriseOne pages on the JD Edwards Resource Library located at:

<http://learnjde.com>

This guide references related information in the following guides:

- *JD Edwards EnterpriseOne Tools Server Manager Guide*
- *JD Edwards EnterpriseOne Application Interface Services Server Reference Guide*
- *JD Edwards EnterpriseOne Application Interface Services Client Java API Developer's Guide*
- *JD Edwards EnterpriseOne Tools System Overview Guide*
- *JD Edwards EnterpriseOne Tools Interoperability Guide*

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.
<code>monospace</code>	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.
> Tutorial	Indicates a link to a recording of the described feature. These recordings are in MP4 format so ensure that you have an appropriate player installed. Access to these recordings requires a valid Oracle account.

Understanding the JD Edwards EnterpriseOne Orchestrator

This chapter contains the following topics:

- [Section 1.1, "JD Edwards EnterpriseOne Orchestrator Overview"](#)
- [Section 1.2, "How It Works"](#)
- [Section 1.3, "EnterpriseOne Architecture for Orchestrator"](#)

1.1 JD Edwards EnterpriseOne Orchestrator Overview

The JD Edwards EnterpriseOne Orchestrator is an integral part of an Internet of things (IoT) configuration with EnterpriseOne and supports other various integrations such as integrations with Cloud services, third-party applications, custom programs, and more. The Orchestrator uses the JD Edwards EnterpriseOne Application Interface Services (AIS) Server as its foundation. It processes orchestrations saved to the AIS Server to transform external data into actionable business processes in EnterpriseOne.

The JD Edwards EnterpriseOne Orchestrator Studio is an intuitive, easy to use web-based application that business analysts can use to create orchestrations and notifications.

The combined power of the JD Edwards EnterpriseOne Orchestrator and AIS Server enables your organization to:

- **Connect IoT data to your business data.** The Orchestrator enables customers to collect, filter, analyze, and act on real-time data as it is being transmitted by IoT devices. JD Edwards EnterpriseOne customers benefit by eliminating costly and error-prone manual processes, by reacting to—or avoiding—business disruptions in real-time, and by analyzing historical data for continuous process improvement.
- **Extend integrations to other solutions.** Starting with EnterpriseOne Tools 9.2.1.4 and Orchestrator Studio 5.1.0, the Orchestrator functionality has expanded to support various integrations including integrations to Cloud services, third-party applications, custom programs, and many more.
- **Transform how you use your entire JD Edwards system.** The Orchestrator enables you to free your EnterpriseOne users from tedious tasks by transforming manual EnterpriseOne business processes into automated operations that meet your specific business needs.

1.1.1 Invoking Orchestrations and Notifications from an EnterpriseOne Interactive or Batch Application

Starting with EnterpriseOne Tools 9.2.3, EnterpriseOne provides the B98ORCH business function for invoking an orchestration or notification from an event in EnterpriseOne. This

capability enables developers to configure EnterpriseOne to automatically launch an orchestration or notification from one of the following events:

- ❑ A table trigger
- ❑ A button on a form
- ❑ Changing the status of an event
- ❑ A UBE

With the B98ORCH business function, you can extend EnterpriseOne applications beyond transactional boundaries by linking disparate ERP tasks into a single, automated business process. Or you can enable an EnterpriseOne application to automatically pass data to a REST-enabled third-party application via an orchestration. Or you can enable an EnterpriseOne application to automatically send notifications to a group of users.

For more information, see "Configuring the B98ORCH Business Function to Invoke an Orchestration or Notification" in the *JD Edwards EnterpriseOne Tools APIs and Business Functions Guide*.

1.1.2 Understanding an Internet of Things Integration

Companies use devices such as sensors and beacons ("things") to monitor everything from the performance of machinery, temperatures of refrigerated units, on-time averages of commuter trains, and so forth. Capturing the data from these devices traditionally involves a complex integration using specialized hardware, expensive network connectivity, and high system integration expenditure to build the machine information into an enterprise business process. Even with a complex integration in place, an operations manager or controller still might have to manually enter the data into a spreadsheet, a software program, or an application in an ERP system. Regardless of the method used, it takes time to transfer the raw data into information that can be acted upon in a way that provides value to the business.

With the JD Edwards EnterpriseOne Orchestrator Studio, you can create orchestrations that enable the transformation of data from disparate devices into actionable business processes in JD Edwards EnterpriseOne. For example, you can create orchestrations that enable EnterpriseOne to:

- ❑ Alert users to a required activity.
- ❑ Alert users to perform preventative maintenance to reduce equipment downtime.
- ❑ Provide audit data for safety compliance and security.

1.2 How It Works

When a new orchestration is saved in the Orchestrator Studio, the name of orchestration is used to define an endpoint on the AIS Server. The endpoint URL is:

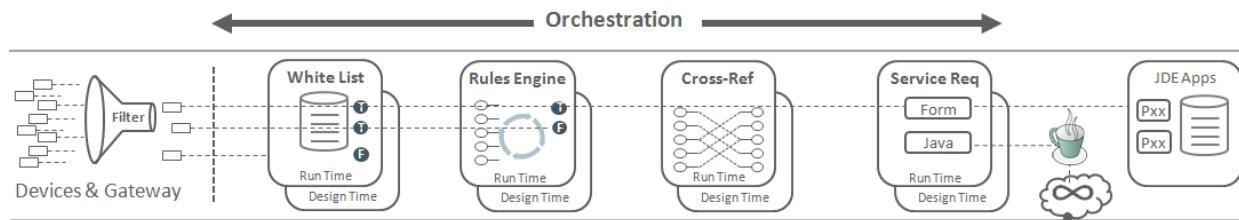
`http://<server>:<port>/jderest/orchestrator/<orchestrationname>`

To invoke an orchestration, calling applications or devices use a post operation to this URL, where <orchestrationname> is the name of your orchestration. The post operation must include security parameters that enable access to the orchestration and any EnterpriseOne application invoked by the orchestration. See [Orchestration Security Considerations](#) for more information.

The illustration in [Figure 1–1](#) shows how the EnterpriseOne Orchestrator processes orchestrations. The illustration depicts processing an orchestration designed to receive data from external devices in an IoT configuration, but you can also use orchestrations to consume

and provide data to Cloud services, third-party applications, custom EnterpriseOne programs, and more.

Figure 1–1 EnterpriseOne Orchestration Processing



Third-party devices and a gateway collect and process information from one or more devices, converts the information to a platform-independent format and communicates this information over the internet. The gateway typically deploys intelligence to filter sensor data, secure data transfer, automate software updating, run diagnostics, start or stop the device, and support other features.

The Orchestrator uses the following orchestration components, created in the Orchestrator Studio, to transform incoming raw data into data that can be used by EnterpriseOne:

- **Orchestration.** The master process that defines the inputs for the orchestration and provides a unique name for the orchestration process in the Orchestrator. An orchestration uses the next four components in this list to run a single orchestration instance.
- **Service request.** An invocation of a JD Edwards EnterpriseOne interactive application or a Java application via a REST service call to the EnterpriseOne Application Interface Services (AIS) Server.
- **Rule.** Contains a set of conditions against which the orchestration input is evaluated to produce a true or false state. Rules can be nested to produce complex evaluations. Rules determine how the orchestration is processed at runtime. You can also use custom Java to define additional rules.
- **Cross reference.** A set of data relationships defined by the designer of the orchestration that enriches the minimal input from devices. For example, a device's serial number can be cross-referenced to a JD Edwards EnterpriseOne Equipment Number for use in service requests.
- **White list.** An initial rudimentary pass/fail check of the incoming message's device signature against a predefined list of signatures. A white list provides an additional layer of security to the Orchestrator security.

You can also use custom Java to create custom client applications (that run on the AIS Server) for viewing and working with the filtered data. You can create a custom Java application to perform a specific business process or a process for storing the data in another database outside of EnterpriseOne.

1.3 EnterpriseOne Architecture for Orchestrator

The Orchestrator uses the JD Edwards EnterpriseOne Application Interface Services (AIS) Server as its foundation. The AIS Server is a REST services server that when configured with the EnterpriseOne HTML Server, enables access to EnterpriseOne forms and data. The Orchestrator processes orchestrations saved to the AIS Server to transfer data between EnterpriseOne and third-party applications and devices.

For an illustration of the AIS Server architecture, see "AIS Server Architecture" in the *JD Edwards EnterpriseOne Tools System Overview Guide*.

Implementing the Orchestrator Studio

This chapter describes how to implement the Orchestrator Studio. It contains the following topics:

- Section 2.1, "Downloading and Installing Orchestrator Studio"
- Section 2.2, "Setting Up the Orchestrator Studio"

Important: This chapter has been updated in support of Orchestrator Studio 7.0.x.0, which requires a minimum of EnterpriseOne Tools 9.2.3.

2.1 Downloading and Installing Orchestrator Studio

Install the Orchestrator Studio through Server Manager. See "Installing an EnterpriseOne Orchestrator Studio" in the *JD Edwards EnterpriseOne Tools Server Manager Guide*, which contains prerequisites and instructions for downloading and installing the Orchestrator Studio.

API and JAR Files for Developing Custom Java for Orchestration

In addition to files for installing the Orchestrator Studio, the Orchestrator download includes files that developers can use to create custom Java for orchestrations. This is an advanced activity that is not required to create orchestrations. The files include:

- AIS_Client_Java_API_2.x.x. This is the API for developing custom Java.
- OrchestratorCustomJava.jar. This file contains the definition of the interfaces for developing custom Java.

See Chapter 7, "Creating Custom Java for Orchestrations" for more information.

Sample Orchestration Files

Oracle provides sample orchestration files that you can use to test your Orchestrator Studio implementation or use as a template for creating your own orchestrations. After you install Orchestrator Studio, you can download Orchestration_Samples_1.1.1 from the Update Center on My Oracle Support (<https://updatecenter.oracle.com/>). In the Update Center, select **EnterpriseOne IoT Orchestrator** in the Type field to locate the download.

The download includes:

- Sample orchestration files:
 - Add Condition Based Maintenance Alert
 - Update Equipment Locations
 - Update Meter Readings

- ❑ Data Pack for EnterpriseOne Applications. This is required for running the sample orchestration files in an EnterpriseOne test environment.

See [Test the EnterpriseOne Orchestrator Implementation](#) for how to use these sample orchestrations to test running orchestrations in an EnterpriseOne environment with pristine data.

You can also use them as a template for creating your own orchestrations. See [Appendix F, "Sample Orchestrations"](#) for more information.

Prebuilt EnterpriseOne Orchestrations

Oracle also provides prebuilt EnterpriseOne orchestrations that are available for download from the Update Center. You can use these prebuilt orchestrations as examples for designing your own orchestrations based on your business requirements. For more information, see "JD Edwards EnterpriseOne Orchestrations" in the *JD Edwards EnterpriseOne Applications Business Interfaces Implementation Guide*.

2.2 Setting Up the Orchestrator Studio

Perform the tasks in the following sections to set up the Orchestrator Studio:

- ❑ [Before You Begin](#)
- ❑ [Define the Path to the AIS Server in the Orchestrator Studio](#)
- ❑ [Test the EnterpriseOne Orchestrator Implementation](#)
- ❑ [Upgrade Orchestration Components](#)

2.2.1 Before You Begin

Orchestrator Studio access requires a separate sign-in with a valid EnterpriseOne user ID and password. Before users can sign in to Orchestrator Studio, an administrator must add each EnterpriseOne user to a list of allowed users in Server Manager. See [Setting Up Allowed Users](#) in this guide.

Also, orchestrations and orchestration components created in the Orchestrator Studio are saved and managed as user defined objects (UDOs) in EnterpriseOne. Therefore, an administrator must set up Orchestrator Studio users with the proper UDO security to access and use the Orchestrator Studio. See [Setting Up UDO Security for Orchestrator Studio Users \(Release 9.2.1\)](#) in this guide.

Note: Setting up users with access to the Orchestrator Studio automatically provides users with access to the Orchestrator Client, a standalone web application for testing orchestrations. You can access the Orchestrator Client from the Orchestrator Studio.

2.2.2 Define the Path to the AIS Server in the Orchestrator Studio

To use the Orchestrator Studio, you must first launch the Orchestrator Studio and define the path to the AIS Server.

1. Open a browser and enter the following URL to access the Orchestrator Studio configuration page:

`http://<ADFSERVERHOST>:<port>/OrchestratorStudio/faces/index.jsf`

You can locate the domain and port number for the Orchestrator Studio in WebLogic Admin Console, inside the Studio deployment settings, under Test.

2. On the Orchestrator Studio configuration page, enter a URL to the AIS Server in a format that includes the AIS Server name and port number, for example:

`http://<aisserver>:<port>`

If the AIS Server uses an SSL configuration, then use the https protocol for the URL.

2.2.3 Test the EnterpriseOne Orchestrator Implementation

Use the Orchestrator Client, a standalone web application that you can access from the Orchestrator Studio, to test your EnterpriseOne implementation.

Note: The Orchestrator Client is also used for testing custom orchestrations that you create before deploying them in a production environment. See [Chapter 9, "Testing Orchestrations in the EnterpriseOne Orchestrator Client"](#) in this guide for more information.

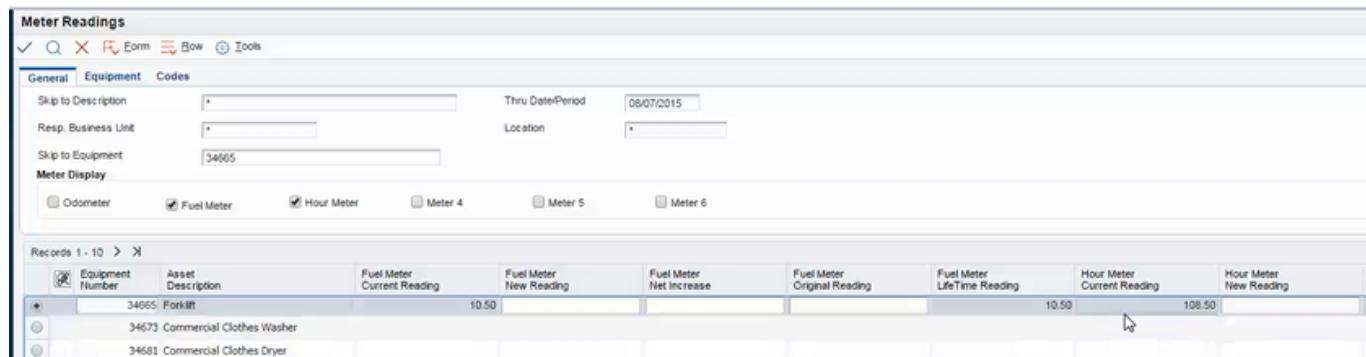
Before you can test the implementation, you must use the Change Assistant in EnterpriseOne to apply the data pack included in the Orchestration Samples download. See "Working with Packages" in the *JD Edwards EnterpriseOne Tools Software Updates Guide* on how to apply updates through the Change Assistant.

The steps in this section use the JDE_ORCH_Sample_UpdateMeterReadings sample orchestration in the JDE_IOT_Orchestrator_XML directory to test your implementation. The JDE_ORCH_Sample_UpdateMeterReadings sample orchestration is designed to update a meter reading record in the Meter Readings program (P12120) in EnterpriseOne.

Before you test the implementation, review the current equipment number record in EnterpriseOne:

1. Access P12120 in EnterpriseOne.
2. In the Skip to Equipment field, enter 34665.
3. Select the **Fuel Meter** and **Hour Meter** check boxes.
4. Click the **Find** button.

In the record for equipment number 34665, notice the values in the Fuel Meter Current Reading and Hour Meter Current Reading.



Next, test the Orchestrator implementation by running the JDE_ORCH_Sample_UpdateMeterReadings sample orchestration in the Orchestrator Client, which if successful, updates the Fuel Meter and Hour Meter with inputs sent through the sample orchestration.

Caution: You must perform this test in an environment with EnterpriseOne pristine data because sample orchestrations are designed to work with pristine data only.

To test your EnterpriseOne Orchestrator implementation:

1. In the Orchestrator Studio, click the **Tools** link and then click the **Orchestrator Client** icon.

You can also access the Orchestrator Client directly using this URL:

`http://<ais_server>:<port>/jderest/client`

2. On the Orchestrator Client Sign In screen, enter your EnterpriseOne user credentials, environment, and role. It is highly recommended that you enter an EnterpriseOne environment used for testing, not a production environment.
3. Click the **Login** button.
4. In the Orchestrator Client, in the Orchestration Name drop-down, select JDE_ORCH_Sample_UpdateMeterReadings. The following name-value pair inputs appear in the Name and Value columns:

Name	Value
EquipmentNumber	34665
NewFuelMeterReading	11.2
NewHourMeterReading	110.15

5. Click the **Run** button to test the orchestration.

The Input area shows the input message in JSON format.

The Output area shows the result of the orchestration. If successful, it displays the resulting forms from each form service call in JSON format. If unsuccessful, it shows an error response in JSON format. A warning may appear even if the update is successful.

To verify that the orchestration invoked the transaction in EnterpriseOne:

1. Access the Meter Readings program (P12120)
2. Search for equipment number 34665. Make sure to select the Fuel Meter and Hour Meter check boxes.

If the results in the grid show the record with the new values as shown in the following image, then your EnterpriseOne Orchestrator implementation is successful.

Equipment Number	Asset Description	Fuel Meter Current Reading	Fuel Meter New Reading	Fuel Meter Net Increase	Fuel Meter Original Reading	Fuel Meter LifeTime Reading	Hour Meter Current Reading
34665 Forklift		11.20					110.15
34673 Commercial Clothes Washer							110.15

If the orchestration fails, verify that you properly followed the prerequisites and installation steps in this chapter. Also, make sure the test was performed in an environment with pristine data. In this environment, the equipment number 34665 that is referenced by the white list entry in P952000 should be in the database.

2.2.4 Set Up a Temporary Directory on the AIS Server for File Transfers

In the Orchestrator Studio, you can create a connector service request to transfer files in and out of EnterpriseOne. When the Orchestrator executes this type of service request, as part of the process, files are saved to a temporary directory on the AIS Server. You must use Server Manager to set up this directory. In Server Manager, access the basic configuration settings for the AIS Server, and use the Temporary Directory setting to establish this directory.

See Also:

- [Configuring a REST Connector to Transfer Files to a REST Service](#)
- [Configuring a Connector to Transfer Files Using File Transfer Protocol \(FTP\)](#)

2.2.5 Upgrade Orchestration Components

If you have legacy orchestration components on the AIS Server that were created in the Orchestrator Studio prior to release 3.0.1 or through manually configured XML files (pre-Orchestrator Studio), you must upgrade these components to user defined objects. See [Upgrading Orchestration Components to User Defined Objects \(Release 9.2.1\)](#) in this guide for more information.

3

Designing an Orchestration

This chapter contains the following topics:

- [Section 3.1, "Understanding the Orchestration Design Process"](#)
- [Section 3.2, "Identifying the Problem and Solution"](#)
- [Section 3.3, "Identifying the Data for the Orchestration"](#)
- [Section 3.4, "Identifying the Rules for the Orchestration"](#)
- [Section 3.5, "Identifying the Cross Reference and White List Information for the Orchestration"](#)
- [Section 3.6, "Identifying the Service Request Information for the Orchestration"](#)

3.1 Understanding the Orchestration Design Process

You might already have a business process in EnterpriseOne that involves manually entering data into EnterpriseOne from a device that collects data. Or you might use a non-EnterpriseOne system to record data from various devices. Or you might not yet understand how data from these devices can be used by JD Edwards EnterpriseOne applications. Because the Orchestrator can accept data from any source that can send a REST call, it does not matter if the data is coming from an IoT device, an external system, or even from another EnterpriseOne application. This chapter uses an IoT device as an example, but regardless of where the input is coming from, your design process is the same.

Before you can create an orchestration, you need to perform an analysis to:

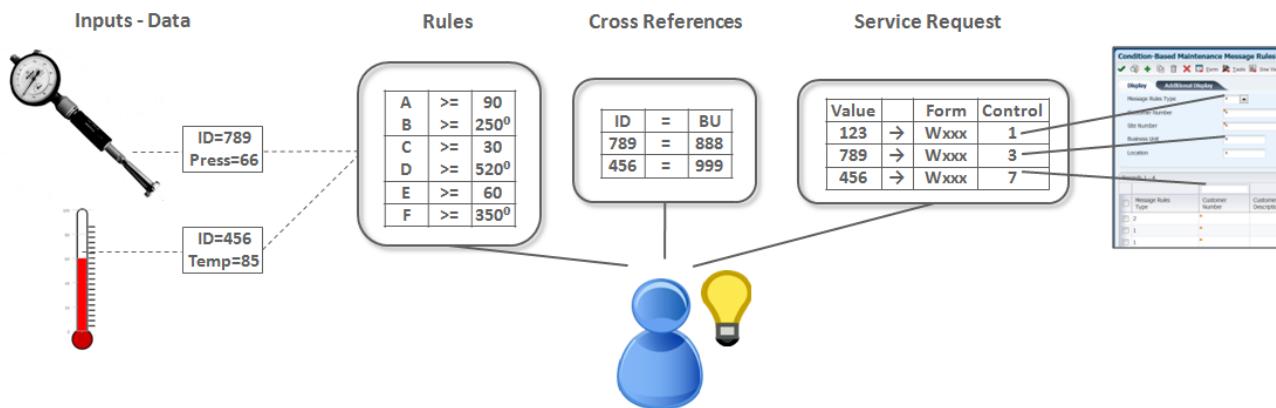
- Identify the problem and the solution.
- Identify the data that you want to collect.
- Define the rules and conditions that determine how to process the data.
- Identify the EnterpriseOne application inputs (fields and grid columns).
- Identify additional applications in which to work with the data, such as a custom Java application to perform a specific business process or a process for storing the data in another database.

You can use a simple worksheet for your analysis or you could use a storyboard, flow chart, or a combination of methods depending on the complexity of your orchestration. Use the information captured from your analysis to configure orchestrations in the Orchestrator Studio as described in [Chapter 4, "Creating Orchestrations with Orchestrator Studio 7.x.x.x"](#).

Example: Company A's Orchestration Design Process

Company A used a storyboard as part of their orchestration design process to illustrate the design of a simple orchestration. [Figure 3–1](#) shows an illustration of the overall result of Company A's design process. The remaining sections in this chapter contain additional details about each part of the design process and examples of how Company A identified the information required for the orchestration.

Figure 3–1 Example of an Orchestration Design Process



3.2 Identifying the Problem and Solution

Begin the analysis by identifying the problem or the data gap, and then identify how you want to use the data in EnterpriseOne, or in other words, determine which EnterpriseOne business process or transaction you want to invoke.

Example: Company A's Problem and Solution

Problem

Company A currently uses sensors to detect issues in certain assets to prevent potential breakdowns. Specifically, the company uses vibration and temperature sensors on various pieces of equipment. The data read from the sensors is not tied into their EnterpriseOne system; instead, it is integrated with a third-party software program that an operations manager has to access several times a day to monitor equipment. It would be ideal if the company could eliminate the need to use a disparate software program to manually oversee the performance of its equipment.

Solution

Company A wants to design a process that uses the EnterpriseOne Condition-Based Maintenance program for monitoring. With the Orchestrator Studio, Company A can create an orchestration with rules and conditions that:

- Invoke a transaction in EnterpriseOne Condition-Based Maintenance that sends a *warning* message if vibration or temperature levels are within a certain range above normal operating levels.
- Invoke a transaction in EnterpriseOne Condition-Based Maintenance that sends an *alarm* message if vibration or temperature levels exceed a certain level.

3.3 Identifying the Data for the Orchestration

After you identify the problem and determine the EnterpriseOne applications or processes that you want to invoke, you need to identify the device data or payload to use in the orchestration.

For example, a reading from a sensor might include a vibration measurement, a temperature measurement, and the date and time of the readings.

Example: Company A's Data Analysis

The "Input - Data" area in [Figure 3–1](#) highlights the data that Company A identified for their orchestration.

3.4 Identifying the Rules for the Orchestration

Next, identify the rules and conditions to determine how to process data from the sensors.

You will use the information from this part of the analysis to create a rule for the orchestration as described in the [Creating Rules](#) section in this guide.

Example: Company A's Rules Analysis

In this part of the analysis, Company A identified the following conditions:

- A vibration reading greater than or equal to 90 and a temperature greater than or equal to 250 will trigger an alarm message.
- A vibration reading greater than or equal to 30 and a temperature greater than or equal to 520 will trigger an alarm message.
- A vibration reading greater than or equal to 60 will trigger a warning message.
- A temperature reading greater than or equal to 350 will trigger a warning message.

The "Rules" area in [Figure 3–1](#) shows the rules and conditions that Company A is using to determine which data should be processed by the orchestration.

3.5 Identifying the Cross Reference and White List Information for the Orchestration

Identifying cross references involves identifying and mapping each piece of data to a value or data item in an EnterpriseOne form field or grid column.

Use the information from this part of the analysis to configure cross references for an orchestration as described in the [Creating Cross References](#) section in this guide.

Also, in this phase you can decide if you want to incorporate a white list. A white list provides an additional security layer for the orchestration by allowing data from only the devices defined in the white list. See the [Creating White Lists](#) section on how to set up a white list.

Example: Company A's Cross Reference Analysis

Company A used this phase of the analysis to map the following inputs to EnterpriseOne fields:

Input	EnterpriseOne Field
sensor ID	equipment number measurement location
equipment number	warning recipient alarm recipient

The "Cross Reference" area in [Figure 3–1](#) highlights the cross references Company A is using in the orchestration.

3.6 Identifying the Service Request Information for the Orchestration

Next, you need to identify how the data is used to invoke a business process or transaction in EnterpriseOne. This involves identifying the EnterpriseOne applications and inputs including the control IDs for EnterpriseOne buttons, fields, and so forth.

You can also determine if you want to use custom Java to execute a custom process or to route data into another database.

Use the information gathered from this phase of the analysis to design the service request for the orchestration as described in the [Creating Service Requests](#) section in this guide.

Example: Company A's Service Request Analysis

The "Service Request" area in [Figure 3–1](#) highlights the data Company A is using for the service request.

4

Creating Orchestrations with Orchestrator Studio 7.x.x.x

Important: This chapter has been updated in support of Orchestrator Studio 7.0.0.0 and 7.1.0.0. The features related to these releases are noted with the release number.

Orchestrator Studio 7.1.x.x is the latest version which requires a minimum of EnterpriseOne Tools 9.2.3.2. If you have not installed Orchestrator Studio 7.1.x.x, see [Chapter 2, "Implementing the Orchestrator Studio"](#) in this guide.

Instructions for using prior versions of the Orchestrator Studio are in the appendices of this guide.

This chapter describes how to take your orchestration design from analysis to implementation. It contains the following topics:

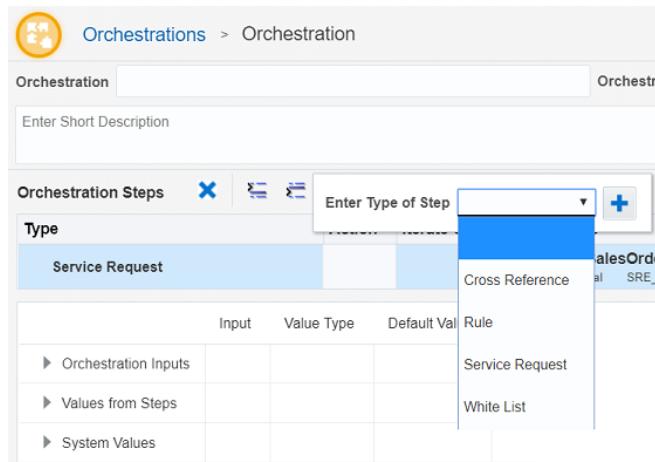
- [Section 4.1, "Understanding the Orchestrator Studio and Orchestrations"](#)
- [Section 4.2, "Accessing the Orchestrator Studio"](#)
- [Section 4.3, "Navigating the Orchestrator Studio"](#)
- [Section 4.4, "Creating Service Requests"](#)
- [Section 4.5, "Creating Rules"](#)
- [Section 4.6, "Creating Cross References"](#)
- [Section 4.7, "Creating White Lists"](#)
- [Section 4.8, "Creating Orchestrations"](#)
- [Section 4.9, "Creating Schedules for Orchestrations"](#)
- [Section 4.10, "Updating Version 1 Orchestrations to Version 2 Orchestrations"](#)
- [Section 4.11, "Reloading Orchestrations and Orchestration Components"](#)
- [Section 4.12, "Supported Input Message Formats"](#)
- [Section 4.13, "Orchestration Security Considerations"](#)
- [Section 4.14, "Exporting Orchestration Components from the Orchestrator Studio"](#)
- [Section 4.15, "Importing Orchestration Files in the Orchestrator Studio"](#)

4.1 Understanding the Orchestrator Studio and Orchestration

The Orchestrator Studio is a web-based application for creating orchestrations and the components that comprise an orchestration. The JD Edwards EnterpriseOne Orchestrator processes these components when executing an orchestration instance on the AIS Server.

Use the Orchestrator Studio to create the following components:

- ❑ **Orchestrations.** An orchestration is the master component that provides a unique name for an orchestration process. The orchestration is where you define the inputs for the orchestration, the expected incoming data. It also includes orchestration steps, which are invocations to the other components described in this list. When the Orchestrator invokes an orchestration, it processes the steps defined in the orchestration.
 - ❑ **Service Requests.** A service request contains the instructions that an orchestration uses to:
 - Perform a business transaction or query data in EnterpriseOne.
 - Send a message about a transaction to EnterpriseOne users or external users.
 - Retrieve Watchlist data from EnterpriseOne.
 - Invoke an EnterpriseOne report.
 - Transfer files between EnterpriseOne and another system.
 - Execute a custom process using custom Java or Groovy.
 - Invoke a REST service, database, a notification, or another orchestration.
 - Import data from a CSV file (Orchestrator Studio 7.0.0.0).
 - ❑ **Rules.** A set of conditions that the input to the orchestration is evaluated against to produce a true or false state. With rules, a false outcome or true outcome can invoke further orchestration steps. You can also nest rules, in which an outcome of one rule can invoke a different rule, to produce complex evaluations. You can also use custom Java to define rules.
 - ❑ **Cross References.** A set of data relationships that map third-party values to EnterpriseOne values. For example, a device's serial number can be cross-referenced to a JD Edwards EnterpriseOne Equipment Number for use in service requests.
 - ❑ **White Lists.** A white list contains an inclusive list of values permitted in the orchestration and terminates the orchestration process if the data is not recognized.
 - ❑ **Schedules.** A schedule defines how often the system executes an orchestration or notification. You can define a schedule using minutes, hours, days, or a Cron string (for example, every Tuesday at 2:00 pm). You can attach the same schedule to multiple components.
- You can also create notifications in the Orchestrator Studio. A notification is a process that can run independent of an orchestration. It enables the system to notify users of business events as they happen. The notification can contain boilerplate text and a shortcut to an EnterpriseOne application and can be configured to execute a Watchlist or an orchestration. To learn how to create notifications, see the *JD Edwards EnterpriseOne Tools Notifications Guide*.
- Figure 4-1 shows the drop-down list of the steps you can add to an orchestration. Each step in an orchestration is simply a reference to a cross reference, rule, service request, or white list component.

Figure 4-1 Orchestration Steps in the Orchestrator Studio

4.1.1 Invoking an Orchestration

When a new orchestration is saved in the Orchestrator Studio, the name of orchestration is used to define an endpoint on the AIS Server. The endpoint URL is:

```
http://<server>:<port>/jderest/orchestrator/<orchestrationname>
```

To invoke an orchestration, calling applications or devices use a post operation to this URL, where <orchestrationname> is the name of your orchestration. The post operation must include security parameters that enable access to the orchestration and any EnterpriseOne application invoked by the orchestration. See [Orchestration Security Considerations](#) for more information.

4.1.2 Reusable Orchestration Components

Orchestration components are reusable. You can include the same component, such as a service request or cross reference, in more than one orchestration. If a component is used as a step in more than one orchestration, you should evaluate how it is used in the other orchestrations before modifying it.

To determine if a component is used by other orchestrations, select the component and then click the "i" button to display a pop-up window that lists the orchestrations where the component is used.

When in doubt, use the "Save As" button to create a new component from an existing one. This enables you to give it a new name and modify it as necessary, and eliminates the risk of breaking other orchestrations where the component is used.

4.1.3 Orchestrations as User Defined Objects

All orchestration components created in the Orchestrator Studio are stored as user defined objects (UDOs) in EnterpriseOne. The Orchestrator Studio includes UDO features for creating, sharing, and modifying orchestration components as UDOs. See [User Defined Object \(UDO\) Features in the Orchestrator Studio](#) for a description of the UDO features.

4.2 Accessing the Orchestrator Studio

The Orchestrator Studio is a web application that runs in a web browser. Ask your system administrator for the URL to the Orchestrator Studio.

Important: Before users can access the Orchestrator Studio, an administrator must set up security to authorize access to the Orchestrator Studio design pages and determine the actions Orchestrator Studio users can perform. See [Chapter 10, "Administering the Orchestrator Studio and Orchestrations"](#) for more information.

To access the Orchestrator Studio:

1. In a web browser, enter the URL to the Orchestrator Studio:

`http://<adf_server>:<port>/OrchestratorStudio/faces/index.jsf`

2. On the Orchestrator Studio Sign In screen, enter your EnterpriseOne User credentials, environment, and role.

Note: It is highly recommended that you enter an EnterpriseOne environment used for testing, not a production environment.

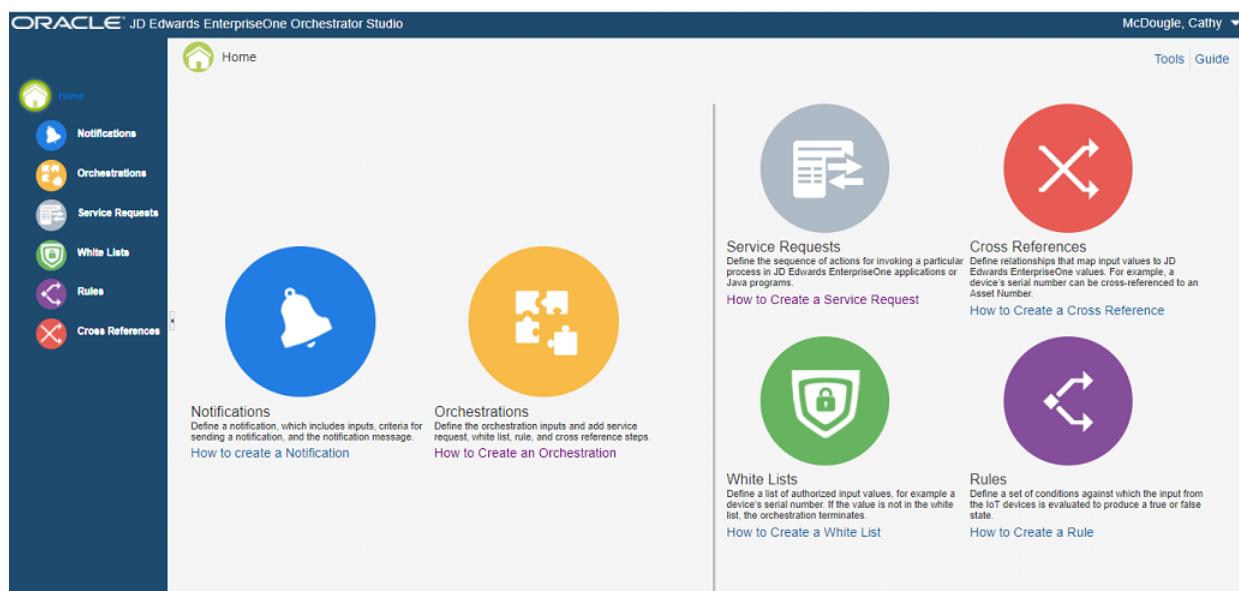
3. Click the **Login** button.

In the Orchestrator Studio, click the drop-down menu in the upper-right corner to view the path to the AIS Server. The drop-down menu also provides a link to log out of the Orchestrator Studio.

4.3 Navigating the Orchestrator Studio

The orchestration component icons on the Orchestrator Studio Home page take you to the design pages for creating and modifying each orchestration component. You can click the **Home** icon at the top left of the Home page to display a side panel, which provides another way to access the orchestration component design pages. You can also access this side panel within the component design pages for easy navigation between the different design pages. [Figure 4–2](#) shows the Home page with the side panel enabled.

Figure 4–2 *Orchestrator Studio Home*



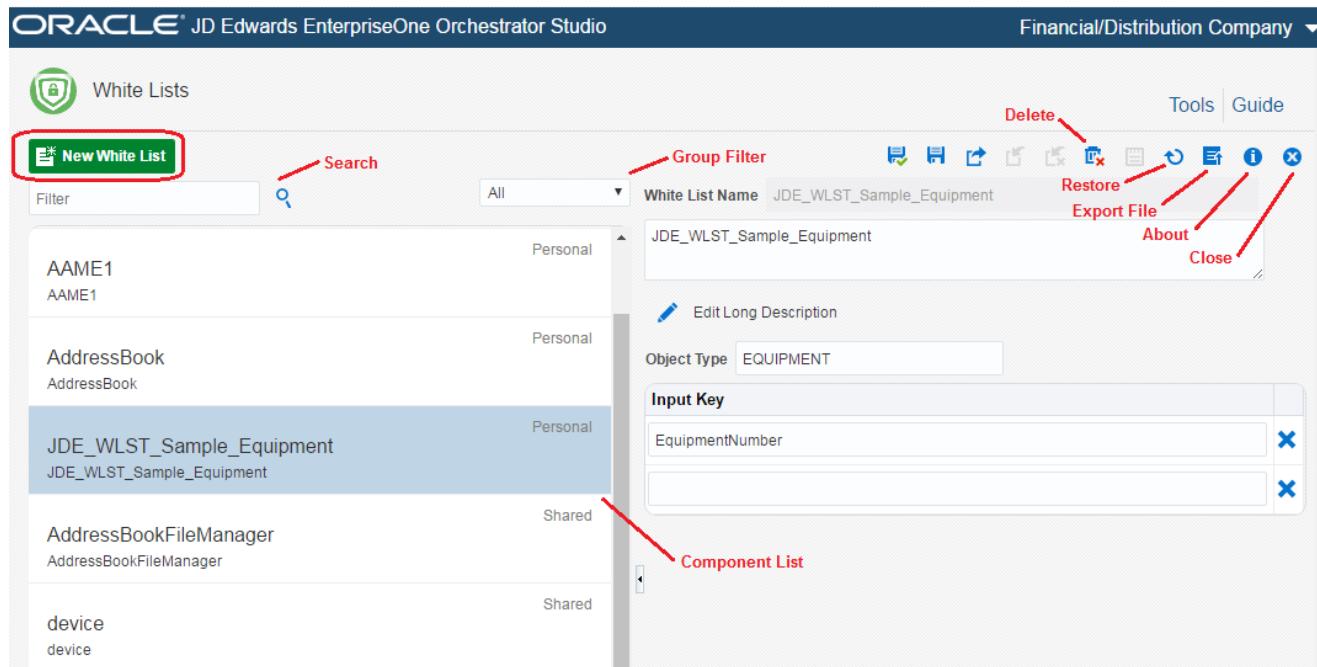
The Tools link in the upper-right corner of the Home page provides access to the Orchestrator Studio Tools page. This page provides links to tools for testing orchestrations, importing orchestration files, creating connection soft coding records for connector service requests, creating schedules, and accessing the JD Edwards EnterpriseOne web client. For more information, see the following topics:

- [Testing Orchestrations in the EnterpriseOne Orchestrator Client](#)
- [Importing Orchestration Files in the Orchestrator Studio](#)
- [Creating Connection Soft Coding Records for Connector Service Requests](#)
- [Creating a Schedule](#)

4.3.1 Orchestrator Studio Design Page Features

All Orchestrator Studio design pages contain the following features, which are highlighted in [Figure 4–3](#):

- **Component list.** Displays a list of existing components.
Use the vertical divider next to the component list to adjust the size of the list. You can click the raised tab on the divider to hide or show the component list.
- **Group Filter drop-down list.** Enables you to display components in the component list by UDO status: Personal, Pending Approval, Rework, Reserved, Shared, or All.
- **Search field.** Search for an existing component in the list.
- **New <Component> button.** Create a new component.
- **i (About).** Takes you to the About page which provides the Status, Detail, Description, and Object Name of the selected component. It also shows a list of the orchestrations where the component is used.
- **Restore All or Restore <Component>.** Restore the component to its original state if you made a mistake and do not want to save your changes.
- **Export File.** Export the component file to your local machine, which you should use only to inspect the XML of the component. See [Exporting Orchestration Components from the Orchestrator Studio](#) in this guide for more information.
- **Close.** Exit the design page.

Figure 4–3 Orchestrator Studio Design Page Features

4.3.2 User Defined Object (UDO) Features in the Orchestrator Studio

Orchestration components are saved and managed as UDOs in EnterpriseOne. The Orchestrator Studio includes UDO buttons, highlighted in [Figure 4–4](#), that enable you to create orchestration components for your own personal use, publish or "share" orchestration components, and modify shared orchestration components created by other users.

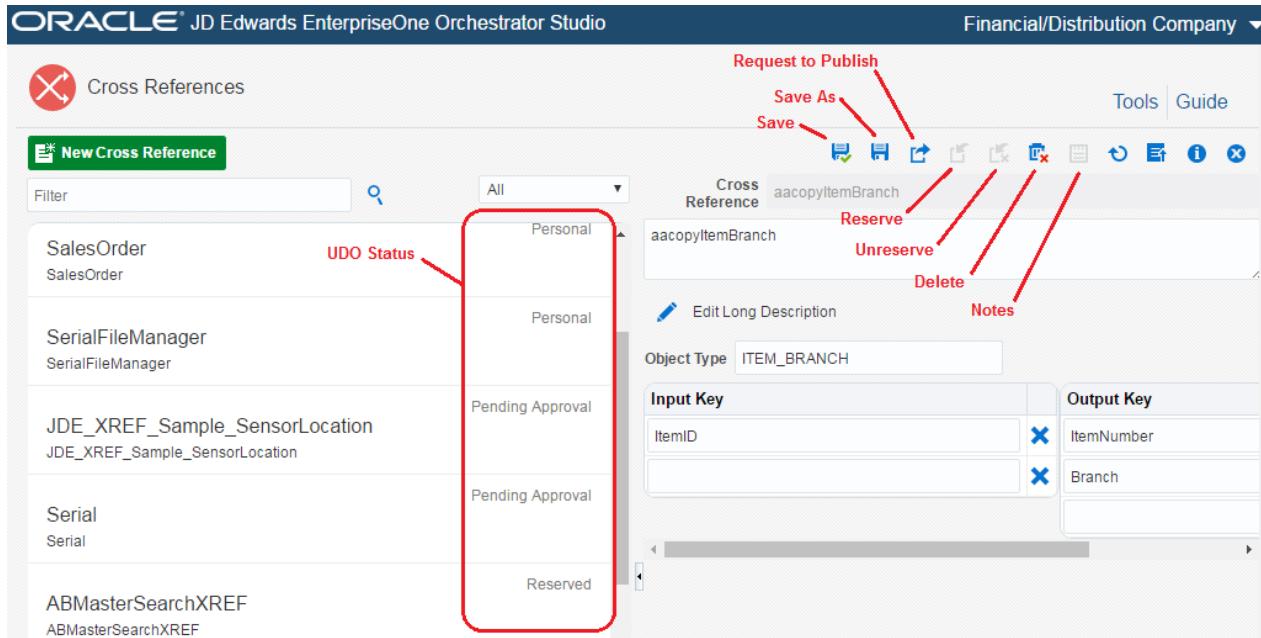
Note: The actions that you are allowed to perform in the Orchestrator Studio depend on the UDO security permissions granted to you by a system administrator. See [Setting Up UDO Security for Orchestrator Studio Users \(Release 9.2.1\)](#) in this guide for more information.

Orchestration components as UDOs enables administrators to use EnterpriseOne administration tools to manage the life cycle of orchestration components. For more information about the life cycle management of UDOs, see "UDO Life Cycle and Statuses" in the *JD Edwards EnterpriseOne Tools Using and Approving User Defined Objects Guide*.

[Table 4–1](#) describes the UDO buttons in the Orchestrator Studio design pages and the life cycle status enacted by each UDO action.

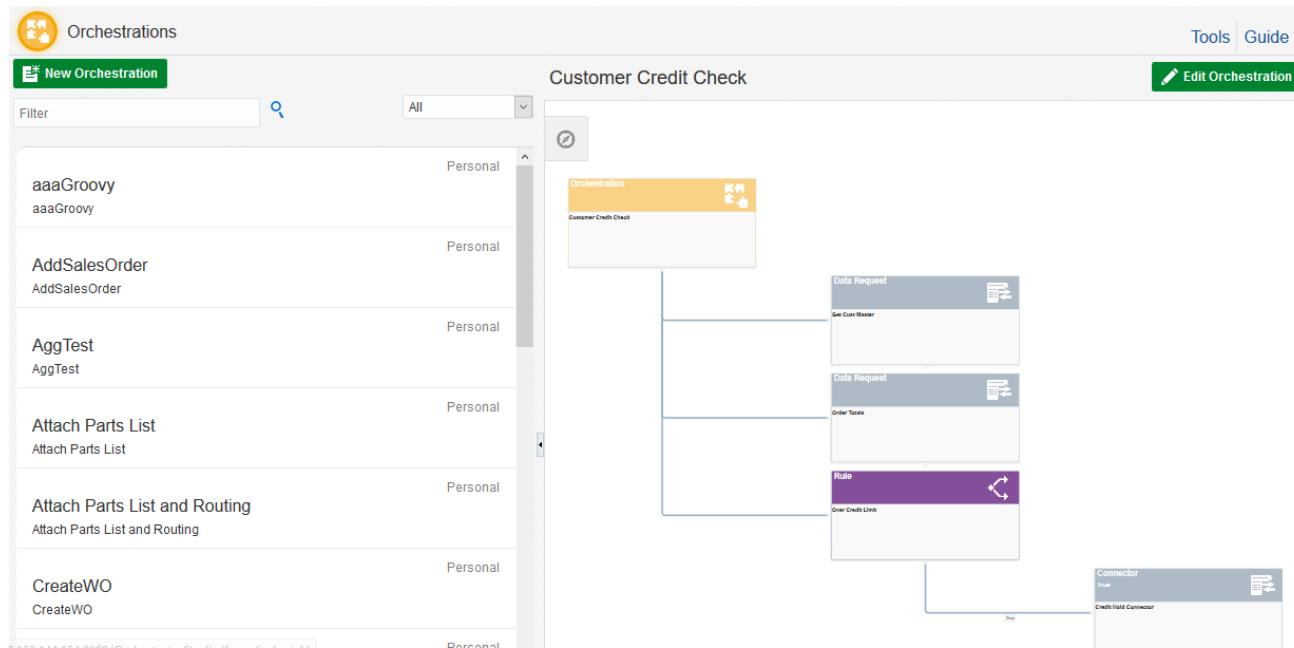
Table 4–1 Description of UDO Features in Orchestrator Studio Design Pages

UDO Button	Description
Save and Save As	Saves the orchestration component to a status of "Personal." Components with a status of "Personal" are components that you are developing and have not been shared for publishing to the AIS Server.
Request to Publish	Sends the orchestration component for approval for sharing. An administrator or approver must approve it in order for the UDO to be shared. The component status changes to "Pending Approval" in the component list and then changes to "Shared" once it is approved. If rejected, that status changes to "Rework." At that point, you can edit the component and then use the Request to Publish button to send it for approval again.
Reserve	Reserves a shared UDO so you can modify it. When reserved, no other users can make changes to it. The component status changes to "Reserved."
Unreserve	Cancels the reserved component, which returns the status of the component to "Shared."
Delete	Deletes a "Personal" UDO. You cannot use this button to delete a shared UDO. Shared UDOs can only be deleted by an administrator.
Notes	Available when the component is in the "Pending Approval" status, this button enables you to add an additional note to send to the approver of the UDO. The Notes button is active only if there was a note added the first time the UDO was sent for approval using the "Request to Publish" button. This feature enables you to add an addendum to the original note.

Figure 4–4 Design Page UDO Features

4.3.3 Working with the Graphical Representation of an Orchestration

Figure 4–5 shows an example of the initial Orchestrations page, which lists all orchestrations that you have access to and provides a graphical representation of an orchestration with all its components.

Figure 4–5 Orchestrations Page

The graphic area includes the following features:

- ☒ **Control Panel**

The Control Panel icon in the upper-left corner of the graphic area contains directional controls to pan left, right, up, and down, as well as zoom in or zoom out. "Zoom to Fit" displays the entire graphical representation in the window. The layout buttons change the layout to vertical, horizontal, tree, radial, or circle, which helps to view more complex orchestrations that contain multiple components.

- ☒ **Informational hover help**

Hover your mouse over a component in the graphical area to view an enlarged image of the component. Hovering over the labels on the lines between a rule component and its child components magnify the "True" or "False" label. A "True" label indicates the child component will be invoked if the conditions in the rule are met. A "False" label indicates the child component will be invoked when the condition of the rule is not met.

- ☒ **Isolate and Restore buttons**

The Isolate button on the left side of a component shows only that component in the graphic area. Restore displays all orchestration components.

- ☒ **Access to the design page for editing the component**

When you click a box representing a component, the Orchestrator Studio takes you to the design page for modifying that particular component.

4.4 Creating Service Requests

This section contains the following topics:

- ☒ [Section 4.4.1, "Understanding Service Requests"](#)
- ☒ [Section 4.4.2, "Creating a Service Request"](#)
- ☒ [Section 4.4.3, "Configuring a Form Request in the Orchestrator Studio"](#)

- Section 4.4.4, "Creating a Form Request with the JD Edwards EnterpriseOne Orchestrator Process Recorder (Release 9.2.2.4)"
- Section 4.4.5, "Configuring a Data Request"
- Section 4.4.6, "Configuring a Message Request"
- Section 4.4.7, "Configuring a Connector Service Request"
- Section 4.4.8, "Configuring a Watchlist Service Request"
- Section 4.4.9, "Configuring a Report Service Request"
- Section 4.4.10, "Configuring Form Request and Data Request Processing"
- Section 4.4.11, "Configuring a Custom Service Request with Java (Advanced)"
- Section 4.4.12, "Configuring a Custom Service Request with Groovy (Advanced)"

4.4.1 Understanding Service Requests

A service request provides the instructions that enable an orchestration to carry out a particular task. You can create the following types of service requests in the Orchestrator Studio:

- **Form Request**

A form request contains the instructions for the orchestration to perform a particular business transaction in EnterpriseOne.
- **Data Request**

Use a data request in an orchestration to query and return values from an EnterpriseOne table or business view. You can also configure a data request to perform an aggregation on data to return aggregate amounts.
- **Message**

Use a message request if you want an orchestration to send a message to an external email address (requires an SMTP server) or EnterpriseOne users through the EnterpriseOne Work Center.
- **Connector**

Use a connector request to invoke an orchestration, notification, REST service or database. For a connector to an orchestration, a connector can invoke a local orchestration or an orchestration on another AIS Server, such as an AIS Server on another EnterpriseOne system in a multisite operation.

Also, you can configure a connector to use FTP or a REST call to transfer report output or other files to an external server.
- **Custom**

Use custom Java or Groovy to execute a custom process.
- **Watchlist**

Use a Watchlist service request to use the information from Watchlists, such as critical or warning states, threshold levels, and number of records, within an orchestration.
- **Report**

Use a report request to invoke a batch version of a report in EnterpriseOne.

Before you create a service request, you must first identify the EnterpriseOne task or business process that you want the service request to perform. See [Identifying the Service Request Information for the Orchestration](#) for more information.

4.4.2 Creating a Service Request

To create a service request:

1. On the Orchestrator Home page, click the **Service Requests** icon.
The Orchestrator Studio displays the initial Service Requests page.
2. On this page, click **Create Service Request**, and from the drop-down list, select the type of service request that you want to create:
 - Form Request

Recommended:

Instead of manually creating a form request in the Orchestrator Studio, use the JD Edwards EnterpriseOne Orchestrator Process Recorder to create it. You can access the Process Recorder in EnterpriseOne and record each step or action that you want the form request to perform, which the Process Recorder then saves as a form request. See [Creating a Form Request with the JD Edwards EnterpriseOne Orchestrator Process Recorder \(Release 9.2.2.4\)](#).

- Custom
- Data Request
- Message
- Connector
- Watchlist
- Report

Alternatively, access the Orchestrations design page and add a Service Request step to an orchestration. At the end of the service request row, click **Edit** (pencil icon) and select the service request type that you want to create.

3. On the Service Requests design page, click the **Product Code** drop-down list to select a product code to associate with the service request.
This gives an administrator the option to manage UDO security for orchestration components by product code.
4. On the *Service Request Type* design page, complete these fields:
 - Service Request.** Enter a name for the service request. Do **NOT** include special characters in the name. You should include the service request type, such as "form request" or "data request," in the name to distinguish it from other service requests listed in the Service Request design page.
 - Short description field.** In the space provided, enter a description with a maximum of 200 characters. This description will appear below the service request name in the component list.
5. (Optional) Click **Long Description** to provide additional details about the purpose of the service request.
6. Click **Save**.

The first time a new service request is saved, it is saved as a "Personal" UDO. After configuring the service request, you can use the UDO buttons described in the [User Defined Object \(UDO\) Features in the Orchestrator Studio](#) section to move the service request to the appropriate status.

7. Refer to the appropriate section for instructions on how to configure the service request type: form request, data request, message request, connector request, Watchlist request, report request, custom Java request, or custom Groovy request.

4.4.3 Configuring a Form Request in the Orchestrator Studio

A form request contains the instructions for the orchestration to perform a particular business process or transaction in EnterpriseOne.

Create the form request as described in [Creating a Service Request](#), and then perform these tasks:

- [Loading EnterpriseOne Application Form Fields and Controls](#)
- [Configuring Form Request Actions](#)
- [Configuring the Order of Execution](#)
- [Populating Multiple Grids with Repeating Inputs \(Optional\)](#)

> Tutorial:

[Click here to view a recording of this feature.](#)

Recommended:

Instead of using the Orchestrator Studio to create a form request, use the JD Edwards EnterpriseOne Orchestrator Process Recorder. You can access the Process Recorder in EnterpriseOne and record each step or action that you want the form request to perform, which the Process Recorder then saves as a form request. See [Creating a Form Request with the JD Edwards EnterpriseOne Orchestrator Process Recorder \(Release 9.2.2.4\)](#).

4.4.3.1 Loading EnterpriseOne Application Form Fields and Controls

1. In the Available Actions area, complete the following fields:
 - **Application.** Enter the application ID.
 - **Form.** Click the drop-down list to select the form.
 - **Version.** Enter the version ID.

If you leave Version blank, the Orchestrator will use the default version when it executes the form request.

Leaving the version field blank with the Application Stack option selected gives you the option to pass in the version from an orchestration input. This enables you to configure an orchestration that includes rules with conditions so that clients can call different versions dynamically.

When you add the form request to an orchestration, click the **Add Inputs to Orchestration** button to add the variable representing the version to the Orchestration Inputs area. The variable is displayed with the application ID followed by _Version, for example P03013_Version.

2. Click the **Form Mode** drop-down list and select the appropriate form mode: **Add**, **Update**, or **Inquiry**.

At runtime, the controls that appear on an EnterpriseOne form are dependent on the form mode as specified in Form Design Aid (FDA). The form mode ensures that you can see all of the form controls in the Orchestrator Studio that are available in the form at runtime.

3. Select the following options as appropriate:

- Application Stack. Application stack processing enables the form request to establish a session for a specific application and maintain the session across calls. With application stack processing, the form request can invoke multiple forms in different applications to complete a business process. Without the application stack processing, each form in the form request is opened independently.

If you use the Application Stack option, you must configure the form request with the actions to navigate between forms. Without this navigation, additional forms will not be called. There are other considerations when selecting values to return from forms in an Application Stack, as described in [Configuring Form Request Actions](#).

- Run Synchronously. Enabled by default, this ensures that the form request completes all actions before the Orchestrator executes the next step in the orchestration. It also ensures that the events in the EnterpriseOne application are run synchronously. This option is recommended if there are steps that follow the form request in the orchestration or if you configure the form request to return a response.

Caution: If the form request involves invoking a control that launches an asynchronous report or any long running process, it is possible that the Orchestrator could time out if Run Synchronously is enabled.

- Bypass Form Processing. This option enables the form request to ignore event rules when loading the form fields and controls in the Available Actions grid. For example, some forms have event rules to perform a find on entry or hide certain fields, which can cause the load to fail or prevent certain fields from loading in the Available Actions grid.

4. Click the Load Form button.

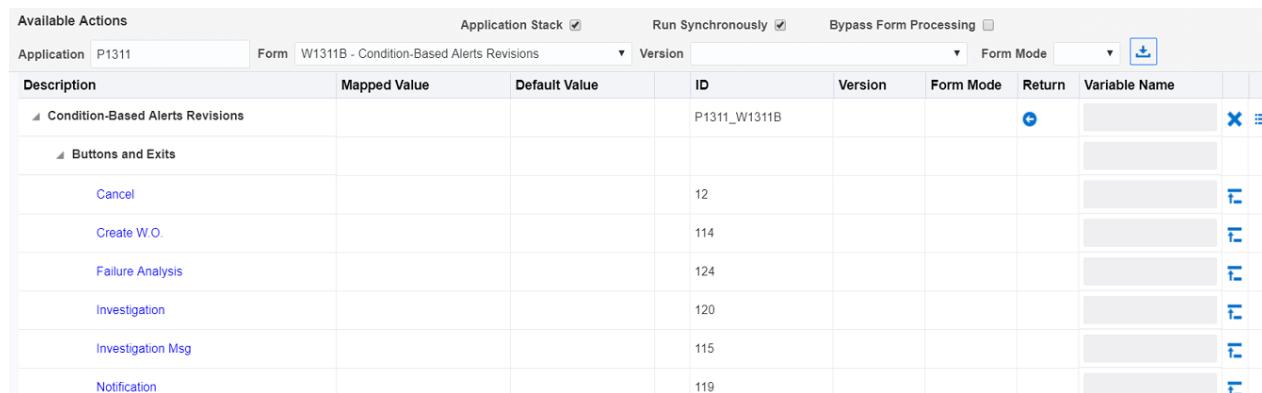
The Orchestrator Studio loads the controls and fields in the grid. The name of the form is displayed in the first row in the grid.

If a form request needs to invoke more than one application to complete the desired task, you can load controls and fields for additional application forms as needed.

4.4.3.2 Configuring Form Request Actions

Figure 4–6 shows the Available Actions area, where you specify the EnterpriseOne fields and controls used to perform the desired business transaction in EnterpriseOne.

Figure 4–6 Available Actions Area in the Form Request Design Page



Available Actions		Application Stack <input checked="" type="checkbox"/>	Run Synchronously <input checked="" type="checkbox"/>	Bypass Form Processing <input type="checkbox"/>				
Description	Mapped Value	Default Value	ID	Version	Form Mode	Return	Variable Name	
Condition-Based Alerts Revisions			P1311_W1311B					
Buttons and Exits								
Cancel			12					
Create W.O.			114					
Failure Analysis			124					
Investigation			120					
Investigation Msg			115					
Notification			119					

Use the following features to configure the actions in the form request. After configuring each action, click the **Add Action** button at the end of each row to add it to the Order of Execution area.

□ **Description** (informational only)

This column displays the controls and fields for each form in a collapsible/expandable parent node named after the EnterpriseOne form. Child nodes categorize other items and include a Buttons and Exits node, a node for displaying the available Query by Example (QBE) fields in a grid (if applicable), and a node for displaying all available columns in a grid.

□ **Mapped Value**

This is an editable column for identifying the inputs to the form request. In each field to which you want to map an orchestration input, enter a variable name. When you add this form request to an orchestration, you map the orchestration input to this variable.

The only fields to which you can map inputs are EnterpriseOne editable fields. The variables names for the inputs in the form request should match the inputs defined in the orchestration to which you add the form request. If they do not match, you can use the "Transformations" feature to map input names after you add the form request to an orchestration. See [Adding Inputs to an Orchestration](#) and [Mapping Orchestration Inputs](#) for more information.

Instead of a mapped value, you can enter a hard-coded value in the Default Value column or you can click the "Text substitution" check box to combine inputs into a text string.

You can also use this field to populate multiple rows in multiple grids. See [Populating Multiple Grids with Repeating Inputs \(Optional\)](#) for more information.

□ **Default Value**

- For an editable field to which you want to map an input, use this column to enter a hard-coded value. You can also add a hard-coded value to use if the mapped value returns a null.
- For a check box or a radio button, you can select it to include it. If there are multiple radio buttons grouped together on a form, select only one. If you select more than one, only the last radio button in the Order of Execution list will be used.
- For a combo box control (a list of items in a drop-down menu), the Studio displays a drop-down menu in the Default Value column in which you can select the appropriate item from the list.

□ **Select All Rows** (row)

Add this action to select all rows in a grid. Use this action if the form has a button to perform an action on all selected grid rows. For example, if you want to update the PO status for all rows in a grid, add this action to the Order of Execution area and then configure the next action to update the PO status.

□ **Select First Row** (row) or **Select Row**

Add this action if the form requires selecting the first row in a grid to perform a particular action.

You can enter a variable in the Mapped Value column of this row, which enables you to map an orchestration input to this variable. Or you can enter a value in the Default Value column to select a specific row. To select the first row in the grid, leave these columns blank when adding this action to the Order of Execution.

□ **Row Number for Update** (row)

If the task requires updating a particular row in an input capable grid, then map the appropriate input value containing the row number to the "Row Number for Update" row, or specify the row number in the Default Value column.

Do NOT use this action if the transaction involves adding rows to a grid.

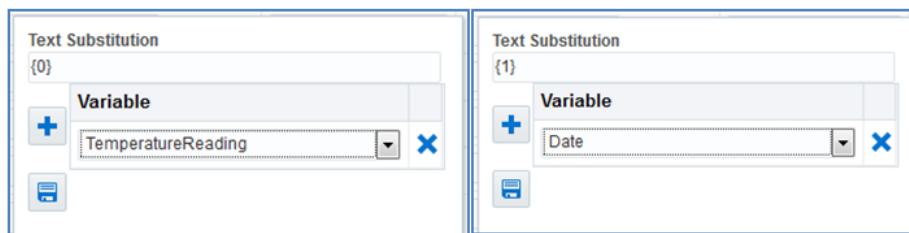
- ¤ **"Text Substitution"** check box column

Use text substitution to combine input values into a text string for an input into an EnterpriseOne field. For example, you can create a text string such as "Temp was {0} at {1}" that contains input values for temperature and time.

Note: If you use text substitution for the input, the field cannot contain a mapped value. The substituted text string becomes the default value for the field when you add it to the Order of Execution list.

1. Click the **Add Text Substitution** button (plus symbol) in the row that contains the field for which you want to substitute the input with text.
2. In the Text Substitution field in the pop-up box, use a variable for the first input by entering {0} (with brackets).
3. In the Variable field, enter the input variable name.
4. Click the **Add** (plus symbol) to add the variable for the text substitution.
5. Add the next variable, {1}, and then in the next blank row below the Variable field, enter the input variable name.
6. Click **Add** to add it to the text substitution.
7. Click **OK** to save the variables.

The following image shows an example of variables added for a temperature input value and a date input value:



- ¤ **ID** (informational only)

This column displays the ID of the control or field in EnterpriseOne.

- ¤ **Form Mode** (informational only)

If a form mode was selected, this column displays the form mode in the form row.

- ¤ **Return**

Select the check box in this column for fields that contain values that you want returned in the orchestration response. If you do not specify any return values, all values on the form will be returned. Return values may be used by customized third-party gateway programs or devices to perform further processing based on the values returned from EnterpriseOne applications.

Starting with Orchestrator Studio 7.0.0.0, when you select a value to return, a "Return Form Data" action is added to the order of execution. Make sure this action follows the

steps needed to populate the returned fields in the form, and make sure that it occurs before an action that clears the data or navigates from the page.

Only one Return action is allowed per form when configuring a form request in the Orchestrator Studio. If you need to return more than one value at different times on the same form, add the same form twice to the form request, configuring the additional return value in the second form. In this scenario, it is recommended to use the Process Recorder, where you can select multiple values to return from any form in the process.

Caution: If using the Application Stack option prior to EnterpriseOne Tools 9.2.3 and Orchestrator Studio 7.0.0.0:

- ☒ Only values from the last form in the application stack are returned; return controls specified for any form except the last form are ignored.
 - ☒ If you use a form to add or update a value that upon saving exits to another form, such as a Find/Browse form, the return values will come from the last form that appears after the save. In this scenario, you need to add the last form to the application stack and select the return controls from the last form.
-

☒ **Variable Name**

Enabled when the Return check box is selected, use this field to enter a variable name for the return value. When you add the form request to an orchestration, this name appears in the orchestration inputs grid, which makes the returned value available for mapping to subsequent steps in the orchestration.

You can also use this field to return a data set from an application grid. See [Retrieving and Passing Data Sets in an Orchestration](#) for more information.

☒ **Return Hidden Fields** (left arrow icon in the Return column)

Use this feature to return data from "hidden" fields or grid columns not displayed in the Available Actions area. Hidden fields are fields that appear only after performing a particular action in a form. In the EnterpriseOne web client, use the field-level help (Item Help or F1 key) to identify the control IDs for hidden fields and grid columns. The control ID is displayed in the Advanced Options section in the help pop-up window.

To return a hidden field:

1. In the Available Actions area, in the row of the form that contains the hidden fields, click the left arrow in the Return column.
A dialog box appears displaying the control IDs and associated variable names of any return fields already selected in the Available Actions area.
2. For hidden fields, enter the control ID of the hidden field in the Return Form Ids field. Optionally, you can enter a variable name to represent the return value in the associated Name field.
For multiple controls, use a bar delimiter to separate the values in both fields, making sure the order of the values in each field match.
3. For hidden grid columns, enter the control ID for the grid column in the Return Grid Ids field. Optionally, you can enter a variable name to represent the return value in the associated Name field.

You must enter a variable name for the return value if you want it to appear in the orchestration inputs list in the orchestration.

An example of the notation for multiple grid columns in a grid is: 1[32, 34, 42]

Where 1 represents the first grid in the form (because some forms can have multiple grids), and 32, 34, 42 each represent a column in the grid.

Use a bar delimiter to separate the variable names, making sure the order of the variable names matches the order of the grid control IDs, as shown in the following example:

Value	ID	Version	Form Mode	Return
Return Form Ids				
8 16 22				
Form Id Names				
User ID Address Phone				
Return Grid Ids				
1[14,13,11,12]				
Grid Id Names				
UDC Hard Coded UDC Special Handling Description 01 Description 02				
OK	Cancel			

- (Optional) "Options" icon (at the end of the first row in the grid)

This option contains settings that determine how the AIS Server processes the form request. See [Configuring Form Request and Data Request Processing](#) for details.

4.4.3.3 Configuring the Order of Execution

After actions are added to the Order of Execution grid, you can reorder them using the up and down arrow buttons to the right of each row. You can also delete any actions using the Delete (X) button. [Figure 4-7](#) shows the Order of Execution area.

You can enter or modify values in the Mapped Value and Default Value columns in the Order of Execution area.

Figure 4–7 Order of Execution in the Service Request Design Page

The screenshot shows the 'Service Requests > Form Request' interface. At the top, it displays 'Service Request AddCBM_Alarm' and 'Product Code 55'. Below this is a text input field containing 'AddCBM_Alarm'. To the right of the input field is a 'Long Description' button. The main area is titled 'Order of Execution' and contains a grid with the following data:

Description	Action	Mapped Value	Default Value			
Asset Number (edit)	SetControlValue	equipmentNumber				
Description	SetControlValue	description				
Alert Level	SetControlValue		2			
Event Date / Time	SetControlValue	date				
Event Time Formatted	SetControlValue	time				
Automated Response Type (edit)	SetControlValue		1			
OK	DoAction					

4.4.3.4 Populating Multiple Grids with Repeating Inputs (Optional)

You can configure a form request to map repeating inputs into one or more grids.

In the Available Actions area, rows for editable grids include a default value of GridData in the Mapped Value column. This value can be used to associate a set of repeating inputs to the grid. If you need to populate more than one grid in an orchestration, change this value to identify each grid. And then make sure that the "name" tag that identifies the repeating inputs in the detail inputs section of the orchestration input matches this value.

4.4.4 Creating a Form Request with the JD Edwards EnterpriseOne Orchestrator Process Recorder (Release 9.2.2.4)

The JD Edwards EnterpriseOne Orchestrator Process Recorder enables you to record a series of actions in EnterpriseOne and save those actions as a new form request. This simplified method to create a form request is an alternative to manually creating a form request in the Orchestrator Studio.

> **Tutorial:**

[Click here to view a recording of this feature.](#)

After launching and starting the Process Recorder in EnterpriseOne, you access an application and perform each of the steps to complete the business process or transaction that you want the form request to perform.

The Process Recorder automatically adds inputs to the form request for each field in which you enter data during the recording. The input is the name of the field, and the data you enter becomes the default value. For example, if you click a field called **Customer Number** and enter 1001, the Process Recorder records an input called Customer Number with a default value of 1001. You can delete, change, or replace the value with a variable by editing the form request in Orchestrator Studio.

When finished, on the last form, you can select any of the controls or grid columns on the form that contain values that you want the form request to return. Starting with EnterpriseOne Tools 9.2.3, you can select return values from any form while recording the form request.

After you save the form request in the Process Recorder, you can open the form request in the Orchestrator Studio, modify it as necessary, and add it to an orchestration. If you have an open Orchestrator Studio session, you will have to close it and sign in again to access the newly created form request.

4.4.4.1 Rules for Creating a Form Request in the Process Recorder

When using the Process Recorder to create a form request, follow these rules:

- ❑ Start the recording and then launch the first application from the carousel, Favorites, a link on an EnterpriseOne page, or Fast Path. Or you can open an application and then start the recording before performing the first action in a form. You cannot start recording after performing an action in an EnterpriseOne application.
- ❑ If the process involves adding multiple records, perform only the steps to add a single record. Later, when you add the form request to an orchestration, you can use the Iterate Over feature in the Orchestrator Studio to configure the form request to add multiple records.
- ❑ Some business processes entail using forms in more than one application. You can record this as long as you access the other forms from a Form or Row menu. However, you cannot return to the EnterpriseOne home page and launch another application. If the business process requires launching a second application from the EnterpriseOne home page, then record it as a separate form request.
- ❑ When recording a process in the Process Recorder, each transaction is saved in EnterpriseOne just as it would be if you were not using the Process Recorder. To ensure that you do not sully production data, it is highly recommended that you use the Process Recorder in an EnterpriseOne test environment. If recording in a production environment, delete any transactions that were created while using the Process Recorder.

4.4.4.2 Prerequisites

The Process Recorder is disabled by default. To make it available in EnterpriseOne, a system administrator must enable both of the following security types:

- ❑ UDO feature security.
A system administrator must enable the RECORDER feature in UDO feature security. See "Managing UDO Feature Security" in the *JD Edwards EnterpriseOne Tools Security Administration Guide*.
- ❑ UDO action security.
A form request is a type of service request, and service requests are saved and managed as a user defined objects (UDOs) in EnterpriseOne. Therefore, users must be authorized to create service requests through UDO action security before they can access and use the Process Recorder to create a form request. See "Managing UDO Action Security" in the *JD Edwards EnterpriseOne Tools Security Administration Guide*.

4.4.4.3 Using the Process Recorder to Create a Form Request

1. On the EnterpriseOne home page, click the user drop-down menu in the upper right corner and select **Record a Process**.
2. In the **Process Recorder** pop-up box, click **Start** to start the recording.

3. Access the first application from the carousel, Favorites, a link on an EnterpriseOne page, or Fast Path.

EnterpriseOne highlights the application title bar and application tab in the carousel (if enabled) of the application that is being recorded.

4. Perform each of the steps to complete the business process or transaction that you want the form request to perform.

When you enter a value in a field, the Process Recorder automatically records the field name as an input and the data that you enter as a default value. Later, you can change the name of the input and delete or change the default value by editing the form request in Orchestrator Studio.

Important: If you return to the EnterpriseOne home page or try launching a separate application from the Carousel while recording, the Process Recorder pauses the recording and will not record those steps. To resume the recording, click the **Paused** button and the Process Recorder will resume recording, returning you to the point in the process in which you initially paused the recording.

5. (Release 9.2.3) If you want the form request to return values from the form that you are on:
 - a. Click **Return Values**.
 - b. Click the controls and columns that contain values you want returned. If you add a control or column by mistake, click it again to remove it from the list.
 - c. Click **Resume** to continue recording the next steps in the process.

When you access the last form in the process and want the form request to return values from all controls and columns on the last form, leave the Return Controls and Return Columns field blank.

Note: If you are on EnterpriseOne Tools 9.2.2.4, you can return values only from the last form in the process.

6. After performing the final step, click **Stop**.

If you stopped the recording prematurely, click **Cancel**, and then continue to record the remaining actions. To discard the process and start over, click **Discard**.

7. To complete the recording, enter a name, product code, and description for the form request.
8. Click **Save**.

If you need to modify the form request, see [Configuring a Form Request in the Orchestrator Studio](#). To add it to an orchestration, see [Adding Steps to an Orchestration](#).

4.4.5 Configuring a Data Request

You can configure a data request to:

- Query and return data from an EnterpriseOne table or business view.
- Perform an aggregation of data from a table or business view and return aggregate amounts.

In a standard data request that returns data directly from a table or business view, you can use variables to define the return fields. In a data request that performs an aggregation, you can use variables to define the "group by" fields and the returned aggregate amounts.

When you add a data request to an orchestration, any variables used to define the returned data in the data request are automatically added to the orchestration as inputs. This enables you to map the returned data to subsequent steps in an orchestration.

Important: Variables are only supported in the first row of a response. If a query contains multiple rows, only the values from the first row will be available as variables.

4.4.5.1 Configuring a Data Request to Return Field Data

A data request includes filtering criteria and indicates the fields that contain data that you want returned. For example, a business analyst wants to add a data request to an orchestration that returns a customer's credit limit amount. The orchestration has "Customer Number" defined as an input. To configure the data request, the business analyst:

- ❑ Sets up filter criteria with a condition that filters on Address Number.
- ❑ Selects the Credit Limit and Alpha Name fields for the return data.
- ❑ Adds the data request to the orchestration, mapping the Customer Number input in the orchestration to the Address Number in the data request.

When the data request is added to the orchestration, the Credit Limit and Alpha Name fields automatically become additional inputs to the orchestration, which the business analyst can then map to subsequent orchestration steps.

> **Tutorial:**

[Click here to view a recording of this feature.](#)

To configure a data request to return field data:

1. Create a data request as described in [Creating a Service Request](#).
2. In the Available Actions area, enter the name of the table or business view in the Table/View Name field.

If you do not know the table or business view name, but you know the application that uses the table or business view, you can use the JD Edwards EnterpriseOne Cross Reference Facility (P980011) to identify the table or business view associated with the application. From the Cross Reference form, click the **Interactive Applications** tab, then click **Business Views Used by an Interactive Application** or **Tables Used by an Interactive Application**. If you do not have access to the Cross Reference Facility, ask an administrator to look up this information for you.

3. Click the **Load** button.

The Orchestrator Studio loads all fields from the table or business view into the grid.

Note: You can use the Data Set Variable Name field to configure the data request to return a data set. See [Retrieving and Passing Data Sets in an Orchestration](#) for more information.

4. Define the filtering criteria for the data request:

- a. In the Fields grid on the left, click the "**Filter**" icon next to the field or fields that contain data that you want to filter on.

The Orchestrator Studio displays each field in the Filter Criteria area on the right.

- b.** For each field in the Conditions box, select the appropriate operand and in the adjacent field, enter a hard coded value or a variable.

A variable appears in the field by default. If you modify the variable name, make sure the syntax includes the \$ sign and brackets, for example:

```
${Address Number 1}
```

Starting with Orchestrator Studio 7.1.0.0, if the field is a date, you can use the down arrow to set a special value, such as today plus or minus the number of days, months, or years that you specify.

- c.** Select the **Match All** or **Match Any** option to determine how the conditions are applied.

You can also click the **Options** button, and from the Query drop-down list, select a predefined query to use for the filtering criteria. You can use a query instead of or in combination with the filtering criteria defined in a data request. The queries that you can see in this list are based on UDO security permissions.

- 5.** Specify the data you want returned:

- a.** In the Fields grid, click the "Return" icon next to each field for which you want data returned.

Each field appears in the "Return Fields and Variable Names" on the right.

- b.** (Optional) You can use a variable for the return by entering a name for the variable in the adjacent blank field.

Note: Return variables do not need the \${ } notation. This notation is only necessary for input variables to distinguish between a variable and a hard coded value.

When you add a data request to an orchestration, these variables are automatically added to the orchestration as inputs, which you can use to map return data to subsequent orchestration steps.

- 6.** (Optional) For the return data, determine the order by which you want the data returned:

- a.** In the grid on the left, select the **Order By** icon next to any field that was selected for the return data.

The Orchestrator Studio adds the field name to the Order By area on the right.

- b.** In the drop-down list next to the field name, select **Ascending** or **Descending**.

- 7.** (Optional) Click the **Options** button to configure settings that control how the AIS Server processes a data request at runtime. See [Configuring Form Request and Data Request Processing](#).

4.4.5.2 Configuring a Data Request with Data Aggregation

> **Tutorial:**

[Click here to view a recording of this feature.](#)

To configure a data request to return aggregate amounts:

- 1.** Create a data request as described in [Creating a Service Request](#).

2. In the Available Actions area, enter the name of the table or business view in the Table/View Name field.

If you do not know the table or business view name, click the **Get View From Form** button and enter the application and form ID to load all fields from the primary business view associated with the form.

3. Click the **Load** button.

The Orchestrator Studio loads all fields from the table or business view into the grid.

4. Slide the **Aggregation** toggle to the right.

This enables the aggregation features in the Fields grid.

Note: You can use the Data Set Variable Name field to configure the data request to return a data set. See [Retrieving and Passing Data Sets in an Orchestration](#) for more information.

5. Click the "**Filter**" icon next to the fields that contain the data that you want to filter on.

The Orchestrator Studio displays each field in the Conditions area on the right.

6. Set up conditions for filtering data:

- a. For each field in the Conditions box, select the appropriate operand and in the adjacent field, enter a hard coded value or a variable.

A variable appears in the field by default. You can modify the variable name, but you must use the \$ sign and brackets in the syntax, for example:

`${Address Number 1}`

Starting with Orchestrator Studio 7.1.0.0, if the field is a date, you can use the down arrow to set a special value, such as today plus or minus the number of days, months, or years that you specify.

- b. Select the **Match All** or **Match Any** option to determine how the conditions are applied.

You can also select the Options button, and from the Query drop-down list, select a predefined query to use for the filtering criteria. You can use a query instead of or in combination with the filtering criteria defined in a data request. The queries that you can see in this list are based on UDO security permissions.

7. Click the "**Aggregate**" icon next to the fields that contain the data for the aggregation.

8. In the pop-up window, select the type of aggregate that you want to perform.

The aggregate options displayed depend on whether the field contains a numeric value or a string.

9. (Optional) In the Aggregations and Variable Names section, click the **Include Count** check box if you want the response to include a count of the records returned. To use the returned count in a subsequent orchestration step, enter a variable name in the adjacent field.

10. (Optional) Use the following features in the Fields grid to further define the data aggregation:

- **"Having" icon.** Click this icon next to a field for which you want to provide additional filtering criteria on an aggregation.

The Data Request design page displays the field in the Having section on the right.

- a. Click the drop-down list next to the field and select the operand.
- b. In the adjacent field, enter a hard coded value or variable.
- "Group By" icon. Click this icon next to any field that you want the aggregate results grouped by. In the "Group By and Variable Name" section on the right, you can enter a variable name.

Using "Group By" may result in multiple rows being returned, one for each unique combination of a group. For example, you might return total orders for an entire year for customers, and group the results by customer. In this case, the data request response will return a unique row for each customer with the customer's total orders for the year.

Note: Variables are only supported in the first row of a response. If a query contains multiple rows, only the values from the first row will be available as variables.

- "Order By" icon. For any fields used in an aggregate or "group by," click this icon to arrange return data in ascending or descending order.
11. (Optional) Click the **Options** button to configure settings that control how the AIS Server processes a data request at runtime. See [Configuring Form Request and Data Request Processing](#).

4.4.5.3 Viewing and Copying JSON Code of a Data Request

After configuring a data request, you can click the **Preview** button to view and copy the JSON code for a data request. Developers can use this code to develop AIS client applications that can make data request calls directly to the AIS Server rather than through an orchestration. See the *JD Edwards EnterpriseOne Application Interface Services Client Java API Developer's Guide* for more information.

4.4.6 Configuring a Message Request

Add a message request to an orchestration to send emails to external email systems or the EnterpriseOne Work Center email system. The following EnterpriseOne Work Center and workflow features are supported in a message request:

- Workflow distribution lists to send messages to a predefined group of EnterpriseOne users.
- Message templates from the data dictionary that contain boilerplate text and values related to a particular business process.
- Shortcuts to EnterpriseOne applications.

For more information about these workflow features, see the *JD Edwards EnterpriseOne Tools Workflow Tools Guide*.

In a message request, you can include variables to pass data from an orchestration input into a message request. You can include variables in the recipient fields, the subject and body, a message template, and a shortcut.

Starting with Orchestrator Studio 7.1.0.0, you can add multiple shortcuts.

> **Tutorial:**

[Click here to view a recording of this feature.](#)

To configure a message request:

1. Create and name the message request as described in [Creating a Service Request](#).
2. To enter recipients (To, Cc, or Bcc), select a recipient type:
 - ¤ Select **Address Book** to send a message to a single EnterpriseOne user. Enter the address book number of the EnterpriseOne user.
 - ¤ Select **Contact** to send a message to an individual in a user's contact list. Enter the address book number of a user and then the number of the contact.
 - ¤ Select **Distribution List** to send a message to the members of an EnterpriseOne distribution list. Enter the address book number of the list in the Parent Address Number field and select its structure type. For more information about structure types, see "Structure Types" in the *JD Edwards EnterpriseOne Tools Workflow Tools Guide*.
 - ¤ Select **Email** to enter an email address for the recipient.
3. In the Subject and body fields, enter text, variables, or a combination of both. To insert variables, see step 6.
4. To include boilerplate text from a message template in the data dictionary:
 - a. Expand the Data Dictionary Text section.
 - b. In the Data Item field, enter the name of the message template data item and click **Load**.
 - c. If the message template contains variables, use the grid in the right to override the variables with text substitution.
5. To include a shortcut to an application:
 - a. Expand the Shortcuts section.
 - b. Complete the Application, Form, and Version fields to specify the form that you want the shortcut to launch.
 - c. Specify the personal form, query, or watchlist that you want the shortcut to launch. (Orchestrator Studio 7.1.0.0)
 - d. Use the Pre Text, Link Text, and Post Text fields to define the text that appears in the message before, as, and after the link, respectively. (Orchestrator Studio 7.1.0.0)
 - e. In the grid, you can use variables to pass in data to the application when the application is launched from the shortcut.
 - f. Click the Add button in the Shortcuts section header and repeat these steps to include multiple shortcuts in a message. (Orchestrator Studio 7.1.0.0)
6. To include variables in the recipient types, subject, body, message template text, or shortcut:
 - a. Type \${var name} where var name is the name of the variable that you want to include.

Make sure the syntax includes the \$ sign and brackets, for example:

`${creditmanager}`

ID	Name	Description	Form Interconnect Value
7	mnAddressNumber	Address Number (AN8)	\${AbNum}
8	szCompany	Company (CO)	00000
11	mn_Tab_ToGoTo	Math Numeric 01 (MATH01)	1d
12	szCountryForPayroll	Country Code (CCPR)	

4.4.7 Configuring a Connector Service Request

This section contains the following topics:

- [Section 4.4.7.1, "Understanding Connector Service Requests"](#)
- [Section 4.4.7.2, "Before You Begin"](#)
- [Section 4.4.7.3, "Configuring a Connector Service Request to Invoke an Orchestration or Notification"](#)
- [Section 4.4.7.4, "Configuring a REST Connector to Invoke a REST Service"](#)
- [Section 4.4.7.5, "Configuring a REST Connector to Transfer Files to a REST Service"](#)
- [Section 4.4.7.6, "Configuring a Database Connector"](#)
- [Section 4.4.7.7, "Configuring a Connector to Transfer Files Using File Transfer Protocol \(FTP\)"](#)

4.4.7.1 Understanding Connector Service Requests

In the Orchestrator Studio, you can create a connector service request to enable an orchestration to:

- Invoke another orchestration or invoke a notification either on your local system or on an AIS Server on another EnterpriseOne system in a multisite operation.
- Invoke a REST service. Referred to as a REST connector, this enables outbound REST calls to external systems as an orchestration step. For example, an orchestration could

make a REST call to a Cloud service and use the data in the response in subsequent orchestration steps.

- ❑ Connect to a database. Referred to as a database connector, this enables orchestrations to read from and write to non-JD Edwards databases using standard SQL protocol. The database must support JDBC. A database connector enables external databases to be the source of input for orchestrations. It also allows data that is not appropriate for transaction tables to be stored for analysis, archive, or integration purposes.
- ❑ Retrieve a file from or send a file to a known location using FTP or SFTP protocols. This is referred to as an FTP connector. Starting with Orchestrator Studio 7.x.x, you can also use an FTP connector to retrieve data from a CSV file.
- ❑ Send a file to a known location using a REST protocol.

4.4.7.2 Before You Begin

A connector service request requires a connection soft coding record, which provides the connection that the connector uses to access resources on an external system. Ask your system administrator to create a connection. See [Creating Connection Soft Coding Records for Connector Service Requests](#) for details.

4.4.7.3 Configuring a Connector Service Request to Invoke an Orchestration or Notification

> **Tutorial:**

[Click here to view a recording of this feature.](#)

1. Create and name the connector service request as described in [Creating a Service Request](#).
2. On the Connector design page, click in the **Connection** field and select a connection under the Orchestrator category.

After you select a connection, the Orchestrator Studio displays the URL to the AIS Server next to the Connection field.

3. Click inside the Orchestration/Notification field and select an orchestration or notification from the drop-down list.

The drop-down list displays a list of available orchestrations followed by a list of available notifications. The orchestrations and notifications are further categorized by UDO status: Personal, Pending Approval, Reserved, or Shared.

Any inputs defined in the called orchestration or notification appear in the Orchestration Input column, with variable names automatically generated in the adjacent Input column.

4. In the Input column, you can modify the variable names as desired or map a hard coded value by entering a value (without the \${ } notation).
5. For an orchestration connector, use the adjacent Output grid to specify values you want returned from the called orchestration. These values must be defined as outputs in the called orchestration. Optionally, enter a variable for the value to make it an orchestration input, which gives you the option to map the value to a subsequent step in an orchestration.

4.4.7.4 Configuring a REST Connector to Invoke a REST Service

Configure a REST connector to invoke a REST service. In the REST connector, you specify an HTTP method and then provide the following additional details required for the connector to complete the request:

- ❑ The path to a particular endpoint in the REST service.

- Parameters, if required by the endpoint.
- Key-value pairs for the HTTP header.

In the Orchestrator Studio, you can test the connector and view the JSON response of the REST service call. You can also use Groovy scripting to refine the data in the connector output. For example, you can configure a Groovy script to refine the contents of the REST service response so that the connector output contains only the data required by the consuming device or program.

To configure a connector to invoke a REST service:

1. Create and name the connector service request as described in [Creating a Service Request](#).
2. On the Connector design page, click the **Connection** field and select a connection under the REST category.

The Orchestrator Studio displays the URL to the REST service available through the connection.

3. Click the **HTTP Method** drop-down list and select the appropriate method.
4. Slide the **Fire and Forget** toggle to the right if you want the orchestration to execute without waiting for a response. If Fire and Forget is enabled, the Output section is hidden because the REST service response is not processed.
5. Expand the Pathing section and use the Value grid to build the path to a REST service endpoint.

Each value you enter in the Value grid is appended to the path, separated by a forward slash (/). You can use variables in the path by adding a value inside \${ }. Click the **Delete** button at the end of a row to delete any values from the path.

The following image shows an example of a path to an endpoint that includes variables.

Value	
forecast	X
q	X
\${state}	X
\${city}.json	X
	X

6. If the REST service takes parameters, use the Parameters section to append parameters to the endpoint.

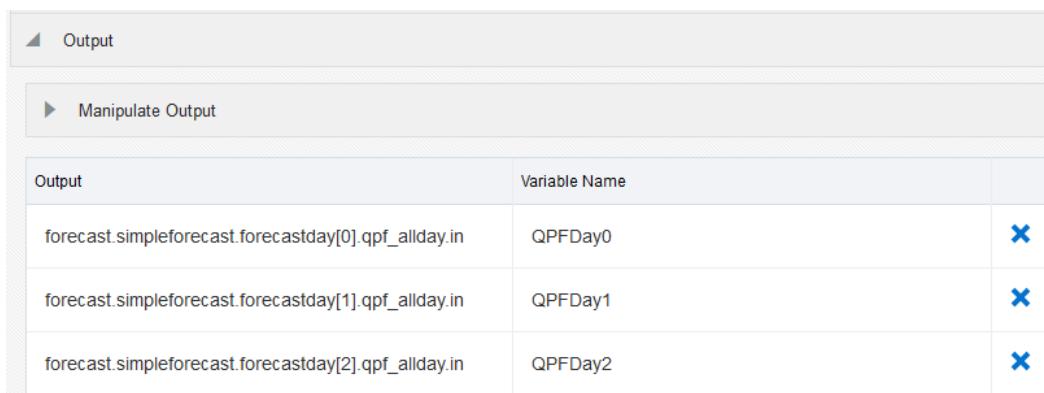
Parameters are appended to the endpoint after a "?" and are separated by a "&". You can include variables by specifying a value inside \${ }.

7. In the Headers section, enter key-value pairs for the header in the Key and Value columns. You can also choose any known values from the drop-down menu in each column.
8. In the Body section (which is not used or displayed if the HTTP method is GET or TRACE), if the HTTP method requires it, specify a payload to send to the REST service.
9. In the Output section, use the following sections to identify the outputs:

Note: The outputs are based on the JSON response of the REST service call. You can see the response after running a successful test of the REST service using the Test button.

- Manipulate Output (advanced). Use Groovy scripting to manipulate the output of the REST call. For example, if you only need certain data in the output, you can use Groovy scripting to refine the data displayed in the output. See [Groovy Template for Manipulating Output from a REST Connector Response](#) for more information.
- Output grid. If the value you want is nested in JSON objects, you can get to the value by separating each JSON object name by a period (.). If the value you want is in an array, you can access it by adding the proper index of the array inside [].

The following image shows an example of outputs defined in the Output section:



Output	Variable Name	
forecast.simpleforecast.forecastday[0].qpf_allday.in	QPFDAY0	X
forecast.simpleforecast.forecastday[1].qpf_allday.in	QPFDAY1	X
forecast.simpleforecast.forecastday[2].qpf_allday.in	QPFDAY2	X

10. To test the connector:
 - a. Click the **Test** button.
 - b. If the connector includes variables, enter values for the variables in the popup box.
If successful, the Orchestrator Studio displays the response.
 - c. Click the **Show Output** button displayed at the end of the response to apply the instructions defined in the Output section, including the Groovy script if specified.
Use the results to validate the path to the output values specified in the Output section.

4.4.7.5 Configuring a REST Connector to Transfer Files to a REST Service

You can configure a REST connector to transfer files to a REST service. When added to an orchestration that includes a report service request, a REST connector can transfer the output of the report service request to an external system. Also, if an orchestration includes a custom

service request that uses a Groovy script to generate and save a file to the AIS Server, you can create a REST connector to transfer the generated file to an external system.

1. Create and name the connector service request as described in [Creating a Service Request](#).
2. On the Connector design page, click the **Connection** field and select a connection under the REST category.

The Orchestrator Studio displays the URL to the REST service available through the connection.

3. Click the **HTTP Method** drop-down list and select **POST**.
4. Enable the **Fire and Forget** toggle if you want the orchestration to execute without waiting for a response.

If Fire and Forget is enabled, the Output section is hidden because the REST service response is not processed.

5. In the File section of the request, enter the File Part Name, which you can find in the documentation for the REST API you are calling.
6. Select the **Report** or **File** option to specify the type of file you are sending, and then complete the following fields accordingly:

For Report, complete these fields:

- **Job Number.** Enter `${variablename}` to use a variable for the job number. When you add this connector to an orchestration, you can map the job number of a report returned from a report service request to this variable. Or instead of a variable, you can enter an actual job number.
- **Execution Server.** Enter the server where the report was generated. Enter `${variablename}` to use a variable for the server name. When you add this connector to an orchestration, you can map the name of this server from the report service request, which returns the name of this server. Or instead of variable, you can enter the server name.
- **File Type.** Select the file type of the output.

For File, complete these fields:

- **Source File Name.** Enter the name of the file that was created in the custom Groovy service request.
 - **Remove File.** Enabled by default, this removes the file from the temporary directory on the AIS Server after the file is transferred.
7. If the REST API being called requires additional information, you can use the table at the bottom of the File section to include additional details.

4.4.7.6 Configuring a Database Connector

You must have knowledge of Groovy scripting language to create a database connector in order to route data to a database. The database must support JDBC. The Connector design page provides a Groovy template that you can use as a basis for creating a Groovy script for a database connector.

1. Create and name the connector service request as described in [Creating a Service Request](#).
2. On the Connector design page, click the **Connection** field and select a connection under the Database category.

The Orchestrator Studio displays an edit area that contains a Groovy script template. Use the Find and "Go to Line" fields and Undo and Redo buttons to work with the script.

3. In the Input grid, enter the inputs that you want to pass to the database. You can enter hard coded values or include variable using the \${ } notation, entering the variable name within the brackets.
4. In the Output column, list the fields added to the returnMap in the Groovy script to make those outputs available to the orchestration. Optionally, enter a variable name in the Variable column if you want to make the values available for mapping to a subsequent step in an orchestration.
5. Click **Save**.

4.4.7.7 Configuring a Connector to Transfer Files Using File Transfer Protocol (FTP)

You can configure a connector to transfer files using FTP or secure file transfer protocol (SFTP). With this type of connector, referred to as an FTP connector, you can:

- ❑ Transfer the output of a report service request to an FTP server.
The connector can transfer the output of a standard EnterpriseOne report or an embedded Oracle BI Publisher report.
- ❑ Transfer other types of files from an FTP server to a temporary directory on the AIS Server.
An administrator must define the temporary directory in the basic configuration settings for the AIS Server. For more information, see [Set Up a Temporary Directory on the AIS Server for File Transfers](#).
- ❑ Transfer files in the temporary directory on the AIS Server to an FTP server.
- ❑ (Orchestrator Studio 7.0.0.0) Retrieve data from a CSV file. The data is imported as an array, which gives you the option to map the data in the array to a subsequent orchestration step.

Note: You can also use a REST connector to transfer a report or file to an external location. See [Configuring a REST Connector to Transfer Files to a REST Service](#)

An FTP connector uses a secure connection created through a soft coding record to receive and send a file from an FTP server.

An FTP connector can transfer only one file at a time.

Configure an FTP Connector to Transfer the Output of a Report Service Request

1. Create and name the connector service request as described in [Creating a Service Request](#).
2. On the Connector design page, click the **Connection** field and select a connection under the FTP category.

If there are no connections available for an FTP connector, ask your system administrator to create one. See [Creating a Soft Coding Record for an FTP Server Connection \(Orchestrator Studio 6.1.0\)](#).

3. Click the **Report** option and complete these fields:
 - ❑ **Job Number.** By default, this field contains a variable. When you add this connector to an orchestration, you can map the job number of a report returned from a report service request to this variable. You can modify the variable name if desired. Or you

can delete the variable (including the special characters) and replace it with an actual job number.

- **Execution Server.** Enter the server where the report was generated. By default, this field contains a variable. When you add this connector to an orchestration, you can map the name of this server from the report service request, which returns the name of this server. You can modify the variable name if desired. Or you can delete the variable (including the special characters) and replace it with a server name.
- **File Type.** Select the type of file that the FTP connector will transfer. The default is PDF for standard EnterpriseOne reports. If transferring an embedded BI Publisher report, then select one of the BI Publisher file types.
- **Path Extension.** Enter the name of the sub-directory on the third-party FTP server where you want the report output sent. The FTP Connection already contains information for the base folder.

4. Click **Save**.

Configure an FTP Connector to Transfer a File from an FTP Server to the AIS Server

1. Create and name the connector service request as described in [Creating a Service Request](#).
2. On the Connector design page, click the **Connection** field and under the FTP category, select a connection to the server from which you want to transfer a file.
If there are no FTP connections available, ask your system administrator to create one. See [Creating a Soft Coding Record for an FTP Server Connection \(Orchestrator Studio 6.1.0\)](#).
3. Select the **File** option.
4. Click the **Type** drop-down list and select **Receive**.
5. In the **Source File Name** field, enter the name of the file that you want to retrieve.
6. In the **Target File Name** field, enter the name of the target file if you want it to be saved with a different name than the source file when saved to the temporary directory on the AIS Server.
7. Click the **Use Temporary File Location** check box to save the file to the temporary directory on the AIS Server.
8. Enable or disable the **Remove File** toggle. Enabled by default, this removes the file from the temporary location on the AIS Server after it is transferred.
9. Click **Save**.

Configure an FTP Connector to Transfer a File from the AIS Server to an External Server

1. Create and name the connector service request as described in [Creating a Service Request](#).
2. On the Connector design page, click the **Connection** field and under the FTP category, select a connection to the server to which you want to transfer the file.
If there are no FTP connections available, ask your system administrator to create one. See [Creating a Soft Coding Record for an FTP Server Connection \(Orchestrator Studio 6.1.0\)](#).
3. Select the **File** option.
4. Click the **Type** drop-down list and select **Send**.
5. In the **Source File Name** field, enter the name of the file that you want to transfer.

6. Click the **Use Temporary File Location** check box to save the file to the temporary directory on the AIS Server.
7. In the **Target File Name** field, enter the name of the target file if you want it to be saved with a different name than the source file when saved to the AIS Server directory.
8. Enable or disable the **Remove File** toggle. Enabled by default, this removes the file from the temporary location on the AIS Server after it is transferred to an external server.
9. Click **Save**.

Configuring an FTP Connector to Import Data from a CSV File (Orchestrator Studio 7.0.0.0)

Use an FTP connector to retrieve data from a CSV file on an FTP server. The data is imported as an array, which gives you the option to map the data set in the array to a subsequent orchestration step. See [Passing a Data Set to a Subsequent Orchestration Step](#) for more information.

1. Create and name the connector service request as described in [Creating a Service Request](#).
2. On the Connector design page, click the **Connection** field and under the FTP category, select a connection to the FTP server where the CSV file is located.

If there are no FTP connections available, ask your system administrator to create one. See [Creating a Soft Coding Record for an FTP Server Connection \(Orchestrator Studio 6.1.0\)](#).

3. Select the **File** option.
4. Click the **Type** drop-down list and select **Receive**.
5. Click the **Import CSV as Array** check box,

The Orchestrator Studio displays additional fields to configure the importing of data from a CSV file.

6. If the CSV file has a header row, click the **CSV Has Headers** check box to exclude the header row.

This ensures that at runtime, the Orchestrator will not import the column headers as data in the array.

7. Complete the following fields:

- ❑ **Delimiter**. Enter the delimiter used in the CSV file to separate values.
- ❑ **Source File Name**. Enter the name of the CSV file.
- ❑ **Path Extension**. Enter the name of the sub-directory on the third-party FTP server that contains the CSV file. The FTP connection already contains information for the base folder.
- ❑ **Data Set Variable Name**. Enter a name for the data set or array in which the data from the CSV file will be imported.

Later, you can reference this data set variable name to map data in the data set to a subsequent orchestration step.

8. Enable or disable the **Remove File** toggle. Enabled by default, this removes the file from the temporary location on the AIS Server after the data is imported.
9. In the Model CSV File section, select a model CSV file to define the columns in the CSV file that you want to import.

The CSV file should be a copy of the CSV file from which the FTP connector will import data, or you can use a different CSV file as long as it has the same column names.

- a. Click **Browse** to select a file and then click **Ok**.

If the CSV Has Headers check box is selected, the grid displays the column names from the CSV file in Member Name and Variable Name columns. Otherwise, the grid displays column 1, column 2, and so forth.

- b. In the grid, you can change the member names or variable names, and you can use the **X** button at the end of the rows to remove any columns from the import.

The variable names are used to represent the values imported into the array. If you map these values to a subsequent orchestration step, you will select these variable names for the mappings.

Caution: When importing the CSV data into an array, the order of the columns, not the names of the columns, is respected. If the CSV has 10 columns and you only specify eight in this grid, the array will contain the first eight columns from the CSV. If column headers are present, the third column in the CSV file will be mapped to the third row of this table, regardless if the column and member names match.

10. Click **Save**.

4.4.8 Configuring a Watchlist Service Request

Use a Watchlist service request to use the information from Watchlists, such as critical or warning states, threshold levels, and number of records, within an orchestration. For more information about the data you can retrieve from a Watchlist, see *JD Edwards EnterpriseOne Applications One View Watchlists Implementation Guide*.

1. Create and name the Watchlist service request as described in [Creating a Service Request](#).

On the Watchlist page, the grid displays all Watchlists that you have been authorized to access through UDO security in EnterpriseOne.

If you need a Watchlist not available in this list, ask your EnterpriseOne administrator to grant you access to the Watchlist.

2. In the Name column in the grid, click the link to the Watchlist that you want this service request to access.

The Watchlist displays information about the Watchlist and areas for selecting the Watchlist details you want returned. To select a different Watchlist, click **Change Watchlist** to return to the list of Watchlists.

3. In the Outputs area, select the Watchlist data you want returned.
4. In the Advanced Outputs area, select any additional details about the Watchlist that you want returned.
5. Click the **Force Watchlist To Update** check box if you want the service request to refresh the Watchlist data in EnterpriseOne before returning Watchlist data. (Recommended)

4.4.9 Configuring a Report Service Request

Use a report service request to invoke a batch version of an EnterpriseOne report from an orchestration. You can configure a report service request to use the settings defined for the report in the Batch Versions program. Or you can configure it to override the processing options, data selection, and data sequencing of a report.

You can also create a report service request to invoke an embedded BI Publisher report, and configure it to use the default settings or override the settings.

A report service request can launch EnterpriseOne reports designed with report interconnects. Report interconnects are inputs added to a report at design time, which enable a report to run automatically without user interaction. Typically, report interconnects are used to launch a report from a button or exit on an EnterpriseOne form.

In the Report design page, the "Fire and Forget" option enables the report to run asynchronously. When the Orchestrator executes the orchestration, this option enables the orchestration to continue to the next action or step without waiting for the report to complete.

At runtime, when the Orchestrator processes the report service request, it respects the EnterpriseOne authorization security for the report. The user initiating the orchestration must be authorized to run the batch version of the report as defined by an administrator in the Security Workbench. If the report service request includes data selection or processing option overrides, the user initiating the orchestration must be authorized in the Security Workbench to perform these overrides as well.

To configure a report service request:

1. Create and name the report service request as described in [Creating a Service Request](#).
2. In the Report field, enter the ID of the report that you want the report service request to invoke.

The Orchestrator Studio loads all versions associated with the report in the Version drop-down list.

3. Click the **Version** drop-down list and select a version of the report.

The Orchestrator Studio loads any settings defined for the version in EnterpriseOne, including data selection, data sequencing, processing options, and output options.

4. Enable **Fire and Forget** if you want the report to run asynchronously.

If you do not enable this setting, if added to an orchestration that has steps following the report service request, the remaining steps will not execute until the report finishes.

5. Select one of the following options to determine how to run the report:

❑ **Blind Execution.** Select this option if you want the report service request to invoke the batch version using the settings defined in EnterpriseOne for the report version.

You can review the settings defined for the version by expanding the Data Selection, Data Sequencing, and Processing Options sections.

❑ **Report Interconnects.** If a report is defined with report interconnects, select this option and enter the values you want to pass into the report in the available fields. This option is disabled if no report interconnects were defined for the report.

❑ **Override Version.** Select this option to override the settings in EnterpriseOne for the version of the report, and then perform these steps:

- a. For Data Selection and Data Sequencing, expand the sections to modify existing criteria, or click the **Add** button to define new criteria.
- b. Expand the Processing Options section and change the settings as desired.

6. Click the **Queue Name** drop-down list and select the queue to which the reported is submitted. If you leave this field blank, when this report is invoked, the system uses the default queue defined in the Enterprise Server configuration settings.

7. To modify the print properties defined for the report, click **Output Options** and complete the fields as appropriate.

For more information about report output options, see "Report Output" in the *JD Edwards EnterpriseOne Tools Report Printing Administration Technologies Guide*.

8. To enable logging for the report, click **Output Options** and then enable the **JDE Log** toggle. Enter a value from 0–6 in the Logging Level field to indicate the level of detail to be captured in the logs.

Any value greater than 0 will force the system to write both the JDE and JDEDEBUG logs. When you select a high value to receive more technical information, you also receive all the information for the lower values. For example, when you enter a value of 3 (object level messages), you also receive information for 2 (section level messages), 1 (informative messages), and 0 (error messages).

4.4.10 Configuring Form Request and Data Request Processing

In the Orchestrator Studio, you can configure settings that control how the AIS Server processes a form request or data request in an orchestration at runtime.

For form requests, these settings are available from the Options icon at the end of the first row of each form listed in the Available Actions grid.

For data requests, these settings are available from the Options button on the Data Request design page.

The settings include:

- **Maximum Records.** 0 is for All Records.
- **Query Name.** From this drop-down list, you can select a predefined query to use for the filtering criteria. You can use a query instead of or in combination with the filtering criteria defined in a data request. The queries that you can see in this list are based on UDO security permissions.
- **Output Type.** Select one of the following format types for the format of the JSON response. Formats include:
 - **Default.** The default format is Version 2 output type for version 1 orchestrations (orchestrations created prior to Orchestrator Studio 5.0.1). The default format is Grid Data output type for version 2 orchestrations (orchestrations created in Orchestrator Studio 5.0.1 and higher).
 - **Grid Data.** Returns grid data in simplified name-value pairs. This is the default output type for version 2 orchestrations (orchestrations created in Orchestrator Studio 5.0.1 and higher).
 - **Version 2.** Returns cell-specific information for each grid row as well as information about each column such as column ID, whether it is visible, editable, and so forth. This is the default output type for orchestrations created prior to Orchestrator Studio 5.0.1, which are referred to as version 1 orchestrations.
- **Stop on Warning** check box (form requests only). Click this check box if you want the processing to stop if the invoked EnterpriseOne form produces a "Warning" message. Keep this check box clear if you want processing to continue after a warning message.
- **Turbo Mode** (form requests only). Use this option to reduce processing time of form requests. See "Using Turbo Mode" in the *JD Edwards EnterpriseOne Application Interface Services Client Java API Developer's Guide* for more information.
- **Caching: Allow.** Click this check box to enable caching of the service request response. If the "Enable Caching by Default" option is enabled in Server Manager, this parameter is ignored and all responses for Read operations will be cached.

- **Caching: Force Update.** Click this check box to force the system to fetch the data from the database for the service request. If the "Enable Caching by Default" setting is enabled in Server Manager, then this parameter for the individual service request is ignored.
- **Caching: Time Stored - Milliseconds.** Enter the amount of time in milliseconds for the service request to use the cache for the response. This setting overrides the value in the "Default Read Cache Time To Live" setting in Server Manager.

4.4.11 Configuring a Custom Service Request with Java (Advanced)

You can create a service request that uses custom Java to execute a custom process or to route data into another database. Before you can create a custom Java service request, you must create the custom Java as described in [Chapter 7, "Creating Custom Java for Orchestrations"](#).

1. After naming the custom service request as described in [Creating a Service Request](#), select the **Java** option.
2. In the **Fully Qualified Class** field, enter the Java class.
This is the custom Java class that you created for the service request.
3. In the first grid, complete the following fields:
 - **Input.** Enter the name of the field in the class.
 - **Input Value.** Enter the input variable name. If the attribute is a boolean and you pass in "true", then you could use a default value.
 - **Default Value.** Enter a default, hard-coded value if there is no mapped value. You can also add a hard-coded value to use if the mapped value returns a null.
4. (Optional) In the second grid, enter a variable name for an output if you want to use the output value in subsequent steps in the orchestration or to another orchestration.
5. If you make a mistake, click the **Restore Custom Java** button to return the custom Java to its last saved state.
6. Click the **Save** button.

The Orchestrator Studio saves the custom Java request. You can then access the orchestration in the Orchestration design page and add this custom Java service request as a step in the orchestration.

4.4.12 Configuring a Custom Service Request with Groovy (Advanced)

You can create a service request that uses Groovy scripting to execute a custom process or to route data into another database. To use Groovy for a service request, the Orchestrator Studio provides an edit area that contains a sample Groovy script with instructions on how to modify the script. You can use the Find and "Go to Line" fields and Undo and Redo buttons to work with the script. [Chapter 8, "Using Apache Groovy for Custom Service Requests, Rules, and Manipulating Output \(Orchestrator Studio 5.1.0 and Higher\)"](#) provides information on how to work with the sample Groovy script.

To configure a customer service request with Groovy:

1. After naming the custom service request as described in [Creating a Service Request](#), select the **Groovy** option.
2. Configure the script to perform the desired action.
3. In the "Input" grid, enter the names of the inputs.

The inputs will be placed in the inputMap of the "main" function and can be retrieved in the script by following the commented out example code.

4. Click the **Load Outputs button.**

This reads the script and adds any values added to the returnMap in the script to the Output grid.

5. (Optional) In the "Output" grid, enter a variable name for an output if you want to use the output value in a subsequent orchestration step.

When you add this service request to an orchestration, this name appears in the orchestration inputs grid, which makes the returned value available for mapping to subsequent steps in the orchestration.

Note: Even if no variables are used, all defined outputs are available for mapping using the Orchestration Output feature. See [Working with Orchestration Output](#) for more information.

6. To test the script:

- Enter a value in the Test Value column for one or more inputs.
- Click the **Test** button.

Orchestrator Studio executes the script using the inputs you entered. If successful, it populates the results in the Test Output column of the Output grid.

If you included orchAttr.writeWarn or orchAttr.writeDebug statements in the script, a Logging popup displays after execution. At runtime, log statements are included in the AIS Server log, which can be used for debugging script issues.

The following example shows the Logging popup, which you can use to verify the values passed to the inputs:

The screenshot shows the Orchestrator Studio interface. At the top, there are tabs for 'Groovy' (selected) and 'Java'. Below the tabs is a toolbar with icons for back, forward, find, go to line, and search. A 'Logging' window is open, displaying a list of log entries. The entries show various system logs and a log entry from line 62: 'averagePrecip 2'. A red box highlights this entry. Below the Logging window is an 'Output' grid. The grid has two columns: 'Input' and 'Test Value'. The 'Input' column lists 'averagePrecip', 'averageLow', 'averageHigh', and 'soilMoisture'. The 'Test Value' column contains '2' for 'averagePrecip', and 'X' marks for the other three inputs. A red box highlights the '2' in the 'Test Value' cell for 'averagePrecip'. At the bottom of the grid are 'Load Outputs' and 'Test' buttons.

Input	Test Value
averagePrecip	2
averageLow	X
averageHigh	X
soilMoisture	X

7. Click the **Save button.**

The Orchestrator Studio saves the custom Groovy service request. You can then access the orchestration in the Orchestration design page and add this custom Groovy service request as a step in the orchestration.

4.5 Creating Rules

This section contains the following topics:

- ❑ [Section 4.5.1, "Understanding Rules"](#)
- ❑ [Section 4.5.2, "Creating a Rule"](#)
- ❑ [Section 4.5.3, "Creating a Custom Rule with Java \(Advanced\)"](#)
- ❑ [Section 4.5.4, "Creating a Custom Rule with Groovy \(Advanced\)"](#)

4.5.1 Understanding Rules

A rule contains conditional logic that the Orchestrator uses to evaluate conditions, such as true or false conditions that determine how the orchestration processes the incoming data. You can define a rule with a list of conditions or you can define a more complex rule using Groovy or a custom Java class.

An orchestration rule with conditions functions similar to an EnterpriseOne query in that each rule has:

- ❑ A "Match Any" or "Match All" setting.
- ❑ One or more conditions defined, each being a binary operation on two values.

After creating a rule and adding it to an orchestration, you use the Action column in the Orchestration design page to determine the orchestration step that is invoked when the conditions in the rule are met. See [Defining the Actions Between a Rule and Dependent Components](#) for more information.

4.5.2 Creating a Rule

Create a rule to define conditions for an orchestration.

> Tutorial:

[Click here to view a recording of this feature.](#)

To create a rule:

1. On the Orchestrator Home page, click the **Rules** icon. And then on the Rules design page, click the **New Rule** button.

Alternatively, access the Orchestrations design page and add a Rule step to an orchestration. At the end of row with the Rule step, click **Edit** (pencil icon) and select **Rule** from the pop-up box.

2. On the Rules design page, enter a name for the rule in the **Rule** field. Do **NOT** include special characters in the name.
3. Click the **Product Code** drop-down list to select a product code to associate with the rule.

This gives an administrator the option to manage UDO security for orchestration components by product code.

4. In the space provided, enter a short description with a maximum of 200 characters. This description appears below the rule name in the component list.

5. Click the **Edit Long Description** button to add a long description to provide more detail about the purpose of the component.
6. Select the Match Value drop-down menu and select one of the following values:
 - **Match All.** Select this option if all conditions in the rule must be met.
 - **Match Any.** Select this option if any conditions in the rule can be met.
7. In the first row in the grid, complete the following columns to add a condition:
 - **Rule Type.** Click the drop-down menu and select String, Numeric, or Date depending on the format of the input.
 - **Value 1.** Enter a variable for the input name.
 - **Operator.** In the drop-down menu, select from the list of operands which include: >, <, >=, <=, =, !=, startsWith, endWith, contains, between, inList.
 - **Literal.** Click this check box to indicate a literal value.
 - **Value 2.** If you selected the Literal check box, manually enter a value.
 - **Literal Value Type.** If you selected the Literal check box, click the drop-down menu and select the format of the literal value: string, numeric, data.
8. Add additional conditions to the rule as needed. If you add a condition by mistake, you can delete it by clicking the **Remove** button at the end of the row with the condition.
9. Click the **Save** button.

The first time a new rule is saved, it is saved as a "Personal" UDO. Thereafter, you can use the UDO buttons described in the [User Defined Object \(UDO\) Features in the Orchestrator Studio](#) section to move the rule to the appropriate status.

After adding a rule and a service request to an orchestration, in the Orchestration design page, you must define the action for the rule to invoke the service request. See [Defining the Actions Between a Rule and Dependent Components](#) for more information.

4.5.3 Creating a Custom Rule with Java (Advanced)

You can create complex rule using custom Java. Before you add a custom Java rule in the Orchestrator Studio, you must create a custom Java class to use for the rule as described in [Chapter 7, "Creating Custom Java for Orchestrations"](#).

To create a custom Java rule:

1. Click the **Rules** icon on the Orchestrator Studio Home page, and on the Rules page, click the **New Custom** button.

Alternatively, access the Orchestrations design page and add a Rule step to an orchestration. At the end of row with the Rule step, click **Edit** (pencil icon) and select **Custom** from the pop-up box.

2. On the Rules design page, select the **Java** radio button.
3. In the Rule field, enter a name for the custom rule. Do **NOT** include special characters in the name.
4. Click the **Product Code** drop-down list to select a product code to associate with the rule. This gives an administrator the option to manage UDO security for orchestration components by product code.
5. In the space provided, enter a short description with a maximum of 200 characters. This description appears below the rule name in the component list.

6. Click the **Edit Long Description** button to add a long description to provide more detail about the purpose of the component.
7. In the Fully Qualified Class field, enter the fully qualified path to the Java class, which includes the name of the Java class.

This is the custom Java class that you created to use for the rule.
8. Complete the following fields in the grid:
 - **Attribute.** Enter the name of the field in the class.
 - **Input Value.** Enter a variable for the input name. If the attribute is a boolean and you pass in "true", then you could enter a hard-coded default value.
 - **Default Value.** Enter a hard-coded value if there is no mapped value. You can also add a hard-coded value to use if the mapped value returns a null.
9. If you make a mistake while configuring any of the fields, you can click the **Restore Rule** button which refreshes the custom Java rule to its last saved state.
10. Click the **Save** button.

4.5.4 Creating a Custom Rule with Groovy (Advanced)

Use Groovy to create a custom rule when conditions in a standard rule will not suffice.

To create a custom rule with Groovy:

1. Click the **Rules** icon on the Orchestrator Studio Home page, and on the Rules page, click the **New Custom** button.

Alternatively, access the Orchestrations design page and add a Rule step to an orchestration. At the end of row with the Rule step, click **Edit** (pencil icon) and select **Custom** from the pop-up box.
2. On the Rules design page, enter a name for the custom rule in the **Rule** field. Do **NOT** include special characters in the name.
3. Click the **Product Code** drop-down list to select a product code to associate with the rule.

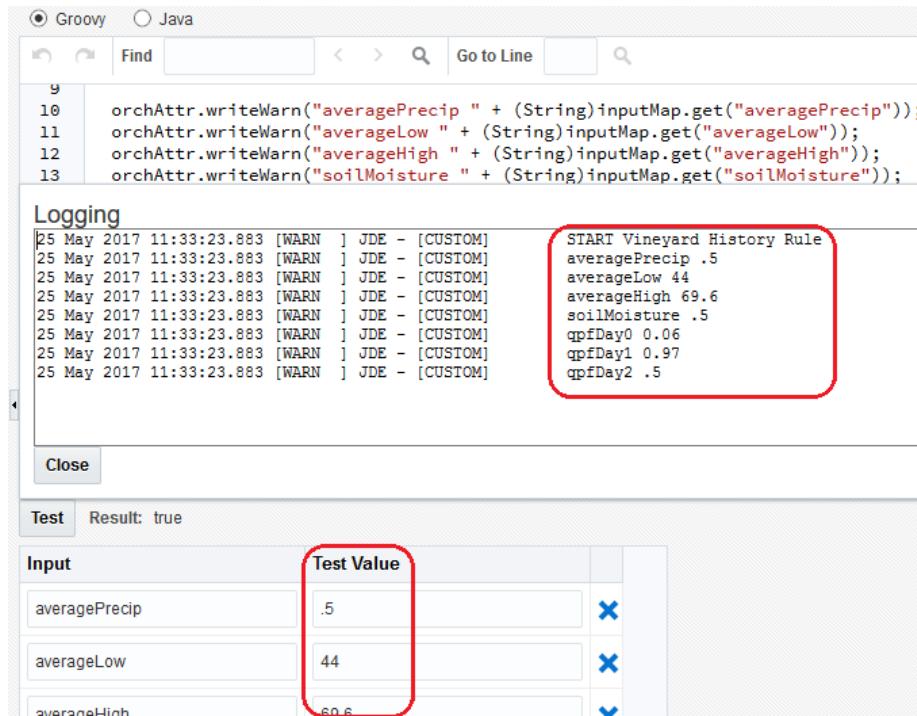
This gives an administrator the option to manage UDO security for orchestration components by product code.
4. In the space provided, enter a short description with a maximum of 200 characters. This description appears below the rule name in the component list.
5. Click the **Edit Long Description** button to add a long description to provide more detail about the purpose of the component.
6. Select the **Groovy** radio button.

The Orchestrator Studio displays an edit area that contains a sample groovy script with instructions on how to work with the script. Use the Find and "Go to Line" fields and Undo and Redo buttons to help edit the script. See [Chapter 8, "Using Apache Groovy for Custom Service Requests, Rules, and Manipulating Output \(Orchestrator Studio 5.1.0 and Higher\)"](#) for more information about the sample Groovy script.

7. Configure the script to perform the desired action.
8. In the "Input" grid, enter the names of the inputs.
9. To test the script:
 - a. Enter a value in the Test Value column for one or more inputs.
 - b. Click the **Test** button.

If you included `orchAttr.writeWarn` or `orchAttr.writeDebug` statements in the script, a Logging popup displays after execution. At runtime, log statements are included in the AIS Server log, which can be used for debugging script issues.

A Logging dialog box displays the test results which you can use to verify the values passed to the inputs, as shown in the following example:



- 10.** Click the **Save** button.

4.6 Creating Cross References

This section contains the following topics:

- ⌘ Section 4.6.1, "Understanding Cross References"
 - ⌘ Section 4.6.2, "Creating a Cross Reference"

4.6.1 Understanding Cross References

The Orchestrator uses a cross reference component to map incoming data (third-party data) to an EnterpriseOne value. The EnterpriseOne value identified in the cross reference is considered the output of the cross reference. The output from the cross reference becomes the data passed to another orchestration step.

For each cross reference that you create in the Orchestrator Studio, you have to create a cross reference record in P952000 in EnterpriseOne. When defining cross reference records for orchestrations in P952000, you must use the "AIS" cross reference type. For more information on how to create these cross reference records in P952000, see [Chapter 5, "Setting Up Cross References and White Lists in EnterpriseOne \(P952000\)"](#) in this guide.

Note: If a cross reference lookup fails in an orchestration, the orchestration is terminated.

4.6.2 Creating a Cross Reference

You must define the orchestration inputs before you can create a cross reference. See [Adding Inputs to an Orchestration](#) for more information.

To create a cross reference:

1. Access the Cross Reference design page:

- ❑ On the Orchestrator Home page, click the **Cross References** icon. And then on the Cross References design page, click the **New Cross Reference** button.
OR
- ❑ After adding a Cross Reference step to the grid in the Orchestration design page, click the **Edit** button next to the step.

The Orchestrator Studio displays the Cross Reference design page.

2. In the Cross Reference field, enter a name for the cross reference. Do **NOT** include special characters in the name.
3. In the space provided, enter a short description with a maximum of 200 characters. This description appears below the cross reference name in the component list.
4. Click the **Edit Long Description** button to add a long description to provide more detail about the purpose of the component.
5. In the Object Type field, enter a name for the object type.

This is the object type used to categorize the orchestration cross references in EnterpriseOne. See "Adding Orchestration Cross References" in the *JD Edwards EnterpriseOne Tools Interoperability Guide* for more information.

6. Click the **Product Code** drop-down list to select a product code to associate with the cross reference.

This gives an administrator the option to manage UDO security for orchestration components by product code.

7. Complete the Input Key and Output Key columns to map the inputs to EnterpriseOne fields:
 - ❑ **Input Key.** The inputs can be one or many values. The inputs must correspond to the value or pipe (|) delimited values in the Third Party Value column in P952000. See "[Chapter 5, "Setting Up Cross References and White Lists in EnterpriseOne \(P952000\)"](#) for more information.
 - ❑ **Output Key.** The outputs can be one or many values. The outputs must correspond to the value or pipe (|) delimited values in the EOne Value column in P952000. See "[Chapter 5, "Setting Up Cross References and White Lists in EnterpriseOne \(P952000\)"](#) for more information.
8. If you enter any values by mistake, you can click the **X** button next to the field to delete the entry.
9. Click the **Save** button, which saves the white list as a "Personal" UDO.

The first time a new cross reference is saved, it is saved as a "Personal" UDO. Thereafter, you can use the UDO buttons described in the [User Defined Object \(UDO\) Features in the Orchestrator Studio](#) section to move the rule to the appropriate status.

The output values in the cross reference are now available for mapping in subsequent orchestration steps.

4.7 Creating White Lists

This section contains the following topics:

- [Section 4.7.1, "Understanding White Lists"](#)
- [Section 4.7.2, "Creating a White List"](#)

4.7.1 Understanding White Lists

A white list contains a list of IDs permitted in the Orchestrator. By adding a white list to an orchestration, only inputs with IDs listed in the white list are permitted for processing by the Orchestrator. You add a white list component as a step in the orchestration.

In addition to specifying the permitted IDs in the white list, you must also specify the white list IDs in P952000 in EnterpriseOne. This is the same application that you use to set up and store orchestration cross references. As with cross references, use the "AIS" cross reference type in P952000 for a white list. In P952000, the Third Party App ID should have a value of WHITELIST. The EnterpriseOne value is not used for white lists and will default to NA.

If a white list lookup fails in an orchestration, the orchestration is terminated.

4.7.2 Creating a White List

1. Access the White List design page:

- On the Orchestrator Home page, click the **White Lists** icon. And then on the White Lists design page, click the **New White List** button.
- OR**
- After adding a White List step to the grid in the Orchestration design page, click the **Edit** button next to the step.

The Orchestrator Studio displays the White Lists design page.

2. In the White Lists design page, click the **New White List** button.
3. In the White List, enter a name for the white list. Do **NOT** include special characters in the name.
4. In the space provided, enter a short description with a maximum of 200 characters. This description appears below the white list name in the component list.
5. Click the **Edit Long Description** button to add a long description to provide more detail about the purpose of the component.
6. In the Object Type field, enter a name for the object type.

The value you enter in the Object Type field must match a cross reference object type in the EnterpriseOne Business Services Cross Reference application (P952000).

The cross reference object type is a named group of records in P952000. For example, you may have thousands of records in P952000. You can use cross reference object types, for example "Equipment" or "Alert_Notification_Recipients" to group cross reference records into manageable categories. You define the cross reference object types as needed. See [Chapter 5, "Setting Up Cross References and White Lists in EnterpriseOne \(P952000\)"](#) for more information.

7. Click the **Product Code** drop-down list to select a product code to associate with the white list.

This gives an administrator the option to manage UDO security for orchestration components by product code.

8. In the Input Key column, enter the input that you want to add as a permitted input.
9. Click the **Save** button.

The first time a new white list is saved, it is saved as a "Personal" UDO. Thereafter, you can use the UDO buttons described in the [User Defined Object \(UDO\) Features in the Orchestrator Studio](#) section to move the white list to the appropriate status.

4.8 Creating Orchestrations

This section contains the following topics:

- ¤ [Section 4.8.1, "Understanding Orchestrations"](#)
- ¤ [Section 4.8.2, "Creating an Orchestration"](#)
- ¤ [Section 4.8.3, "Adding Inputs to an Orchestration"](#)
- ¤ [Section 4.8.4, "Adding Steps to an Orchestration"](#)
- ¤ [Section 4.8.5, "Mapping Orchestration Inputs"](#)
- ¤ [Section 4.8.6, "Retrieving and Passing Data Sets in an Orchestration"](#)
- ¤ [Section 4.8.7, "Working with Orchestration Output"](#)

4.8.1 Understanding Orchestrations

Creating an orchestration in the Orchestrator Studio involves:

- ¤ Naming the orchestration and specifying the input format for the orchestration.
The input format can be JDE Standard, Oracle Cloud IoT, or Generic.
- ¤ Adding inputs to the orchestration.
The inputs define the data passed to the orchestration from a device, third-party application, custom program, or Cloud service. For example, if designed to receive input from a sensor, the input may include an ID that identifies the sensor and other data that the orchestration will receive, such as temperature, date, or any other data you want to capture from the sensor.
- ¤ Adding steps to the orchestration.
Each step is a component of the orchestration: a service request, rule, cross reference, or white list. At a minimum, an orchestration requires only a service request step, which provides the actions or the instructions to perform a particular task in EnterpriseOne.
- ¤ Configuring transformations.
You use transformations to map orchestration inputs to inputs defined in each orchestration step, such as inputs in a rule, cross reference, white list, or service request component.

Important: Remember that when you are ready to "request to publish" an orchestration, you need to make sure that you also request to publish all components associated with the orchestration. This enables an administrator to save all required components to the proper directory on the AIS Server for processing by the Orchestrator. If a component is missing, the orchestration process will end in error.

4.8.2 Creating an Orchestration

> **Tutorial:**

See the following tutorials on how to create an orchestration:

[Creating and Testing an Orchestration](#)

[Creating an Orchestration with Multiple Components](#)

To create an orchestration:

1. On the Orchestrator Studio Home page, click the **Orchestrations** icon.
2. On the Orchestrations page, click the **New Orchestration** button.
3. On the Orchestration design page, enter a unique name for the orchestration in the Orchestration field. Do **NOT** include special characters in the name.

All orchestrations created in Orchestrator Studio 5.0.1 or higher are saved as version 2 orchestrations. You cannot change this value in the Orchestrator Version drop-down list. If you want to update a version 1 orchestration created in a previous version of the Orchestrator Studio, see [Updating Version 1 Orchestrations to Version 2 Orchestrations](#).

Note: When you save an orchestration, the orchestration name is used to define an endpoint on the AIS Server. The endpoint URL is:

`http://<server>:<port>/jderest/orchestrator/<orchestrationname>`

To invoke this orchestration, third-party applications or devices use a post operation to this url, where <orchestrationname> is the name of the orchestration.

4. Click the **Product Code** drop-down list to select a product code to associate with the orchestration.

This gives an administrator the option to manage UDO security for orchestration components by product code.

5. In the space provided, enter a short description with a maximum of 200 characters. This description appears below the orchestration name in the component list.

6. Click the **Edit Long Description** button to provide more detail about the component.

Use this field to describe the purpose of the orchestration and any details that differentiate the orchestration from other orchestrations that you create.

7. Click the **Input Format** drop-down menu and select the appropriate format:

- ❑ **JDE Standard** (default)
- ❑ **Oracle Cloud IoT**
- ❑ **Generic**

See [Supported Input Message Formats](#) for more information about which input format to use.

8. At this point, you can click **Save** before defining inputs or steps for the orchestration.

The Orchestrator Studio saves the orchestration as a "Personal" UDO. Next, add inputs to the orchestration as described in [Adding Inputs to an Orchestration](#).

You can also use the Save As button and rename an existing orchestration to create a new one.

Caution: If you use Save As to create a copy of an orchestration, only the orchestration is copied. The Orchestrator Studio does NOT create a copy of the components that are associated with the orchestration's steps. That is, both the original orchestration and the new orchestration use the same components that comprise the orchestration. Therefore, in the new orchestration, do NOT modify the components in any way that would break other orchestrations that use the same components.

4.8.3 Adding Inputs to an Orchestration

Orchestration inputs identify the data an orchestration consumes from a calling custom program, a third-party application, an IoT device, or Cloud service. For example, you could have an orchestration consuming data from an IoT device, with a "SensorID" input to pass a sensor ID value to an orchestration and a "TemperatureReading" input to pass a temperature value to an orchestration.

For each input, you specify an input type, which can be a string, numeric, or various date formats.

In the Orchestration design page, there are four different types of inputs you can add to an orchestration:

- ☒ **Orchestration Inputs.**

You can manually enter orchestration inputs, including arrays, to identify the data an orchestration consumes from third-party applications or devices.

After adding a component as a step in the orchestration, you can use the **Add Inputs to Orchestration** button at the end of the step to automatically add inputs defined in a component as inputs to the orchestration. You can remove any inputs as needed, and then map the orchestration inputs to the appropriate inputs in the orchestration steps.

- ☒ **Value from Steps.**

When you add a form request, data request, or cross reference configured to return values from EnterpriseOne, the variables defined to represent the return values appear as additional orchestration inputs. This gives you the option to map data returned from one component to a subsequent component in an orchestration.

Also, if you add a service request step that generates output, the Orchestrator Studio automatically adds a <servicerequestname>.output as an orchestration input. This is a variable that represents the JSON output of the service request, which gives you the option to map the JSON to a subsequent orchestration step.

- ☒ **System Values.**

New orchestrations include the following default inputs: User Address Book Number, User Name, and System Date. These inputs represent the originator of the orchestration when executed at runtime. This gives you the option to map these inputs to an orchestration step.

- ☒ **Variables. (Orchestrator Studio 7.1.0.0)**

You can enter variables that are available as inputs to other steps or that can be used to return data. In order to change the variable value in a step, you must define it in the output of the step. When the step runs, the variable will change and will be available in future steps. This gives you the ability to declare variables at the beginning of an orchestration without having to specify them as orchestration inputs. As the orchestration progresses

through its steps, it can use and redefine the values of these variables, and these values are available to subsequent steps in the orchestration.

Each component that you add to an orchestration—a service request, rule, white list, or cross reference—also includes one or more inputs. For example, inputs in a rule evaluate data received from orchestration inputs to determine the next step in an orchestration. When you assemble an orchestration, you map the orchestration inputs to the inputs defined in the components. See [Mapping Orchestration Inputs](#).

To manually add orchestration inputs in the Orchestration Inputs grid:

1. In the first row in the inputs area, enter the name of the input in the Input column.
2. In the Value Type column, select the input value type. Valid values are:
 - String
 - Numeric
 - Array

Note: Array is not a valid value type if you are adding a variable input.
(Release 9.2.3.2)

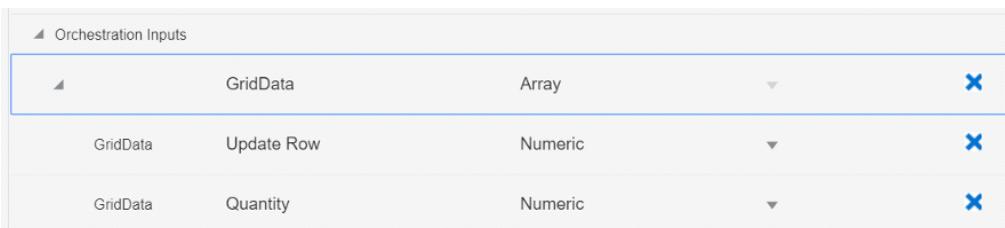
If the input is a date, you can use any of the following date formats:

- dd/MM/yyyy
- dd/MM/yy
- yyyy/MM/dd
- MM/dd/yyyy
- MM/dd/yy
- yy/MM/dd

You can also use the following date formats, which create additional inputs derived from the passed value:

- Milliseconds
 - yyyy-MM-dd 'T' HH:mm:ss.SSSZ
3. Optionally, use the Default Value column to enter a default value that you want to pass through the orchestration input.
 4. If you add an array for an input, the grid expands so you can define the inputs in the array.

As shown in the following example, a user added an array named GridData to the highlighted row. And then entered the inputs "Update Row" and "Quantity" for the array in the subsequent rows.



Orchestration Inputs			
	Input	Type	
	GridData	Array	X
	GridData	Update Row	X
	GridData	Quantity	X

5. Click **Save** to save your changes.

To generate orchestration inputs from an orchestration component:

1. In the row with the orchestration step, click the **Add Inputs to Orchestration** button. The inputs defined in the component appear as inputs to the orchestration.
2. Use the **X** button at the end of each row to remove any unnecessary inputs.
3. Modify the **Value Type** and **Default Value** columns as needed.
4. Click **Save**, and then see [Mapping Orchestration Inputs](#) to map these inputs to the appropriate component inputs.

4.8.4 Adding Steps to an Orchestration

Each component (service request, rule, cross reference, white list) that you add to an orchestration is an orchestration step. It is recommended that you create the component that you want to add to an orchestration before you add it as an orchestration step.

[Figure 4–8](#) shows an orchestration that contains multiple rule, cross reference, and service request steps:

Figure 4–8 Steps in the AddConditionBased Alert Sample Orchestration in the Orchestrator Studio

Type	Action	Iterate Over	Name
Cross Reference			JDE_XREF_Sample_SensorLocation Personal XRE_1611280001CUST
Cross Reference			JDE_XREF_Sample_AlertNotificationRecipients Personal XRE_1611280002CUST
Rule			JDE_RULE_Sample_CBMAlar_1 Personal RUL_1611280002CUST
Form Request	True		JDE_SREQ_Sample_AddCBALert_Alarm Personal SRE_1611280004CUST
Rule	False		JDE_RULE_Sample_CBMAlar_2 Personal RUL_1611280003CUST
Form Request	True		JDE_SREQ_Sample_AddCBALert_Alarm Personal SRE_1611280004CUST
Rule	False		JDE_RULE_Sample_CBMAlar_3 Personal RUL_1611280001CUST

Each row in the Orchestration Steps grid represents a step in the orchestration. The grid displays the following information about each step:

- ❑ **Type.** Displays the component type: Rule, Service Request, Cross Reference, or White List.
- ❑ **Action.** Use this column to define which subsequent step is invoked based on the criteria in the preceding rule. See [Defining the Actions Between a Rule and Dependent Components](#) for details.
- ❑ **Iterate Over.** Use this column to pass a data set retrieved from a form request or data request to a subsequent orchestration step. See [Passing a Data Set to a Subsequent Orchestration Step](#) for more information.
- ❑ **Name.** Displays the name of the component.

Use the "Insert Step Before" and "Insert Step After" buttons to add a step before or after another step in the orchestration. For example, if you determine that you need to add a white list to an existing orchestration, you can insert the white list as an initial step before the other steps.

4.8.4.1 Adding the Initial Step to an Orchestration

1. Click the **Add Step** button (+ symbol).
2. In the "Enter Type of Step" pop-up field, select one of the following steps to add to the orchestration, and then click **Ok**.

- Cross Reference**
- Rule**
- Service Request**
- White List**

The Orchestrator Studio displays the first step in the grid.

3. Define the component for the step:
 - a. To use an existing component for the new step, click the drop-down menu in the Name column and select a component.
 - b. To create a new component for the step, click the **Edit** (pencil) icon at the end of the row to access the design page for creating the new component.

After creating the component, when you return to the Orchestration design page, you can select the component from the drop-down list in the orchestration steps grid to add it.

See the appropriate topics in this chapter on how to create each type of orchestration component.

4. In the Transformations area on the right side of the page, map the orchestration inputs to inputs in the orchestration steps. See [Mapping Orchestration Inputs](#).

4.8.4.2 Adding Additional Steps to an Orchestration

1. Select a step in the grid, and then click either the **Insert Step Before** or **Insert Step After** button to add an additional step before or after the selected step.
2. In the "Enter Type of Step" pop-up box, select the step that you want to add.
3. Click the **Ok** button.

4.8.4.3 Removing a Step from an Orchestration

Caution: Before you delete a step, make sure the step is not used by any other component in the orchestration.

1. Select the step that you want to remove.
2. Above the grid with the steps, click the **Remove Step** button (the X button).

If you make a mistake when updating an orchestration, click the **Restore All** button in the upper-right corner of the design page to restore the orchestration to its last saved version, which erases any changes made since the last save.

4.8.4.4 Defining the Actions Between a Rule and Dependent Components

The Orchestration design page contains an Action column for defining the actions between a rule step and other orchestration steps in an orchestration. After you create a rule with conditions and add it as a step to an orchestration, you need to define the action the orchestration takes if the condition in the rule is met. For example, if a rule contains a condition that when met should invoke a service request step, then in the row with the service request step, you set the Action column to **True**.

You can also define a **False** action to invoke a different orchestration step when a condition in the initial rule is NOT met. A **False** action can invoke a different Service Request step or another rule step that contains additional conditions for which the incoming data is evaluated against. Thus, you can design an orchestration with as many rules and service requests as your business requirements necessitate.

[Figure 4–9](#) shows an example of an orchestration with two rule steps and two service request steps, with the following defined actions:

- ❑ For the first service request step, the action is set to **True** to instruct the orchestration to invoke this service request step when the condition in the first rule step is met.
- ❑ The action for the second (nested) rule step is set to **False** to instruct the orchestration to invoke this rule when the condition in the first rule is NOT met.
- ❑ The action for the second service request step is set to **True** to instruct the orchestration to invoke this service request when the condition in the second rule is met.

Figure 4–9 Defining the Orchestration Actions

Type	Action	Iterate Over	Name	
Cross Reference			JDE_XREF_Sample_SensorLocation Personal XRE_1611280001CUST	✓
Cross Reference			JDE_XREF_Sample_AlertNotificationRecipients Personal XRE_1611280002CUST	✓
Rule			JDE_RULE_Sample_CBMAlarm_1 Personal RUL_1611280002CUST	✓
Form Request	True		JDE_SREQ_Sample_AddCBAlert_Alarm Personal SRE_1611280004CUST	✓
Rule	False		JDE_RULE_Sample_CBMAlarm_2 Personal RUL_1611280003CUST	✓
Form Request	True		JDE_SREQ_Sample_AddCBAlert_Alarm Personal SRE_1611280004CUST	✓

To set the Action column to True or False:

1. In the Orchestration design page, in the appropriate Rule or Service Request step row, click in the Action column.
2. Select either **True** or **False** as appropriate.
3. Click the **Save** button.

After completing the orchestration, you can use the Orchestrator Client to test the orchestration in a test environment. You can access the Orchestrator Client from the drop-down menu in the upper-right corner of the Orchestrator Studio. See [Chapter 9, "Testing Orchestrations in the EnterpriseOne Orchestrator Client"](#) for more information.

4.8.4.5 Defining Error Handling for Orchestration Steps (Orchestrator Studio 7.0.0.0)

In the Orchestrations design page, you can set up error handling for individual orchestration steps. In the orchestration step, you can determine how the Orchestrator handles an error, whether it:

- ❑ Continues processing the orchestration after the error occurs, effectively ignoring the error.
- ❑ Cancels the orchestration.
- ❑ Cancels the orchestration and invokes an alternate orchestration or notification. For example, you might call a notification to alert subscribers if an orchestration is canceled. Or you might run an orchestration to clean up work already performed by the orchestration before it was canceled.

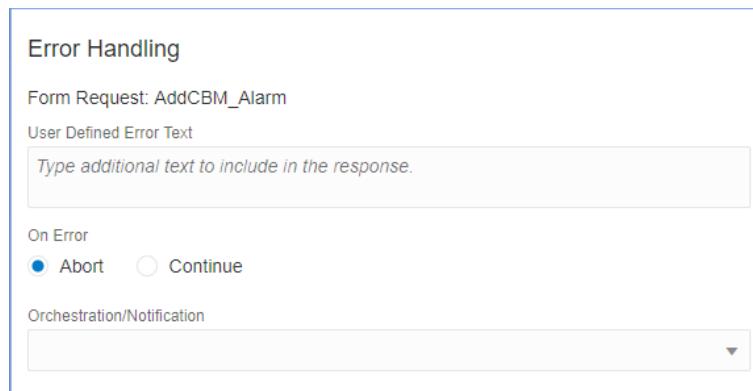
You have the option to include additional text in the error, and you can include a variable to pass a value into the error text.

Regardless of the option you choose, the system still generates an exception when it encounters an error with an orchestration step. For more information on how to monitor exceptions, see [Chapter 6, "Orchestrator Health and Exception Monitoring \(Release 9.2.3\)".](#)

To define error handling for an orchestration step:

1. At the end of the step in the grid, click the **Error Handling** icon.

The Error Handling dialog box displays with the type and name of the step, as shown here:



2. In the User Defined Error Text field, you can enter additional details about the error to display in the exception details. You can insert variables in the error text using the \${ } notation, where you enter the variable name between the brackets.

If you configure the step to continue on error, this error text will appear in the response in a new array called "Continued on Error." If you configure the step to abort on error, this text will appear in the returned exception as userDefinedErrorText. The error text is also added to the AIS Server log.

3. For On Error, select one of the following options:

- ❑ Abort
- ❑ Continue

4. (Optional) If you choose to Abort on error, you can configure the orchestration step to invoke another orchestration or notification:

- a. Click the **Orchestration/Notification** drop-down list and select an orchestration or notification to invoke after an error.
- b. Map the inputs as appropriate.

Note: In the exception, the response of the alternate orchestration or notification is displayed under the Exception Process Response label.

4.8.5 Mapping Orchestration Inputs

On the Orchestration design page, use the Transformations area to map orchestration inputs to inputs defined in an orchestration step, such as inputs in a rule, cross reference, white list, or service request component. The Transformations area includes an Auto Map button for automatically mapping any matching inputs in the orchestration and orchestration step.

If an orchestration and an orchestration component use different input names to identify the same data, you can use the grid in the Transformations area to map the inputs. This facilitates the reuse of components, so you can avoid having to create a new component simply to match the input names (from a third-party application or device) in a new orchestration.

To better understand how to map inputs, see the example depicted in [Figure 4–10](#). In this example, two orchestration inputs have the same name as the inputs in the service request. These inputs were automatically mapped using the Transformations Auto Map button. The orchestration also contains a SerialNumber input to identify equipment, which is represented in EnterpriseOne as "EquipmentNumber." To map these inputs in the Transformations area, "SerialNumber" was selected in the Available Values drop-down list next to the EquipmentNumber service request input.

Figure 4–10 Transformations Example

Service Request Input	Available Values
Equipment Number	SerialNumber
Latitude	Latitude
Longitude	Longitude

Initially, when you have a small number of orchestrations in your system, it is recommended to keep all input names consistent among the orchestration and all components (orchestration steps) used by orchestration. But as your library of orchestrations and orchestration components increases, instead of creating new components with input names that match the orchestration inputs, you can use transformations to map the orchestration inputs to an existing component.

To map inputs:

1. In the Orchestration design page, select an orchestration step to which you want to map orchestration inputs.
2. Click the Transformations **Auto Map** button to map matching inputs. The Studio automatically maps inputs with the same name; the matching orchestration inputs appear in

the Available Values column next to the matching input in the "Orchestration Step Input" column.

3. To map an orchestration input to an orchestration step input that does not have the same name:
 - a. In the Transformations table, click the drop-down menu in the Orchestration Input column next to the orchestration step input.
 - b. Select the orchestration input to map to the orchestration step input.
- For example, in [Figure 4–10](#), SerialNumber is mapped to EquipmentNumber in the Service Request Input.
4. You can map a literal value to the orchestration step input by entering a value in the Default Value column.
 5. (Orchestrator 6.1.0) If you are mapping values from an array, you can configure the orchestration to iterate over the orchestration step so the step is called once for each row in the array:
 - a. After defining the array in the Orchestration Inputs section, click the **Iterate Over** drop-down list in the Transformations area and select the name of the array input.
 - b. Make sure to map the orchestration inputs for the array to the inputs defined in the orchestration step, as shown in the following example:

Type	Action	Iterate Over	Name
Service Request	GridData	Update SO Personal	SRE_1711270008CUST

Input	Value Type	Default Value
GridData	Array	
GridData	Numeric	
GridData	Numeric	

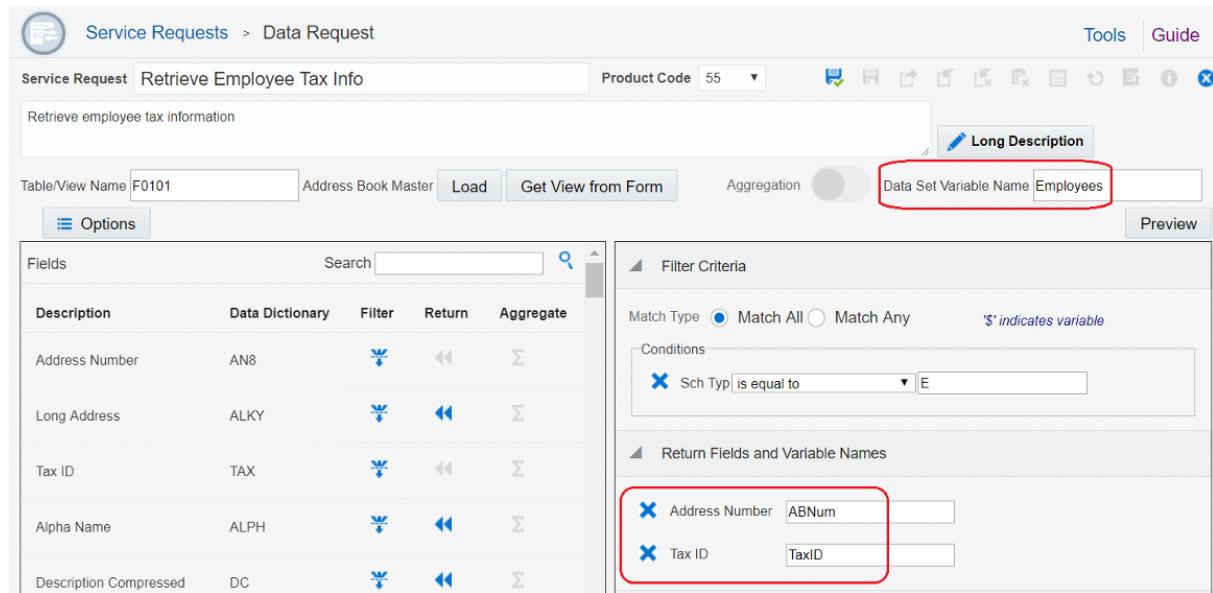
Transformations		Iterate Over	Available Values
Service Request Input	Row Numb	GridData	Row Numb
Quantity	P4210_Version	Quantity	Quantity

Notice that in the Orchestration Steps grid, the name of the array was automatically inserted into the Iterate Over column in the Service Request step.

4.8.6 Retrieving and Passing Data Sets in an Orchestration

You can configure a form request or a data request to return a data set. You can also configure an orchestration to pass a data set returned from a form request or a data request to another step in the orchestration. At runtime, the orchestration executes the form request or data request once per row in the data set.

[Figure 4–11](#) shows an example of a data request configured with a data set named "Employees" that will retrieve the address book number and tax ID of all employee records in EnterpriseOne.

Figure 4–11 Data Request Configured to Retrieve a Data Set

4.8.6.1 Configuring a Data Request to Retrieve a Data Set

1. On the Data Request page, enter a name for the data set in the **Data Set Variable Name** field.
2. Define filtering criteria for the data set.
3. In the grid, click the "Return" icon next to the fields that you want the data request to return in a data set.
4. In the "Return Fields and Variable Names" area, enter a variable name for each return field.

When you add this data request to an orchestration, you can use these variables to map the data in the data set to a subsequent orchestration step.

For more information about configuring a data request, see [Configuring a Data Request](#).

4.8.6.2 Configuring a Form Request to Retrieve a Data Set

Note: Data sets from grids on a power form are not supported.

1. In the form request, locate the row with the *Form Name-Grid* node.
2. In the *Form Name-Grid* row, enter a name for the data set in the Variable Name column. If you configure the orchestration to pass the data in the data set to a subsequent orchestration step, you enter this variable name in the Iterate Over column of the orchestration step to which you want to pass the data.
3. For each column that you want to include in the data set, click **Return**.
4. For each column that you specified to include in the data set, enter a variable name so that the returned column can be mapped to a subsequent orchestration step.

For more information about configuring a form request, see [Configuring a Form Request in the Orchestrator Studio](#).

4.8.6.3 Passing a Data Set to a Subsequent Orchestration Step

You can configure an orchestration to pass each row of a data set to an orchestration step.

1. Add the form request or data request defined with a data set to the orchestration.
2. In the **Iterate Over** column of the orchestration step to which you want to pass the data set, enter the data set name defined in the form request or data request.
3. Map the data set fields to the inputs in the orchestration step. See [Mapping Orchestration Inputs](#).
4. Click **Save**.

4.8.7 Working with Orchestration Output

Use the Orchestration Output page to view the JSON representation of orchestration output.

Viewing the output enables you to validate the fields that contain the data you want returned in the orchestration output for output mapping.

Orchestration output can include values from fields and grid columns that you specify as returns when setting up a form request, data request, or cross reference in an orchestration. It can also include the response from a connector, custom service request, or the result of a rule (true or false).

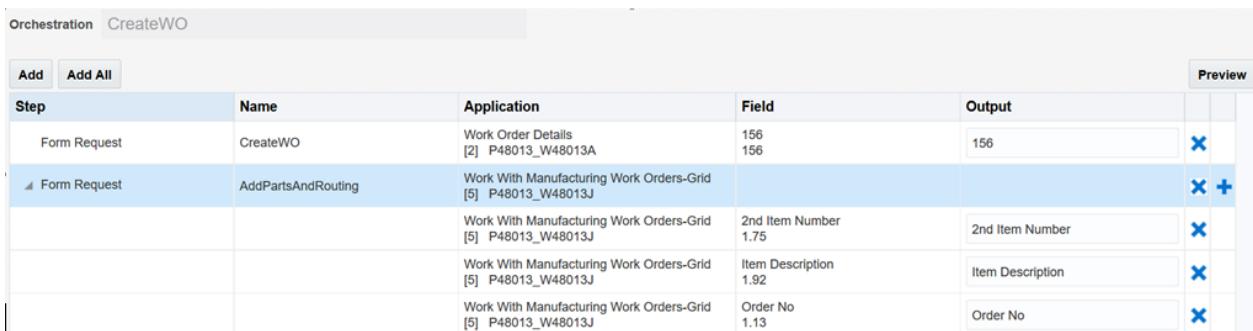
On the Orchestration Output page, you can use Groovy scripting to manipulate the contents of the response to refine the orchestration output as required by the parameters in the consuming device or program.

Note: Orchestration output is available only with version 2 orchestrations created in Orchestrator Studio 5.0.1 or higher. For orchestrations created prior to Orchestrator Studio 5.0.1, you must update them to version 2 orchestrations to view orchestration output.

To work with orchestration output:

1. On the Orchestration design page, click the **Orchestration Output** button.
2. If you want to view the JSON code of all return fields in an orchestration, click **Add All**.

The grid displays all of the components or steps in the orchestration that contain return fields, as shown in the following example:



The screenshot shows the 'Orchestration Output' grid for the 'CreateWO' orchestration. The grid has columns for Step, Name, Application, Field, and Output. The 'Add' and 'Add All' buttons are at the top left. The 'Preview' button is at the top right. The grid data is as follows:

Step	Name	Application	Field	Output
Form Request	CreateWO	Work Order Details [2] P48013_W48013A	156 156	156
Form Request	AddPartsAndRouting	Work With Manufacturing Work Orders-Grid [5] P48013_W48013J		
		Work With Manufacturing Work Orders-Grid [5] P48013_W48013J	2nd Item Number 1.75	2nd Item Number
		Work With Manufacturing Work Orders-Grid [5] P48013_W48013J	Item Description 1.92	Item Description
		Work With Manufacturing Work Orders-Grid [5] P48013_W48013J	Order No 1.13	Order No

3. Instead of Add All, you can individually select the return fields for which you want to preview the JSON code:
 - a. Click **Add** to view a list of all steps (orchestration components) in the orchestration that contain return fields.

You can expand each step to see the return fields. The Output Field column displays the variable name defined for a return field.

- b. In the Select column, click the check box next to any return field that you want to add to your orchestration output, and then click **OK**.

The field will be returned in JSON format, which you can also preview on the design page.

4. To modify the return fields for which you want to see the JSON representation:

- ❑ You can change the variable name for the return field in the Output column.
- ❑ Click **Delete (X)** at the end of row to remove a return field from the JSON preview.
- ❑ Click **Add (+)** to add any additional grid controls not currently selected for output.

5. Click the **Preview** button.

The following image shows an example of the JSON code for three grid return fields: 2nd Item Number, Item Description, and Order Number.

Add	Add All			
Step	Name	Application	Field	Output
Form Request	AddPartsAndRouting	Work With Manufacturing Work Orders [5] P48013_W48013J		
		Work With Manufacturing Work Orders [5] P48013_W48013J	2nd Item Number 1.75	2nd Item Number
		Work With Manufacturing Work Orders [5] P48013_W48013J	Item Description 1.92	Item Description
		Work With Manufacturing Work Orders [5] P48013_W48013J	Order No 1.13	Order No

6. In the Preview area, you can click the **Copy to Clipboard** button so you can paste the JSON code into your program.

7. (Advanced) In Orchestrator Studio, edit the provided Groovy script template in the Manipulate Output area to manipulate the contents of the response to refine the orchestration output.

See [Groovy Template for Manipulating Output from an Orchestration Response](#) for more information.

8. Click the **Test** button to view the output.

The Orchestrator Studio displays a preview of the output manipulated by the Groovy script.

9. To save the orchestration outputs, return to the Orchestration design page and click **Save**.

4.9 Creating Schedules for Orchestrations

This section contains the following topics:

- ❑ [Understanding Schedules](#)
- ❑ [Creating a Schedule](#)
- ❑ [Adding a Schedule to an Orchestration](#)

4.9.1 Understanding Schedules

The Orchestrator Studio gives you the option to create and assign a schedule to an orchestration. A schedule determines how often the Orchestrator executes an orchestration. For

example, with a schedule, you can create an orchestration that regularly checks for a Watchlist threshold, that when exceeded, sends a notification to the appropriate users.

You define a schedule using minutes, hours, days, or a Cron string. Cron is a scheduling utility using cryptic strings to define an interval. Then you associate a schedule to an orchestration to determine how often it runs. You can attach the same schedule to multiple orchestrations.

You must have been granted proper UDO security to create schedules. Schedules are managed as UDOs, which means you need the proper UDO security in order to publish and share your schedules for others to use, or use schedules that others have published.

The scheduler is a process on the Application Interface Services (AIS) server that starts, stops, and manages schedules. An administrator uses the AIS Server REST API to start, stop, and manage schedules through the scheduler.

4.9.2 Creating a Schedule

Create a schedule to define how often the Orchestrator should execute an orchestration. You can define a schedule using minutes, hours, days, or a Cron string. Cron is a time-based job scheduler that can be used to schedule jobs to run periodically at fixed times, dates, or intervals (for example, every Tuesday and Friday at 9:00 am).

To create a schedule:

1. On the Orchestrator Studio Home page, click the **Tools** link in the upper-right corner.
2. On the Tools page, click the **Schedules** icon.
The Orchestrator Studio displays the Schedules design page.
3. Click the **New Schedule** button.
4. In the Schedules field, enter a name for the schedule. Do **NOT** include special characters in the name.
5. Click the **Product Code** drop-down list to select a product code to associate with the schedule.

This gives an administrator the option to manage UDO security for orchestration components by product code.

6. In the space provided, enter a short description with a maximum of 200 characters. This description should clearly describe the frequency of the schedule so that it can be attached to notification as needed.
7. Click the **Edit Long Description** button to add a long description to provide more detail about the purpose of the component.
8. Do one of the following:
 - In the **Schedule to Run** section, select a number of minutes, hours, or days to define how often you want the schedule to run.
If you select minutes, you cannot run more often than every five minutes.
 - In the **Or Enter a Cron String** section, enter a Cron string to define the schedule.
Cron is a time-based job scheduler that can be used to schedule jobs to run periodically at fixed times, dates, or intervals. There are many third-party Cron expression generators available that can help you create a Cron string.
9. Click the **Save** or **Save As** icon in the upper-right corner.

The first time a new schedule is saved, it is saved as a "Personal" UDO. Thereafter, you can use the UDO buttons described in the User Defined Object (UDO) Features section to move the schedule to the appropriate status.

4.9.3 Adding a Schedule to an Orchestration

Adding a schedule to an orchestration does not invoke the orchestration as scheduled. Starting the scheduler is a separate step. After you add a schedule, ask an administrator to start and administer the schedule using REST API services.

For more information about REST API services for starting, stopping, and managing the Scheduler, see *JD Edwards EnterpriseOne Tools REST API for the Application Interface Services Server*.

Note: In order to use the Scheduler with an orchestration, it must be a version 2 orchestration. If you want to update a version 1 orchestration created in a previous version of the Orchestrator Studio, see [Updating Version 1 Orchestras to Version 2 Orchestras](#).

To add a schedule to an orchestration:

1. Select the orchestration to which you want to add a schedule and open it in the Orchestration design page.
2. On the Orchestration design page, click the **Schedule** drop-down list and select a schedule.
3. Click **Save** to save your changes.

4.10 Updating Version 1 Orchestras to Version 2 Orchestras

When executed by the Orchestrator on the AIS Server, orchestrations programmatically call AIS services on the AIS Server to perform specific actions in EnterpriseOne.

Starting with EnterpriseOne Tools 9.2.1.2 is the availability of version 2 AIS services. Version 2 AIS services include all services originally available on the AIS Server (referred to as version 1 services), plus additional services that enable you to create an orchestration that includes the following types of service requests:

- Data request
- Message
- Connector

All orchestrations that you create with Orchestrator Studio 5.0.1 or higher use version 2 AIS services and are referred to as version 2 orchestrations.

All orchestrations created prior to Orchestrator Studio 5.0.1 use version 1 AIS services and are referred to as version 1 orchestrations.

In Orchestrator Studio 5.0.1 or higher, you can update version 1 orchestrations. However, if you add any of the components in the preceding list that rely on version 2 AIS services, you must save the orchestration as version 2 by selecting **Version 2** from the Orchestrator Version drop-down list.

Caution:

If you save a version 1 orchestration as version 2, the following parameters are used for the service request by default:

- ❑ Output Type = grid data
- ❑ Turbo Mode = low

This changes the format of the response, which will affect any device consuming the output returned from the orchestration.

4.11 Reloading Orchestras and Orchestration Components

Reload Files enables you to reload orchestrations and orchestration components to the state in which they were last saved in the Orchestrator Studio. Use this feature if you do not want to save any modifications that you made.

To reload all files, click the drop-down menu in the upper-right corner of the Orchestrator Studio and then click the **Reload Files** link.

Each individual orchestration component panel also contains a Restore button that you can use to restore an individual component.

4.12 Supported Input Message Formats

The Orchestrator supports three input message formats for orchestrations: a standard JD Edwards EnterpriseOne format, a generic format, and Oracle Cloud IoT format. [Example 4–1](#) and [Example 4–2](#) show an example of the code for each format.

Example 4–1 Standard JD Edwards EnterpriseOne Input Message Format

```
{
  "inputs": [
    {
      "name": "equipmentNumber",
      "value": "41419"
    },
    {
      "name": "description",
      "value": "test"
    },
    {
      "name": "date",
      "value": "1427774400000"
    },
    {
      "name": "time",
      "value": "12:15:15"
    },
    {
      "name": "temperature",
      "value": "99"
    }
  ]
}
```

Example 4–2 Generic or Oracle Cloud IoT Input Message Format

```
{  
    "equipmentNumber": "41419",  
    "description": "test",  
    "date": "1427774400000",  
    "time": "12:15:15",  
    "temperature": "99"  
}
```

4.12.1 Additional Supported Input Message Formats for the Generic Input Format

Additional formats are supported when using the generic input format, as long as the orchestration input values are defined using the full path to the elements used. You may have a deeper JSON structure like this.

```
{  
    "equipmentNumber": "41419",  
    "equipementDetail": {  
        "type": "thermometer",  
        "readingDetail": {  
            "temperature": 200,  
            "units": "F"  
        }  
    }  
}
```

To reference the temperature within the orchestration, you can use the full path delimited by periods, for example:

```
equipmentDetail.readingDetail.temperature
```

4.12.2 Differences Between Input Message Formats

The following list describes the differences between the input message formats:

- The iterateOver attribute for the orchestrationStep element is supported only by the standard JD Edwards EnterpriseOne input message format.
- When using the "detail" form action type with the standard format, it will automatically iterate over all detailInputs and repeatingInputs in order to add multiple rows to a grid. If the generic format is used, only a single grid row can be added.

As shown in [Example 4–3](#), "detailInputs" would correspond to grid data; "repeatingInputs" would correspond to individual rows that contain "inputs" that correspond to columns.

If you have a power form with two grids, you could populate both grids using two "detailInputs" structures with different "name" values that correspond to the different grids. "repeatingInputs" would contain a list of rows and for each row you would define a list of "inputs".

You could also define a CrossReference orchestration step with `iterateOver="GridData"` that converts the item value into an EnterpriseOne specific item number. For example, if A123=220 and A124=230, the single orchestration step would convert both.

Example 4–3

```
{  
    "inputs": [
```

```
{
    {
        "name": "BranchPlant",
        "value": "30"
    },
    {
        "name": "customer",
        "value": "4242"
    }
],
"detailInputs": [
    {
        "name": "GridData",
        "repeatingInputs": [
            {
                "inputs": [
                    {
                        "name": "item",
                        "value": "A123"
                    },
                    {
                        "name": "Quantity",
                        "value": "3"
                    }
                ]
            },
            {
                "inputs": [
                    {
                        "name": "item",
                        "value": "A124"
                    }
                ]
            }
        ]
    }
]
}
```

4.13 Orchestration Security Considerations

Before the EnterpriseOne Orchestrator can process an orchestration, authentication of the JD Edwards EnterpriseOne user ID and password must take place. It is the responsibility of the originator of the service request to tell the orchestration about the user. The user's credentials must be supplied in a basic authorization header or in the JSON body of the request. The user must also have authorized access to the EnterpriseOne application in which the resulting transaction takes place. The following code is an example of credentials in the JSON body of the request:

```
{
    "username": "JDE",
    "password": "JDE",
    "environment": "JDV900",
    "role": "*ALL"
}
```

The AIS service used with orchestrations is stateless; each call passes credentials to establish a separate EnterpriseOne session. After the transaction is complete, the session closes.

In addition to passing credentials for authentication, you can employ a second level of security for the Orchestrator through whitelisting. Whitelisting enables an initial rudimentary pass/fail check of the incoming device signature against a predefined list of signatures. If a value passed to the Orchestrator is not a valid value included in the orchestration's white list, the Orchestrator rejects the input. For more information, see [Creating White Lists](#) in this guide.

4.13.1 Restricting Access to Exposed Orchestrations

If you expose orchestrations for business partners or customers to invoke, it is recommended to use an http proxy to allow access to only the endpoints required to execute the orchestration. Configure the proxy to restrict all endpoints except /orchestrator, /discover, /tokenrequest, and /tokenrequest/logout. This allows external users set up with the proper UDO security to discover and call orchestrations.

4.13.2 How to Maintain a Single Session for Multiple Calls to an Orchestration

You can maintain a single AIS Server session for multiple calls to an orchestration by first performing a token request to get a session token. Otherwise, each call to an orchestration on the AIS Server establishes a separate session, which can decrease AIS Server performance.

When performing a token request, a session is established and the AIS Server returns an AIS token in the response. The token is then used for all orchestration calls in the same session.

The amount of time the session remains open is established through the session lifetime settings in the AIS server. The third party client is responsible for ensuring a valid token is available or re-requesting a new token once a previous token has timed out. If a valid token is not available, the client will receive an error in the response stating the token is invalid.

4.14 Exporting Orchestration Components from the Orchestrator Studio

Caution: Do not use the Export and Import tools to share orchestration component files with other users. With orchestration components stored as UDOs in EnterpriseOne, the management of orchestration components, including the sharing and modifying of shared orchestration components and promoting of orchestration components to the appropriate AIS Server directory for test or production purposes, is administered through the life cycle management tools in EnterpriseOne.

You can export any of the orchestration component types from the Orchestrator Studio to view the source XML files of the components. This is only recommended for advanced users for troubleshooting purposes or for making manual customizations. The Orchestrator Studio exports the source XML files in a zip file.

To export an *orchestration* component, the Orchestrator Studio gives you the option to export only the selected orchestration component or to export the orchestration and all components associated with it. For example, if an orchestration has a service request component and a rule component associated with it, if you export all components, the zip file will contain XML files for the orchestration, service request, and rule.

To export orchestration components:

1. On a component design page, select the component to export and then click the **Export File** button in the upper-right corner.

If you are exporting an orchestration, a dialog box appears in which you can click **All** or **Orchestration Only**.

2. Follow the instructions in the browser to save the zip file to a location on your local machine.

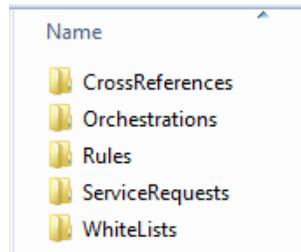
4.15 Importing Orchestration Files in the Orchestrator Studio

Caution: Do not use the Export and Import tools to share orchestration component files with other users. With orchestration components stored as UDOs in EnterpriseOne, the sharing and modifying of shared orchestration components, as well as the promoting of orchestration components to the appropriate AIS Server directory for test or production purposes, is administered through the life cycle management tools in EnterpriseOne.

You might need to import orchestration component XML files that were exported from the Orchestrator Studio for advanced troubleshooting or customization purposes. The Orchestrator Studio Import tool enables you to import individual orchestration XML files or a zip file containing XML files.

Caution: You cannot import orchestration component files that were exported from the EnterpriseOne Object Management Workbench - Web application. The zip file created from an OMW - Web export stores the XML files in a structure that cannot be read by the Orchestrator Studio Import tool.

If importing a zip file that contains orchestration XML files and dependent orchestration component XML files, the zip should contain the following folders with each of the XML files stored in the appropriate folder:



If you import an XML file that has the same name as a current orchestration component (UDO) in the Orchestrator Studio, you have the following options:

- If the UDO is at a status of "Personal" or "Reserved," you can overwrite the UDO with the XML file.
- If the UDO is in a "Shared" status, you cannot overwrite it. But you are given the option to import it as a new UDO with a status of "Personal."

To import files:

1. On the Orchestrator Studio Home page, click the **Tools** link in the upper-right corner.
2. On the Tools page, click the **Import Files** icon.
3. On Import Files, click the **Choose File** button.

4. Locate the orchestration component XML files or zip file that contains the orchestration component files that you want to import.

After selecting the files to import, the Import tool checks the XML files that you selected against the current UDOs in the Orchestrator Studio and displays the options that are available for importing the files, as shown in the example in [Figure 4–12](#).

Figure 4–12 Orchestrator Studio Import Tool

Select File for Import		
Orchestrations-1461866371912.zip	Update...	Submit
Number of Files 9		
Number of Files Existing on Server 2		
Select All		
Cross Reference		
JDE_XREF_Sample_SensorLocation.xml	No update allowed at current status	<input type="checkbox"/>
JDE_XREF_Sample_AlertNotificationRecipients.xml	Select to add or else reserve record and re-import to update	<input type="checkbox"/>
Service Requests		
JDE_SREQ_Sample_AddCBAAlert_Warning.xml	Overwrite existing file	<input checked="" type="checkbox"/>
JDE_SREQ_Sample_AddCBAAlert_Alarm.xml	File is ready to submit	<input checked="" type="checkbox"/>
Rules		
JDE_RULE_Sample_CBMAAlarm_3.xml	File is ready to submit	<input checked="" type="checkbox"/>
JDE_RULE_Sample_CBMAAlarm_1.xml	File is ready to submit	<input checked="" type="checkbox"/>
JDE_RULE_Sample_CBMAAlarm_2.xml	File is ready to submit	<input checked="" type="checkbox"/>
JDE_RULE_Sample_CBMWarning.xml	File is ready to submit	<input checked="" type="checkbox"/>
Orchestrations		
JDE_ORCH_Sample_AddConditionBasedAlert_Generic.xml	File is ready to submit	<input checked="" type="checkbox"/>
Submit		

5. Review the import option for each file to determine how to proceed with the import. The options are:

- ❑ No update allowed at current status.

The XML file name is the same as an existing component, which has a status of "Pending Approval," so it cannot be overwritten.

- ❑ Select to add or else reserve record and re-import to update.

A component with the same name exists in the Orchestrator Studio in a "Shared" status. Click the check box if you want to import the file as a new UDO. A new name will be generated for the new component upon import.

If you want to overwrite the current component in the Orchestrator Studio, clear the check box. And then change the status of the current component to "Reserved" and reimport the XML file to overwrite the component.

- ❑ File already exists.

The XML file matches a component that is in a "Personal" or "Reserved" status. Click the check box to import it and overwrite the current component.

- ¤ File is ready to submit.

The XML file is unique and does not already exist in the Orchestrator Studio. Click the check box to import it.

6. You can also click the **Select All** check box to import all XML files that are available for importing.

7. Click the **Submit** button.

The Orchestrator Studio imports the selected components into the components list on the respective design page.

Setting Up Cross References and White Lists in EnterpriseOne (P952000)

Use the EnterpriseOne Business Service Cross Reference (P952000) application to create and manage cross reference and white list records for EnterpriseOne orchestrations. Orchestration cross references and white lists contain key-value data pairs used by the Orchestrator. You must add records for all orchestration key-value data pairs in P952000.

For example, a device might provide a machine ID that equates to the Equipment Number field in EnterpriseOne. After including this in the orchestration cross reference, you also have to add a record for this cross reference to P952000 in order for the orchestration to invoke and perform the intended transaction in EnterpriseOne.

For instructions on how to add cross reference and white list records in EnterpriseOne, see the "Setting Up Orchestration Cross References" in the *JD Edwards EnterpriseOne Tools Interoperability Guide*. Also, remember the following details when creating records for cross references and white lists:

- In P952000, you must first create one or more cross reference object types for grouping or categorizing cross reference and white list records. You may have thousands of records in P952000. You use cross reference object types to group these records into manageable categories. You define the cross reference object types as needed. See "Adding Cross-Reference Object Types" in the *JD Edwards EnterpriseOne Tools Interoperability Guide*.
- When creating a cross reference or white list record, you must select the **AIS** option for the Cross Reference Type, which specifies that these records are for use with orchestrations.

Cross Reference Type	Cross Reference Object Type	Third Party App ID	Third Party Value	EOne Value		
AIS	ABMASTER	ABFilter	25	<25		
AIS	ABMASTER	ABFilter	50	<50		
AIS	ABMASTER	WHITELIST	OpenScript	C	OpenScript	C

 The 'Cross Reference Type' column has radio buttons for AIS, CODE, and All. The 'Cross Reference Object Type' column contains ABMASTER. The 'Third Party App ID' column contains ABFilter, ABFilter, and WHITELIST respectively. The 'Third Party Value' column contains 25, 50, and the third row's value is not visible. The 'EOne Value' column contains <25, <50, and the third row's value is not visible."/>

- When you add a white list record, you must enter WHITELIST for the Third Party App ID. This automatically changes the EOneValue column value to NA because the EOneValue column is not applicable to a white list.

Add Business Service Cross Reference

Records 1 - 2				
Cross Reference Type	Cross Reference Object Type	Third Party App ID	Third Party Value	EOne Value
AIS	ADDRESS	WHITELIST	1234	NA

- When setting up cross references, you can enter multiple key cross references by delimiting the values with a pipe (|). These will be consumed based on the cross reference definition in the orchestration. You can do the same for white lists.

Add Business Service Cross Reference

Records 1 - 2				
Cross Reference Type	Cross Reference Object Type	Third Party App ID	Third Party Value	EOne Value
AIS	ITEM_BRANCH	TPApp	A-293-B2	200 M30

- To expedite creating cross reference records for Orchestrator in P952000, you can enter all cross reference record information into a spreadsheet, and then use the Import tool in P952000 to import all records at once rather than entering one record at a time. See "Importing Data from an External Spreadsheet to a Grid" in the *JD Edwards EnterpriseOne Tools Foundation Guide* for more information.

Orchestrator Health and Exception Monitoring (Release 9.2.3)

This chapter contains the following topics:

- [Section 6.1, "Understanding the EnterpriseOne Orchestrator Monitor"](#)
- [Section 6.2, "Prerequisites"](#)
- [Section 6.3, "Accessing the Orchestrator Monitor"](#)
- [Section 6.4, "Monitoring UDOs Based on Different EnterpriseOne User, Environment, or Role"](#)
- [Section 6.5, "Refreshing the Data Displayed in the Orchestrator Monitor"](#)
- [Section 6.6, "Resetting the Data Displayed in the Orchestrator Monitor"](#)
- [Section 6.7, "Monitoring Orchestrator Health"](#)
- [Section 6.8, "Monitoring Orchestrator Exceptions"](#)
- [Section 6.9, "Managing Orchestrator Health and Exception Records in EnterpriseOne"](#)

6.1 Understanding the EnterpriseOne Orchestrator Monitor

The EnterpriseOne Orchestrator Monitor (P980060X) is an EnterpriseOne application that enables you to perform a health check of your EnterpriseOne Orchestrator environment. It provides information about which objects are performing well and which ones might need fine tuning, as well as details about exceptions so that you can take corrective actions to resolve any issues.

With the Orchestrator Monitor, you can monitor the following user defined objects (UDO) processed by the EnterpriseOne Orchestrator:

- [Orchestrations](#)
- [Notifications](#)
- [Schedules](#)

For schedules, the Orchestrator Monitor displays only exception information; it does not provide health details.

Orchestrator Health Pane in My Worklist

Oracle provides a downloadable My Worklist composed page, which includes a Watchlist pane, Message Center, and the Orchestrator Health pane. The Orchestrator Health pane in My Worklist provides another way to monitor the health of your orchestrations and notifications without having to access the Orchestrator Monitor. However, it does not provide exception

information. For information on how to download and access My Worklist, see "My Worklist" in the *JD Edwards EnterpriseOne Tools Foundation Guide*.

6.2 Prerequisites

Before users can monitor UDOs, an administrator must install the Orchestrator Monitor and enable Orchestrator health and exception tracking in Server Manager. See [Setting Up Orchestrator Health and Exception Monitoring \(Release 9.2.3\)](#).

6.3 Accessing the Orchestrator Monitor

You can access the Orchestrator Monitor from EnterpriseOne or from the Orchestrator Studio 7.0.x.0.

In EnterpriseOne, click **Navigator**, **EnterpriseOne Menus**, **EnterpriseOne Life Cycle Tools**, **Orchestrator Management**, **Orchestrator Monitor** (P980060X).

From the Orchestrator Studio, click the **Tools** link, and then click the **Orchestration Monitor** icon. Use your EnterpriseOne credentials to sign in.

If the Orchestrator Monitor does not display health or exception information, contact your system administrator and request UDO view security access to the orchestrations, notifications, and schedules you want to monitor.

6.4 Monitoring UDOs Based on Different EnterpriseOne User, Environment, or Role

The Orchestrator Monitor displays health and exception details of the UDOs (orchestrations, notifications, and schedules) based on the EnterpriseOne credentials, environment, and role that you signed in with. If you have UDOs running in a different environment or under a different user name or role, you can change the data displayed in the Orchestrator Monitor based on this criteria. You can also display results based on the product code assigned to UDOs when they were created.

To change the UDOs displayed in the Orchestrator Monitor:

1. Click the triple bar icon to access the Search Criteria panel.
2. Complete any or all fields as necessary to display information about objects deployed under a different user, environment, role, or product code.
3. Click **Search**.

6.5 Refreshing the Data Displayed in the Orchestrator Monitor

Orchestrations and notifications can run intermittently or at scheduled intervals. As a result, health and exception data can change frequently. On the Health tab, you can click **Refresh** or press **F5** to refresh the data. On the Exceptions tab, the Orchestrator Monitor refreshes the data each time you perform any of the following actions:

- ❑ Press **F5**.
- ❑ Switch between the Health tab and the Exceptions tab.
- ❑ Switch between the Exception Chart view and the Exceptions List view.
- ❑ After you enter a custom date range and click **Search**.

- Change your search criteria or filter results.
- After you use the additional search field and click **Search**.
- Click or expand an exception record.

6.6 Resetting the Data Displayed in the Orchestrator Monitor

The data displayed in the Orchestrator Monitor is based on records saved to the EnterpriseOne Health (F980061) and Exception (F980060) tables. An administrator can use the Orchestrator Health and Exceptions program (P980060) in EnterpriseOne to view or delete historical records that contain data that you no longer want to monitor in your current environment.

By deleting health and exception records, you are essentially resetting the data displayed in the Orchestrator Monitor.

See [Managing Orchestrator Health and Exception Records in EnterpriseOne](#) for information on how to view and delete health and exception records in EnterpriseOne.

6.7 Monitoring Orchestrator Health

The Health tab in the Orchestrator Monitor provides statistics about the health of orchestration and notification UDOs. It lists only the UDOs that you are authorized to monitor through UDO view security.

The top right of the Health tab displays the overall successes and failures of your orchestrations and notifications.

The Health grid displays 10 records at a time. Use the controls at the bottom of the page to access any additional records.

6.7.1 Filtering the UDOs Displayed in the Health Tab

To view one or a refined set of UDOs in the Health tab, click the **Name** field at the top of the grid and select a UDO from the drop-down list. Click it again to add additional UDOs. You can also use the **Show All**, **Failures**, or **Successes** options to change the data displayed based on the option selected.

6.7.2 Understanding Orchestrator Health Data

The default view in the Health tab displays high-level information about the performance of each UDO. Expand each row to find additional performance details.

The data shown is based on records stored in the EnterpriseOne Health (F980061) table. An administrator can reset the data in these tables. See [Resetting the Data Displayed in the Orchestrator Monitor](#) for more information.

High-Level Health Information

Note: In most columns, you can click the column heading to sort data by a particular column.

- Name

This is the name of the UDO as given by the creator of the UDO. The icon next to it indicates if it is an orchestration  or notification .

- Health - Last 10



This bar chart shows up to the last 10 instances that the UDO was executed, with each bar representing a single instance. The instances are listed earliest to latest, from left to right. A red bar indicates a failure. A green bar indicates a success. The height of each bar indicates the length of time taken to process the UDO. Move your cursor over each bar to display the date and time the instance was executed and the time it took to complete.

- Success Rate

The success rate of ALL instances of the UDO, not just the last 10 instances shown in the bar chart.

- Shortest

The shortest time it took an instance to complete, measured in seconds.

- Longest

The longest time it took an instance to complete, measured in seconds.

- Last Success

The last time the instance ran successfully.

- Last Fail

If there was a failure, the date and time of the last failure.

- Runs Per Day

The average number of times the UDO is executed each day.

Detailed Performance Information

Expand a row to view the following additional details:

- UDO Name

This is the ID of the UDO in EnterpriseOne.

- Environment

The environment in which the UDO was executed.

- Product Code

The product code associated with the UDO.

- First Run

The date and time that the first instance of the UDO was run.

- Successes

The total number of successes.

- Failures

The total number of failures.

- Average Success

The average time in seconds to successfully process all instances of the UDO.

- Average Failure

The average time in seconds of instance failures.

- Last Run

The time in seconds for the last instance to complete.

- Health - Last 10

This section provides the same details presented by the hover help in the Health - Last 10 bar chart.

6.8 Monitoring Orchestrator Exceptions

The Exceptions tab in the Orchestrator Monitor displays details about exceptions that occur when the Orchestrator processes orchestration, notification, and schedule UDOs. Use the Chart option to view exceptions graphically in a chart. Use the List option to view exceptions in a list.

You can view exception records only for UDOs to which you have been granted access through UDO view security.

The data shown is based on records stored in the EnterpriseOne Exceptions (F980060) table. An administrator can reset the data in these tables. See [Resetting the Data Displayed in the Orchestrator Monitor](#) for more information.

6.8.1 Possible Causes of Orchestrator Exceptions

The following list describes the types of exceptions that can occur:

- JSON payload parse failure.
- Any non 200 status response from the HTML Server, which includes connection failures and security errors.
- Any non 200 status response from external REST calls (includes connection failures).
- Any failure to connect to an external database.
- Any failure to find orchestration components due to security errors.
- Invalid orchestration inputs, including data type conversion errors.
- Invalid form request, data request, or cross reference request due to failure to execute.
- Cross reference or whitelist not found when an orchestration is terminated.
- Any exception thrown from a Groovy script in a rule or service request.

For schedules, the following circumstances can generate an exception:

- The cron string used to invoke the schedule is invalid.
- The name of the orchestration called by the schedule is invalid.
- Failure to connect to the scheduler. For a resilient scheduler, failure to connect to the resiliency database. For more information about scheduler resilience, see "Configuring Scheduler Resilience" in the *JD Edwards EnterpriseOne Application Interface Services Server Reference Guide*.

6.8.2 Viewing and Changing Exceptions Displayed in the List View

The List view displays 10 exception records at a time. Use the controls at the bottom of the page to view additional records. Each record contains general information about the exception, which you can expand to view detailed exception information.

The Orchestrator Monitor can access a maximum of 1000 exception records and displays a message if this amount is exceeded. You can change the search criteria or filter the results to refine the list of exceptions. If you exit and then return to the Orchestrator Monitor, the Orchestrator Monitor refreshes the results based on your previous search criteria.

To change the search criteria:

1. Click the **Select Range** drop-down list and select a preset range.

You can also select **Custom Range** from the list and click the date fields to manually set the date and time for the range.

2. Click **Search**.

3. You can enter a value in the "additional search criteria" field to display exception records based on Name, Exception, UDO Name, Host Address, or Host Name. This search is case sensitive.

4. To display exceptions for a particular UDO type, click the **Type** drop-down list and select **Orchestration**, **Notification**, or **Schedule**. Or select **All** to view all exceptions.

The Orchestrator Monitor automatically refreshes the list based on the search criteria.

To filter over the search results:

In the Filter Results field, enter a value to filter on any of the values in the exception record header (information in the collapsed exception record). For example, you can enter the name of a UDO, a status code, user, or other values displayed in the exception record header. The filter criteria is not case sensitive.

You cannot use the Filter Results field to filter on exception record details.

6.8.3 Reviewing General Exception Information in the List View

The List view displays the following general information about orchestration, notification, and schedule exceptions:

- ❑ Request Received

The date and time the request to the orchestration, notification, or schedule on the AIS Server was received.

- ❑ Name

The name of the UDO. The icon next to it indicates if it is an orchestration , notification , or schedule .

- ❑ Exception

The name and details of the exception.

- ❑ Status Code

The HTTP status that was returned when the orchestration or notification was called.

- ❑ User

The user who invoked the orchestration or notification. In the case of a scheduled object, the user who initiated the schedule start.

- ❑ Environment

This is the environment that the user who invoked the UDO signed in to. In other words, it is the environment in which the UDO was executed. To view the performance of UDOs deployed to a particular environment, see [Monitoring UDOs Based on Different EnterpriseOne User, Environment, or Role](#) for more information.

- Product Code

The product code associated with the UDO when it was created.

6.8.4 Reviewing Detailed Exception Information in the List View

Expand an exception row to find additional details about the exception, which include:

- UDO Name

The ID of the UDO in EnterpriseOne.

- URL

The URL to the orchestration or notification on the AIS Server.

- Host Address

The IP address of the requesting host, that is, the IP address of the machine that made the request to the AIS Server. If invoked by the scheduler, this is the IP address of the AIS Server.

- Host Name

The name of the host requesting the service. If invoked by the scheduler, this is the name of the AIS Server.

- Role

The role of the user who originated the request to the service (orchestration or notification).

- Processing Time

The processing time before the exception occurred.

- Schedule Name

If the service was invoked by the scheduler, the UDO ID of the schedule (for example SCH_1807120003CUST) used by the scheduler is displayed here.

- HTTP Method

The HTTP method used to invoke the service.

- Step Trace (orchestrations only)

If step trace is enabled for an orchestration, click this button to view details about each step in the orchestration. See [Understanding Orchestration Step Trace Details](#) for more information.

- Input JSON (orchestrations and notifications only)

Click this button to view the JSON input that was sent to this UDO when invoked.

- Trouble Shooting (schedules only)

Click this button to view additional details to help with troubleshooting the issue.

- Exception Message

Click this button to view the exception message.

- Application Errors

If the exception occurred as a result of an EnterpriseOne application error, click this button to view the details of the error, which include errors displayed on the EnterpriseOne form that was executed.

- ☒ Group (icon)

If an exception occurs with an orchestration or notification that calls or is called by another orchestration or notification UDO, click the Group icon to view a list of all associated UDOs that had exceptions during the execution of that request.

- ☒ Object Name

For notifications that call an orchestration, when the failure is due to the orchestration, this field displays the orchestration object name.

6.8.5 Understanding Orchestration Step Trace Details

If an orchestration configured for step tracing generates an exception, the Orchestrator Monitor includes step trace details in the exception record.

Step trace details include the orchestration or notification name, inputs and outputs, the start time and end time, duration, the status, and the orchestration response.

For an orchestration exception, the step trace lists all successfully completed steps of an orchestration. The step trace details do not include the step that failed. If the orchestration calls another orchestration, it lists the successfully completed steps in the called orchestration. When you know the last step that completed successfully, you can open the orchestration in the Orchestrator Studio and identify the step that follows to identify where the exception occurred.

Note: An administrator can access the same step trace details through the AIS Server log file in Server Manager.

6.8.6 Viewing Exceptions in the Chart View

Use the **Chart** view to display exception data visually in a chart. The bottom of the page provides options to display the data in a pie chart or a vertical, horizontal, stacked, or unstacked bar chart.

In the Chart view, you can display information by UDO name, exception, or application error. The key to the right of the chart changes to identify the data represented in the chart.

Figure 6–1 shows an example of exceptions displayed in a vertical bar chart.

Figure 6–1 Chart View in the Orchestrator Monitor

Each bar or section in a pie chart displays a number related to the data you selected to display. If grouping data by UDO name, the number represents the number of exceptions for the UDO. If grouping data by exception, the number represents the number of times the exception occurred. If grouping data by application error, the number represents the number of times the application error occurred.

6.8.7 Changing How Exceptions Are Displayed in the Chart View

In the Chart view, you can select a different date range, change how you want the data grouped by, or display exceptions for a particular UDO type.

To change the exception information displayed in the Chart view:

1. Click the **Select Range** drop-down list and select a preset range.

You can also select **Custom Range** from the list and click the date fields to manually set the date and time for the range.

2. Click **Search**.
3. Use the two Group By fields to display data with a combination of Name (UDO name), Exception, or Application Error and Day, Month, or Year.
4. In addition to the date range, you can enter a value in the "additional search criteria" field to display exception data based on these exception record details: Name, Exception, UDO Name, Host Address, or Host Name.
5. To display data based on a particular UDO type, click the **Type** drop-down list and select **Orchestration**, **Notification**, or **Schedule**. Or select **All** to view data for all UDOs.

The chart automatically refreshes based on your selection.

6.8.8 Accessing Detailed Information from the Chart

Click any bar or section in a chart to access more detailed information about that selection. For example, if you click a bar representing the number of exceptions for a particular UDO in one month, the Orchestrator Monitor displays details about exceptions that occurred for that UDO in that month. Descriptions of the details provided can be found in the [Reviewing Detailed Exception Information in the List View](#) section.

6.9 Managing Orchestrator Health and Exception Records in EnterpriseOne

Important: Before managing health and exception records in EnterpriseOne, make sure an administrator has enabled access to the Orchestrator Health and Orchestrator Exceptions program (P980060). See [Setting Up Orchestrator Health and Exception Monitoring \(Release 9.2.3\)](#)

EnterpriseOne stores Orchestrator health and exception records in the Health (F980061) and Exception (F980060) tables. EnterpriseOne provides the Orchestrator Health and Orchestrator Exceptions program (P980060) so you can access these records outside of the Orchestrator Monitor. This program enables you to:

- ❑ Delete health and exception records for which you no longer want to see data in the Orchestrator Monitor. This essentially resets the data displayed in the Orchestrator Monitor.
- ❑ Create a watchlist over a query of exception records so that you can be alerted of new exceptions. For example, you can create a basic watchlist over a query of exceptions in the Orchestrator Exceptions program. You can configure the watchlist with an error threshold of 1 so you are alerted to any failures that occur through the Watchlist feature. For more information on how to create queries and watchlists, see:
 - "Creating and Saving Queries to Search for Data" in the *JD Edwards EnterpriseOne Tools Using and Approving User Defined Objects Guide*.
 - "Adding a One View Watchlist" in the *JD Edwards EnterpriseOne Applications One View Watchlists Implementation Guide*.

"Exceptions Since Yesterday" UDOs

Oracle provides downloadable "Exceptions Since Yesterday" query and watchlist UDOs for P980060 so you can be alerted to Orchestrator exceptions in EnterpriseOne. You can also create a notification based on these UDOs. If you do not have access to these UDOs, contact your system administrator or see [Downloading and Installing the Orchestrator Monitor](#) in this guide for download instructions.

6.9.1 Managing Health Records in EnterpriseOne

To view health records in EnterpriseOne:

1. From the EnterpriseOne Navigator, select **EnterpriseOne Life Cycle Tools, Orchestrator Management, Orchestrator Health** (P980060_W980060B).
2. Click **Find** to load all records.
3. You can also search for health records based on values that you enter in the following fields:
 - ❑ **Object Type**

To find records based on a particular UDO type, enter ORCH (for orchestrations) or NTF (for notifications). EnterpriseOne does not monitor or store health information for schedules.

- **UDO Name**

This is the object ID of the UDO in EnterpriseOne. Before filtering results, load all records and then look in the "UDO Name" column to identify this value.

- **Product Code**

Enter a product code to search for UDOs associated with a particular product code.

To delete health records:

1. In the Orchestrator Health form, click the check box next to the records you want to delete.
2. Click **Delete**.

6.9.2 Managing Exception Records in EnterpriseOne

To view exception records, you must have UDO view security enabled for the orchestration, notification, and schedule UDOs that you want to view. For more information about UDO view security for Orchestrator UDOs, see [Setting Up UDO Security for Orchestrator Studio Users \(Release 9.2.1\)](#).

To view exception records in EnterpriseOne:

1. From the EnterpriseOne Navigator, select **EnterpriseOne Life Cycle Tools, Orchestrator Management, Orchestrator Exceptions** (P980060_W980060A).
2. Click **Find** to load.

The Orchestrator Exceptions form displays exception records only for the UDOs to which you have UDO view security access.

3. You can also search for exception records based on values that you enter in the following fields:

- **Object Type**

To find records based on a particular UDO type, enter ORCH, NTF (for notifications), or SCHEDULE.

- **UDO Name**

This is the object ID of the UDO in EnterpriseOne. Before filtering results, load all records and then look in the "UDO Name" column to identify this value.

- **Product Code**

Enter a product code to search for UDOs associated with a particular product code.

To delete exception records:

1. In the Orchestrator Exceptions form, click the check box next to the records you want to delete.
2. Click **Delete**.

Creating Custom Java for Orchestras

This chapter contains the following topics:

- Section 7.1, "Understanding Custom Java for Orchestras"
- Section 7.2, "Creating Custom Java"
- Section 7.3, "Deploying Custom Java"

7.1 Understanding Custom Java for Orchestras

You can include a custom Java class for service requests and rules within an orchestration. Within the custom Java classes, any number of private attributes can be declared. As long as the accessor (get/set) methods are generated for the attributes, the JD Edwards EnterpriseOne Orchestrator can assign attributes from input values from the orchestration. Then within the appropriate method, those values can be used to make AIS calls into EnterpriseOne or any other logic to either evaluate a rule or perform another action.

7.2 Creating Custom Java

When creating custom Java classes, you must reference these JAR files, which are dependencies:

- **OrchestratorCustomJava.jar.** This contains the definition of the interfaces.
- **AIS_Client.jar 1.1.0 or higher.** This contains the loginEnvironment attribute and enables AIS calls to JD Edwards EnterpriseOne.

It is not necessary to include these JARs with the deployment as they are already deployed as part of the AIS Server deployment. After creating a custom Java class, you deploy the Java class or Java classes to a JAR file.

A custom rule should implement the CustomRuleInterface class included with the OrchestratorCustomJava.jar. The interface requires a loginEnvironment variable of type com.oracle.el.aisclient.LoginEnvironment and an evaluate() method that takes no parameters and returns a Boolean value.

A custom service request should implement the CustomServiceRequestInterface class also included with the OrchestratorCustomJava.jar. The interface will require a loginEnvironment variable of type com.oracle.el.aisclient.LoginEnvironment and a process() method that takes no parameters and returns a javax.ws.rs.core.Response.

7.2.1 Using Custom Java for the Orchestration Service Request

Custom service request Java classes should implement the `com.oracle.e1.rest.orchestrator.customjava.CustomServiceRequestInterface` which requires the following methods:

- ❑ `setLoginEnvironment(com.oracle.e1.aisclient.LoginEnvironment loginEnvironment)`. The method used to perform AIS calls.
- ❑ `process()`. The method that returns `javax.ws.rs.core.Response` which is called automatically after all the attributes are set.

7.2.2 Using Custom Java for the Orchestration Rule

Custom rule Java classes should implement the `com.oracle.e1.rest.orchestrator.customjava.CustomRuleInterface` which requires the following methods:

- ❑ `setLoginEnvironment(com.oracle.e1.aisclient.LoginEnvironment loginEnvironment)`. The method used to perform AIS calls.
- ❑ `evaluate()`. The method that returns Boolean and is called automatically after all the attributes are set.

7.3 Deploying Custom Java

You must deploy the JAR file that contains the custom Java to the AIS Server. Follow the instructions accordingly depending on the server on which the AIS Server is installed:

- ❑ [Deploying Custom Java on AIS Server on Oracle WebLogic Server](#)
- ❑ [Deploying Custom Java on AIS Server on IBM WebSphere Application Server](#)

7.3.1 Deploying Custom Java on AIS Server on Oracle WebLogic Server

To deploy the custom Java JAR file to an AIS Server on Oracle WebLogic Server:

1. Deploy the JAR as a shared library on the same WebLogic Server on which the AIS Server (otherwise referred to as the `JDERestProxy`) is deployed.
2. Restart the AIS Server using Server Manager.
3. Edit the `weblogic.xml` inside the `JDERestProxy.war/WEB-INF` to reference the custom java shared library, for example:

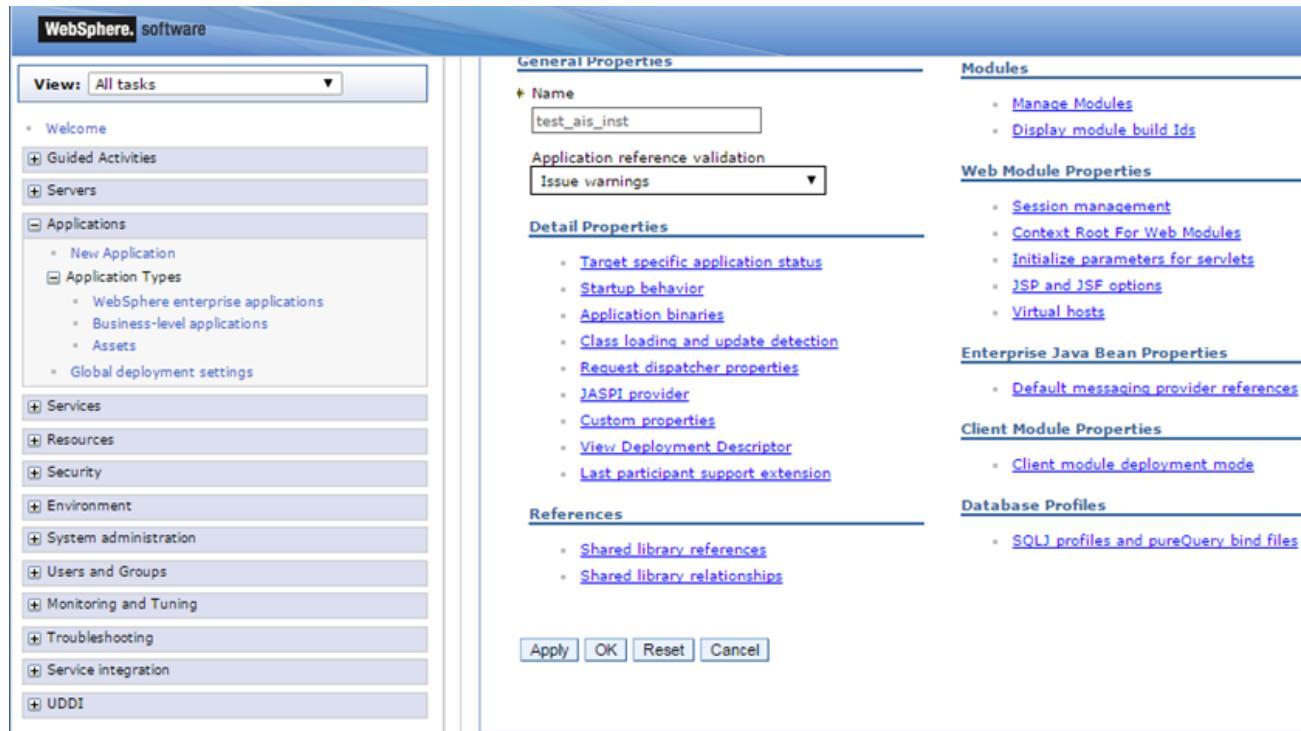
```
<?xml version="1.0" encoding="UTF-8"?>
<weblogic-web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.bea.com/ns/weblogic/weblogic-web-app
    http://www.bea.com/ns/weblogic/weblogic-web-app/1.0/weblogic-web-app.xsd"
    xmlns="http://www.bea.com/ns/weblogic/weblogic-web-app">
    <session-descriptor>
        <cookie-path>/jderest</cookie-path>
        <cookie-http-only>true</cookie-http-only>
    </session-descriptor>
    <context-root>jderest</context-root>
    <library-ref>
        <library-name>CustomJava</library-name>
    </library-ref>
</weblogic-web-app>
```

4. Redeploy `JDERestProxy` from Server Manager.

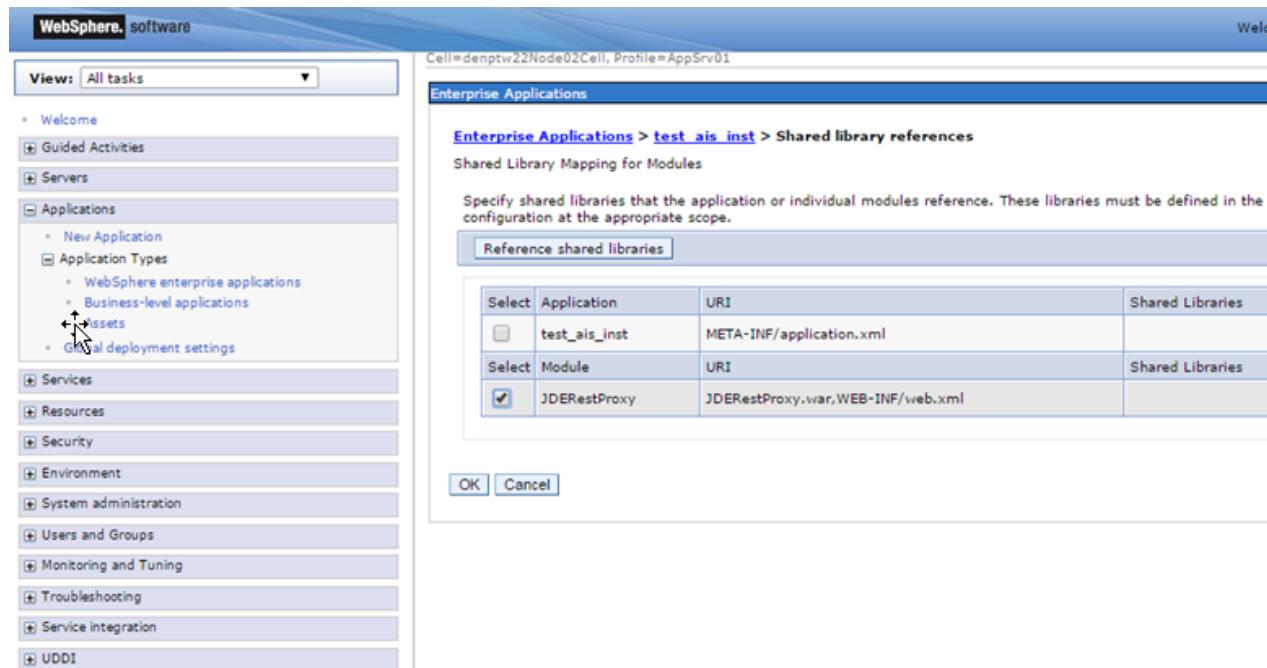
7.3.2 Deploying Custom Java on AIS Server on IBM WebSphere Application Server

To deploy the custom Java JAR file to the AIS Server on Websphere Application Server:

1. Add the JAR as a shared library:
 - a. On WebSphere, expand Environment and select **Shared Libraries**.
 - b. In ClassPath, add the path to the JAR location on the server.
2. Associate the shared library with the JDERestProxy application:



- a. In the left pane, expand **Applications**, **Application Types**, and then select **WebSphere enterprise applications**.
- b. Select the appropriate AIS deployment.
- c. Under References, select **Shared library references**.



- d. Select the **JDERestProxy** check box and then select the **Reference shared libraries** button.
 - e. Use the directional arrow to move the JAR file to the Selected group.
 - f. Click **OK**.
3. Synchronize Server Manager with the deployed application:
Saving the configuration in Websphere will redeploy the application, but you must synchronize Server Manager to recognize the deployed application.
 - a. In Server Manager, locate the AIS Server and update a setting in the Configuration section. This is required so that Server Manager detects a change in the AIS Server when you click the Synchronize Configuration button.
 - b. Apply the changes and then return to the AIS Server home page.
 - c. Click the **Synchronize Configuration** button to restart the AIS Server.

Using Apache Groovy for Custom Service Requests, Rules, and Manipulating Output (Orchestrator Studio 5.1.0 and Higher)

This chapter contains the following topics:

- [Section 8.1, "Understanding Groovy for Orchestration Components"](#)
- [Section 8.2, "Groovy Template for a Custom Service Request"](#)
- [Section 8.3, "Groovy Template for a Custom Rule"](#)
- [Section 8.4, "Groovy Template for Manipulating Output from a REST Connector Response"](#)
- [Section 8.5, "Groovy Template for Manipulating Output from an Orchestration Response"](#)
- [Section 8.6, "Additional Attributes and Methods Available in the Groovy Script Templates"](#)

8.1 Understanding Groovy for Orchestration Components

In the Orchestrator Studio, programmers can use Apache Groovy, a scripting language for the Java platform, to extend the functionality of orchestrations. In the Orchestrator Studio, you can use Groovy to:

- Create a custom service request that uses complex logic to perform transactions that cannot be configured in a standard service request.
- Create a custom rule with complex conditions that cannot be configured in a standard rule.
- Manipulate output from a REST connector response. For example, if an orchestration with a REST connector service request returns information in XML format, you can convert it to JSON for output mapping.
- Manipulate the output from an orchestration response.

Using Groovy scripts has two advantages over using custom Java. First, the Orchestrator Studio provides an editing window for creating the Groovy script, so you do not have to use an external editor to create and test your custom program. The editing window contains a Groovy template that you can use to help get you started. Second, the Orchestrator includes the Groovy script natively as part of the orchestration, so you do not have to deploy a custom program like you do with custom Java. The Groovy script simply executes as part of the orchestration.

8.2 Groovy Template for a Custom Service Request

In the Orchestrator Studio, the Custom Service Request design page contains a Groovy template that you can use to create a custom service request.

Figure 8–1 Groovy Code Template for a Custom Service Request

```

1 import groovy.transform.CompileStatic;
2 import com.oracle.el.common.OrchestrationAttributes;
3 import java.text.SimpleDateFormat;
4 @CompileStatic
5 HashMap<String, Object> main(OrchestrationAttributes orchAttr, HashMap inputMap)
6 {
7     HashMap<String, Object> returnMap = new HashMap<String, Object>();
8     // Add logic here
9     // String stringVal = (String)inputMap.get("inputStringVal");
10    // BigDecimal numVal = new BigDecimal((String)inputMap.get("inputNumVal"));
11    // SimpleDateFormat format = new SimpleDateFormat(orchAttr.getSimpleDateFormat());
12    // Date dateVal = format.parse((String)inputMap.get("inputDateVal"));
13
14    // orchAttr.writeWarn("custom log entry - warning");
15    // orchAttr.writeDebug("custom log entry - debug");
16
17    // returnMap.put("Limit Status", (String)inputMap.get("inputStringVal"));
18
19    return returnMap;
20 }

```

The following list describes the highlighted sections in the preceding code:

1. Add or remove import lines as necessary.
2. Do not modify the function definition or script definition.
3. Copy and paste the code in the commented lines and use them in your script. The code lines include:
 - ¤ Parameters for defining a string, numeric value, date format, and date value.
 - ¤ Parameters for including a warning or log statements in the AIS Server log, which can be used for debugging script issues.
 - ¤ Parameter for populating the returnMap for outputs.

The Groovy template also includes an orchAttr that enables you to include additional information in the Groovy script. See [Additional Attributes and Methods Available in the Groovy Script Templates](#).

8.3 Groovy Template for a Custom Rule

In the Orchestrator Studio, the Custom Rule Request design page provides the following code template to help you create a custom rule using Groovy:

Figure 8–2 Groovy Code Template for a Custom Rule

```

1 import groovy.transform.CompileStatic;
2 import com.oracle.el.common.OrchestrationAttributes;
3 import java.text.SimpleDateFormat;
4 @CompileStatic
5 Boolean main(OrchestrationAttributes orchAttr, HashMap inputMap)
6 {
7     Boolean result=false;
8     // Add logic for the rule here
9     // String stringVal = (String)inputMap.get("inputStringVal");
10    // BigDecimal numVal = new BigDecimal((String)inputMap.get("inputNumVal"));
11    // SimpleDateFormat format = new SimpleDateFormat(orchAttr.getSimpleDateFormat());
12    // Date dateVal = format.parse((String)inputMap.get("inputDateVal"));
13
14    // orchAttr.writeWarn("custom log entry - warning");
15    // orchAttr.writeDebug("custom log entry - debug");
16
17    return result;
18 }

```

The following list describes the highlighted sections in the preceding code:

1. You can add or remove import lines as necessary.
2. Do not modify the function definition or script definition.
3. Copy and paste the code in the commented lines and use them in your script.

The code lines include parameters for defining a string, numeric value, date format, and date value. They also include parameters for including a warning or log statements in the AIS Server log, which can be used for debugging script issues.

The Groovy template also includes an orchAttr that enables you to include additional information in the Groovy script. See [Additional Attributes and Methods Available in the Groovy Script Templates](#).

8.4 Groovy Template for Manipulating Output from a REST Connector Response

In the Orchestrator Studio, the Connector design page provides a Groovy code template that you can use to manipulate the output from a REST connector response.

The response can contain output in XML, which you can convert to JSON using Groovy. For example, your orchestration might include a step with a REST connector to a third-party system, and its response may return in XML. You could write a Groovy script to reformat that response in JSON.

Figure 8–3 Groovy Code Template for Manipulating REST Connector Output

```
1 import groovy.transform.CompileStatic;
2 import groovy.json.JsonSlurper;
3 import groovy.json.JsonBuilder;
4 import com.oracle.el.common.OrchestrationAttributes;
5 @CompileStatic
6 String main(OrchestrationAttributes orchAttr, String input)
{
7     def jsonIn = new JsonSlurper().parseText(input);
8     // modify jsonIn
9     def jsonOut = new JsonBuilder(jsonIn).toString();
10    // orchAttr.writeWarn("custom log entry - warning");
11    // orchAttr.writeDebug("custom log entry - debug");
12    return jsonOut;
13 }
14 }
```

The following list describes the highlighted sections in the preceding code:

1. You can add or remove import lines as necessary.
2. Do not modify the function definition or script definition.

The main function uses "string in" and "string out," which cannot change.

Call other functions in the script and define them below.

3. Copy and paste the code in the commented lines and use them in your script.

The code lines include parameters for defining a string, numeric value, date format, and date value, as well as writing log entries and populating the returnMap for outputs. The code lines also include parameters for including a warning or log statements in the AIS Server log, which can be used for debugging script issues.

The Groovy template also includes an orchAttr that enables you to include additional information in the Groovy script. See [Additional Attributes and Methods Available in the Groovy Script Templates](#).

8.5 Groovy Template for Manipulating Output from an Orchestration Response

The Orchestration Outputs design page provides a Groovy template that you can use to:

- ❑ Refine the outputs of an orchestration response as required by the parameters in the consuming device or program.
- ❑ Add static text to the response such as details about a company or customer.
- ❑ Delete information from the response so it cannot be read by the consuming program.

Figure 8–4 shows the Groovy template that you can use to manipulate orchestration output:

Figure 8–4 Groovy Code Template for Manipulating Orchestration Output

```

1   import groovy.transform.CompileStatic;
2   import groovy.json.JsonSlurper;
3   import groovy.json.JsonBuilder;
4   import com.oracle.el.common.OrchestrationAttributes;
5   @CompileStatic
6   String main(OrchestrationAttributes orchAttr, String input)
{
8     def jsonIn = new JsonSlurper().parseText(input);
9     // modify jsonIn
10    def jsonOut = new JsonBuilder(jsonIn).toString();
11    // orchAttr.writeWarn("custom log entry - warning");
12    // orchAttr.writeDebug("custom log entry - debug");
13    return jsonOut;
14 }

```

The following list describes the highlighted sections in the preceding code:

1. You can add or remove import lines as necessary.
2. Do not modify the function definition or script definition.

The main function uses "string in" and "string out," which cannot change.

Call other functions in the script and define them below.

3. Copy and paste the code in the commented lines and use them in your script. The code lines include parameters for defining a string, numeric value, date format, and date value. They also include parameters for including a warning or log statements in the AIS Server log, which can be used for debugging script issues.

The Groovy template also includes an orchAttr that enables you to include additional information in the Groovy script. See [Additional Attributes and Methods Available in the Groovy Script Templates](#).

8.6 Additional Attributes and Methods Available in the Groovy Script Templates

All Groovy scripts are passed an OrchestrationAttributes instance called orchAttr that contains additional information that can be used in the Groovy script. [Table 8–1](#) provides a description of the attributes. [Table 8–2](#) describes the methods available in the OrchestrationAttributes class.

Table 8–1 Attributes in the OrchestrationAttributes Class

Attribute	Type	Description
orchestrationName	String	The name of the currently running orchestration.
token	String	The token for the current session.
langpref	String	The language of the execution user.
locale	String	The locale of the execution user.
dateFormat	String	The date format of the execution user.
dateSeparator	String	The date separator of the execution user.
simpleDateFormat	String	The Java simple date format of the execution user.
decimalFormat	String	The decimal format of the execution user.
addressNumber	Integer	The address number of the execution user.

Table 8–1 (Cont.) Attributes in the OrchestrationAttributes Class

Attribute	Type	Description
alphaName	String	The name of the execution user.
appsRelease	String	The current application release.
country	String	The country code of the execution user.
username	String	The user name of the execution user.
environment (Orchestrator Studio 6.1.0)	String	The current environment of the execution user.
tempDir (Orchestrator Studio 6.1.0)	String	The temp directory configured in the rest.ini.

Table 8–2 Methods in the OrchestrationAttributes Class

Method	Response Type	Description
writeWarn(String message)	String	Write a message as a warning to the log.
writeDebug(String message)	String	Write a message as a debug to the log.
writeWarn(String message, Exception e)	String	Write a message as a warning to the log with an exception.
writeDebug(String message, Exception e)	String	Write a message as a debug to the log with an exception.
getOrchestrationName()	String	Get the name of the currently running orchestration.
getToken()	String	Get the token for the current session.
getLangPref()	String	Get the language preference for the execution user.
getLocale()	String	Get the locale of the execution user.
getDateFormat	String	Get the date format of the execution user.
getDateSeparator()	String	Get the date separator.
getSimpleDateFormat()	String	Get the Java simple date format of the execution user.
getDecimalFormat()	String	Get the decimal format of the execution user.
getAddressNumber()	Integer	Get the address number of the execution user.
getAlphaName()	String	Get the name of the execution user.
getAppsRelease()	String	Get the current application release.
getCountry()	String	Get the country code for the execution user.
getUsername()	String	Get the username for the execution user.
getEnvironment() (Orchestrator Studio 6.1.0)	String	Get the current execution environment.
getTempDir() (Orchestrator Studio 6.1.0)	String	Get the temp directory configured in the rest.ini.

Table 8–2 (Cont.) Methods in the OrchestrationAttributes Class

Method	Response Type	Description
getTempFileName(String fileName) (Orchestrator Studio 6.1.0)	String	Build a fully qualified file name in the temp directory configured in the rest.ini. The fileName parameter is the name of the file to be appended to the directory.
getUniqueTempFileName(String baseFileName) (Orchestrator Studio 6.1.0)	String	Build a fully qualified unique file name in the temp directory configured in the rest.ini. The baseFileName parameter is the name of the file to be appended to the directory. If that file name already exists, a unique ID will be appended to baseFileName until a unique name is found.
toString() (Orchestrator Studio 6.1.0)	String	Outputs a JSON string representing all values in this class.

Testing Orchestrations in the EnterpriseOne Orchestrator Client

This chapter contains the following topics:

- [Understanding the Orchestrator Client](#)
- [Testing an Orchestration in the EnterpriseOne Orchestrator Client](#)

9.1 Understanding the Orchestrator Client

The EnterpriseOne Orchestrator Client is a web application for testing orchestrations. It is available with a deployed AIS Server and runs in a web browser. You can access the Orchestrator Client from the Orchestrator Studio. Oracle recommends that you use the Orchestrator Client with an EnterpriseOne test environment, as any tests performed result in EnterpriseOne transactions that add data to the database.

In the Orchestrator Client, you can test your personal orchestrations or shared orchestrations to which you have been granted access. When you select an orchestration to test, the Orchestrator Client displays the inputs defined for the orchestration in the Inputs area, where you enter values for the inputs and then run the test. As an alternative, the Orchestrator Client gives you the option to enter raw JSON code for the input.

Figure 9–1 shows an example of testing an orchestration designed to place a customer on credit hold, where customer number 3001 was entered for the Customer input:

Figure 9–1 JD Edwards EnterpriseOne Orchestrator Client

The screenshot shows the JD Edwards EnterpriseOne Orchestrator Client interface. At the top, it says "ORACLE® JD Edwards EnterpriseOne Orchestrator Client". On the right, there are "Logout" and other navigation buttons. Below the header, the "Orchestration Name" is set to "Credit Hold". To the right of the name are buttons for "Version 2", "Generic Inputs", "Clear", and "XML Cache Refresh".

The main area is divided into sections. On the left, there's a "JSON Input" checkbox and a "Run" button. Below that is a table titled "Inputs(2)". The table has columns "Name" and "Value". It contains two rows: one for "Customer" with value "3001" and another for "Customer Name" with an empty value. There are also "Add", "Edit", and "Delete" icons for each row.

To the right of the inputs table are two panes. The top pane is labeled "Input" and contains the following JSON code:

```
{
  "inputs": [
    {
      "name": "Customer",
      "value": "3001"
    },
    {
      "name": "Customer Name",
      "value": ""
    }
  ]
}
```

The bottom pane is labeled "Output" and contains the following JSON code:

```
{
  "Customer Name": "Global Enterprises"
}
```

A green checkmark icon is located at the top right of the "Output" pane.

The Orchestrator Client displays a green check mark if the orchestration completes successfully. The Input area displays the orchestration request in JSON, and the Output area displays the orchestration response in JSON. If the orchestration was unsuccessful, the Orchestrator Client displays an "x" symbol and the Output area displays the error in JSON.

To execute the test, the Orchestrator Client invokes the orchestration on the AIS Server, passing the test values to the orchestration and in turn to the orchestration's service request to perform a transaction in EnterpriseOne. The Orchestrator Client passes the request to the AIS Server in JSON format. The AIS Server provides a JSON over REST interface (HTTP), a light-weight interface that enables AIS clients to interact with EnterpriseOne applications and forms.

9.1.1 Using cURL to Simulate Testing from a Third-Party Application or IoT Device

As an alternative to testing orchestrations in the Orchestrator Client, you can simulate a test from third-party applications or IOT devices using cURL, an open source command-line tool for transferring data with URL syntax via various protocols, including HTTPS.

When an orchestration is saved in the Orchestrator Studio, the name of orchestration is used to define an endpoint on the AIS Server. The endpoint URL is:

`http://<server>:<port>/jderest/orchestrator/<orchestrationname>`

To invoke this orchestration, third-party applications or devices use a post operation to this url, where <orchestrationname> is the name of the orchestration. In the body of the post, pass the JSON string expected by the orchestration. You can find this JSON string in the Orchestrator Client, which displays the JSON string in the Input box when you test an orchestration.

9.2 Testing an Orchestration in the EnterpriseOne Orchestrator Client

> Tutorial:

[Click here to view a recording of this feature.](#)

To access the Orchestrator Client:

1. On the Orchestrator Studio Home page, click the **Tools** link in the upper-right corner.
2. On the Tools page, click the **Orchestrator Client** icon.
3. On the Orchestrator Client Sign In screen, enter your EnterpriseOne User credentials, environment, and role.

It is highly recommended that you enter an EnterpriseOne environment used for testing, not a production environment.

4. Click the **Login** button.

To test an orchestration:

1. In the Orchestrator Client, click the **Orchestrator Name** drop-down list and select an orchestration.

The Orchestrator Client displays the inputs or name-value pairs in the Inputs area, which should be defined in the orchestration as the expected input to the orchestration. If the inputs do not appear, return to the Orchestrator Studio and define inputs for the orchestration before continuing.

2. In the Value column, enter a value for each input.

These are the values that the orchestration will pass to EnterpriseOne.

3. If needed, you can click the **add icon** (plus symbol) to add and define additional inputs.
4. Click the **Generic Inputs** check box if the orchestration uses the generic format for inputs. Leave it unchecked if the orchestration uses the standard JD Edwards EnterpriseOne format.

For more information about input formats, see [Supported Input Message Formats](#).

5. Click **Run** to test the orchestration.

The Input and Output areas show the orchestration input and response respectively in JSON format.

If the test was successful, the Orchestrator Client displays a green check mark next to the Input and Output areas. If desired, you can access EnterpriseOne and confirm that the transaction was completed by the orchestration.

If unsuccessful, the Orchestrator Client displays an "x" symbol and the Output area displays the error response in JSON format. If the orchestration fails, modify the orchestration and test it again. Make sure to click the **XML Cache Refresh** button to ensure the modified files are used in the next test.

6. To test another orchestration or start over, click the **Clear** button to reset the Orchestrator Client, which clears all values in the form.

To use JSON input for the orchestration test:

1. Click the **JSON Input** check box.
2. Enter the input in JSON format directly in the Input area.
3. Click the **Format** button to verify that the syntax of the JSON is correct.

If not formatted correctly, a dialog box displays an error message about the JSON.

4. Click **Run** to test the orchestration.

Administering the Orchestrator Studio and Orchestrations

This chapter contains the following topics:

- Section 10.1, "Understanding Orchestration Life Cycle Management (Release 9.2.1)"
- Section 10.2, "Setting Up User Access to the Orchestrator Studio"
- Section 10.3, "Upgrading Orchestration Components to User Defined Objects (Release 9.2.1)"
- Section 10.4, "Clearing Orchestration Cache on the AIS Server"
- Section 10.5, "Creating Connection Soft Coding Records for Connector Service Requests"
- Section 10.6, "Setting Up Orchestration Error and Exception Tracking (Release 9.2.2.4)"
- Section 10.7, "Setting Up Orchestrator Health and Exception Monitoring (Release 9.2.3)"

10.1 Understanding Orchestration Life Cycle Management (Release 9.2.1)

Starting with EnterpriseOne Tools 9.2.1 and Orchestrator Studio 3.0.1, orchestration components created in the Orchestrator Studio are stored as user defined objects (UDOs) in EnterpriseOne. Each orchestration component type—orchestration, service request, rule, cross reference, white list—is a separate UDO type in EnterpriseOne. Orchestrator Studio 6.0.x provides the capability to create notification and schedule components, which are also stored and managed as UDOs in EnterpriseOne.

Important: If you created orchestrations in a release prior to Orchestrator Studio 3.0.1 and EnterpriseOne Tools 9.2.1, you need to convert your orchestration components to user defined objects. See [Upgrading Orchestration Components to User Defined Objects \(Release 9.2.1\)](#) for more information.

Storing orchestration components as UDOs enables you to use the following EnterpriseOne administration tools to manage the life cycle and security for orchestration UDOs:

- Object Management Workbench - Web (P98220W)

Use this application to move orchestration UDOs between projects, check out and check in objects, and transfer objects between path codes. After Orchestrator Studio users create and test orchestrations in a test environment, use P98220W to transfer the objects to an AIS Server in a production environment. See the *JD Edwards EnterpriseOne Tools Object Management Workbench for the Web Guide* for more information.

- User Defined Object Administration (P98220U)

An administrator or person in a supervisor role uses this application to approve or reject orchestration UDOs for sharing. Typically, you can inspect UDOs in P98220U before approving them or rejecting them. However, you can only inspect orchestration component UDOs in the Orchestrator Studio. See the *JD Edwards EnterpriseOne Tools Using and Approving User Defined Objects Guide* for more information on how to use P98220U.

- Security Workbench (P00950)

Use Security Workbench to set up UDO feature, UDO action, and UDO view security, which authorizes access to the Orchestrator Studio design pages and determines the actions users can perform in the design pages. See [Setting Up UDO Security for Orchestrator Studio Users \(Release 9.2.1\)](#) in this guide for more information.

It is recommended to set up different instances of the AIS Server: one instance for designing and testing orchestrations and another instance for production. Running two instances can also help with troubleshooting orchestration issues in a production environment. In the Object Management Workbench - Web application, you can move an orchestration from a production environment to a test environment for troubleshooting.

10.2 Setting Up User Access to the Orchestrator Studio

Setting up user access to the Orchestrator Studio involves:

- [Setting Up Allowed Users](#)
- [Setting Up UDO Security for Orchestrator Studio Users \(Release 9.2.1\)](#)

10.2.1 Setting Up Allowed Users

Use Server Manager to grant EnterpriseOne users access to the Orchestrator Studio.

1. In Server Manager, access the General section of the AIS Server Configuration Group settings.
2. In the "List of Allowed Admin Service Users" field, enter a comma delimited list of EnterpriseOne user IDs, for example:

List of Allowed Admin Service Users: USR10,USR22, USR12

When added to this list, the EnterpriseOne users can use their user ID and password to sign in to the Orchestrator Studio.

Note: These group settings in Server Manager also include an option to activate the AIS Admin Service, which enables administrators to clear orchestration cache on the AIS Server. See [Clearing Orchestration Cache on the AIS Server](#) for more information.

10.2.2 Setting Up UDO Security for Orchestrator Studio Users (Release 9.2.1)

Out of the box, Orchestrator Studio users do not have access to Orchestrator Studio design pages or permission to create, publish, or modify orchestration UDOs. Access to the design pages and authorization to work with orchestration UDOs is controlled through UDO security in the EnterpriseOne Security Workbench (P00950).

Each orchestration component type is managed as a separate UDO type in EnterpriseOne. Therefor, you have to set up UDO security for each orchestration UDO type.

Before setting up UDO security for orchestration UDO types, each orchestration UDO type must be set up with the proper "Allowed Actions" in the OMW Configuration System (P98230) application. See "Define Allowed Actions for UDO Types" in the *JD Edwards EnterpriseOne Tools Security Administration Guide* for more information.

The following sections provide details and recommendations for setting up UDO feature, UDO action, and UDO view security for orchestration UDOs.

UDO Feature Security for the Orchestrator Studio

Each orchestration component design page in the Orchestrator Studio is considered a feature of the Orchestrator Studio. You must use UDO feature security to activate each design page. Out of the box, the design pages are not activated.

UDO feature security is NOT set up by user, role, or *PUBLIC. It is a system setting for activating or deactivating each component design page in the Orchestrator Studio.

Because development of an orchestration can include all five types of orchestration components (orchestration, service request, rule, cross reference, and white list), you must activate all five component design pages through UDO feature security. If you do not activate all five through UDO feature security, users cannot access any of the component design pages.

Starting with Orchestrator Studio 6.1.0 and EnterpriseOne Tools 9.2.2.4, users can use the Process Recorder in EnterpriseOne to create a form request. The Process Recorder is not available by default. You must enable the RECORDER feature in UDO feature security for users to access it.

See "Managing UDO Feature Security" in the *JD Edwards EnterpriseOne Tools Security Administration Guide*.

UDO Action Security for Orchestrator Studio Users

UDO action security controls the actions users can perform in the Orchestrator Studio. You must set up UDO action security for each orchestration component type. The three UDO action security options are:

- Create. Enables users to create UDOs for personal use. Without this permission, users can only view shared UDOs to which they have been granted access through UDO view security.
- Create and Publish. Enables users to create and "Request to Publish" UDOs to share UDOs with other users.
With this permission, a user can select the "Request to Publish" button to share a UDO. However, the UDO must be approved before it can be shared with other users.
- Modify. Enables users to modify shared UDOs. "Modify" action security inherits "Create" and "Create and Publish" permissions.

Recommendation: To simplify UDO action security for Orchestrator Studio users, it is recommended that you grant "Create, Publish, Modify" permissions to all Orchestrator Studio users for each orchestration component type. Also, users must assign a product code to each orchestration component that they create. This gives you the option to set up UDO action security by product code so that all users can work with orchestration components associated with a particular product code.

See "Managing UDO Action Security" in the *JD Edwards EnterpriseOne Tools Security Administration Guide*.

UDO View Security for Orchestrator Studio Users

UDO view security authorizes Orchestrator Studio users to view UDOs that have been shared. UDO view security is set up by user, role, or *PUBLIC. You can set up UDO view security for each shared UDO or for all shared UDOs of a particular UDO type.

Recommendation: To simplify administration, you can set up UDO view security for users by orchestration UDO type, instead of setting it up for each individual shared orchestration UDO. This enables Orchestrator Studio users to view all published ("shared") orchestration component UDOs. This eliminates a system administrator from having to set up UDO view security each time an orchestration component is shared. Also, users must assign a product code to each orchestration component that they create. This gives you the option to set up UDO view security by product code so that all users can view orchestration components associated with a particular product code.

See "Managing UDO View Security" in the *JD Edwards EnterpriseOne Tools Security Administration Guide*.

10.3 Upgrading Orchestration Components to User Defined Objects (Release 9.2.1)

Upgrade orchestration components to UDOs if you meet the following criteria:

- ❑ You installed a minimum of Orchestrator Studio 3.01 and EnterpriseOne Tools 9.2.1.
AND
- ❑ You have legacy orchestration components on the AIS Server that were created with either a previous version of the Orchestrator Studio or through manually configured XML files (pre-Orchestrator Studio).

Before you can upgrade orchestration objects to UDOs, you must have permissions to access all orchestration component design pages through UDO feature security. See [Setting Up UDO Security for Orchestrator Studio Users \(Release 9.2.1\)](#).

To upgrade orchestration components to UDOs:

1. Locate the directory on the AIS Server that contains the five orchestration component folders.
2. Add the five folders to a zip file. Do NOT include the parent folder in the zip.
3. Sign in to the Orchestrator Studio.
4. In the Orchestrator Studio Home page, click the **Tools** button in the upper right corner.
5. On the Tools page, click the **Import Tool** icon.
6. Locate the zip file with the orchestration files and click the **Upload** button.

For more information, see [Importing Orchestration Files in the Orchestrator Studio](#).

10.4 Clearing Orchestration Cache on the AIS Server

The AIS Server caches all orchestration files processed by the Orchestrator. If Orchestrator Studio users modify orchestration components that are currently in use, an administrator must clear the AIS Server cache for the modifications to take effect. Clearing the cache forces the AIS Server to reload files from disk to cache.

Note: Clearing the cache is not necessary to run new orchestration files.

You can clear the cache using either of the following methods:

- Use the AIS Administration Service to clear the AIS Server cache (**recommended**). The AIS Administration Service is a REST service on the AIS Server that is exposed like any other service on the AIS Server.
- Restart the AIS Server. This method is not recommended because:
 - If the Orchestrator is currently running an orchestration, it will result in invalid processing of that orchestration.
 - It results in server downtime, which might impact other applications that use the AIS Server.

Regardless of the method you use, Oracle recommends that you clear the cache only on an AIS Server instance used for developing and testing orchestrations. Either method clears the file cache for all orchestrations; you cannot clear the cache for individual orchestrations.

10.4.1 Activating the AIS Administration Service

Before you can use the AIS Administration Service, in Server Manager, you must set up the AIS Administration Service users and activate the AIS Administration Service.

1. In Server Manager, locate the AIS Server instance, and then access the General section of the AIS Server Configuration Group settings.
2. In the "List of Allowed Admin Service Users" field, enter a comma delimited list of EnterpriseOne user IDs, for example:

List of Allowed Admin Service Users: USR10,USR22,USR12

3. Click the **Enable Admin Service** option.

This enables the allowed Admin Service users to run the AIS Administration Service.

10.4.2 Using the AIS Administration Service to Clear Cache

Run the AIS Administration Service by placing a URL in a browser with valid EnterpriseOne credentials. The AIS Administration Service takes an EnterpriseOne username and password as input. You can invoke the service using either a GET or a POST HTTP method.

The URI is:

```
http://<ais_server>:<port>/jderest/adminservice
```

The AIS Administrator Service is able to take credentials in several forms.

For the GET call, you can either use basic authorization or you can provide the username and password as URL parameters, for example:

```
http://<ais_server>:<port>/jderest/adminservice?username=<userid>&password=<pwd>
```

For the POST operation you can use basic authorization, provide username and password as parameters, or provide username and password as JSON input, for example:

```
{
  "username": "user",
  "password": "pwd"
}
```

If the service succeeds, the response looks like this:

```
{"message": "All XML File Caches and X-Ref/Whitelist Caches have been cleared and refreshed.", "timeStamp": "2015-04-30:14.26.58"}
```

If the service fails, these are the possible reasons:

- ❑ Invalid credentials, which is indicated by the following response:

```
{"message": "Error: Authorization Failure Server returned HTTP response code: 403 for URL: http://<jas_server>:<port>/jde/FormServiceRequest", "exception": "com.oracle.e1.rest.session.E1LoginException", "timeStamp": "2015-04-30:14.28.59"}
```
- ❑ Service disabled in settings, which is indicated by the following response:

```
{"message": "Error: Admin Service is disabled in configuration. No action has been taken.", "timeStamp": "2015-04-30:14.31.41"}
```
- ❑ User has not been added to the AIS Server's AdminServiceUserList setting in Server Manager, which is indicated by the following response:

```
{"message": "User is not authorized to use the Admin Service", "timeStamp": "2015-07-07:14.23.25"}
```

10.5 Creating Connection Soft Coding Records for Connector Service Requests

This section contains the following topics:

- ❑ [Section 10.5.1, "Understanding Connection Soft Coding Records"](#)
- ❑ [Section 10.5.2, "Creating a Soft Coding Record for an Orchestrator Connection"](#)
- ❑ [Section 10.5.3, "Creating a Soft Coding Record for a REST Connection"](#)
- ❑ [Section 10.5.4, "Creating a Soft Coding Record for a Database Connection"](#)
- ❑ [Section 10.5.5, "Creating a Soft Coding Record for an FTP Server Connection \(Orchestrator Studio 6.1.0\)"](#)

10.5.1 Understanding Connection Soft Coding Records

Connector service requests require soft coding records to provide a secure access to external resources, such as a REST service, database, or an orchestration or notification on another EnterpriseOne system. A system administrator can set up a soft coding record for a single connection, and a business analyst can create one or more connectors that use this connection to access the resource on the external system.

You can create soft coding records for the following connections:

- ❑ Orchestrator connection. A connection to an AIS Server Orchestrator where orchestrations or notifications reside.
- ❑ REST connection. A connection to an external system where a REST service resides. A REST connection supports Oauth 2.0 authorization which allows orchestrations to exchange REST calls and data with Oracle Cloud services and other third-party systems. Starting with Orchestrator Studio 6.1.0, REST connections also support the transferring of files to an external REST service.
- ❑ Database connection. A connection to an external database using a JDBC driver.
For a database connection, the JDBC driver for the database must be in the AIS Server classpath.
- ❑ (Orchestrator Studio 6.1.0) Connection to an external server for transferring files through FTP/SFTP. Note that you can also use a REST connection to transfer a connector to an external system through a REST connector or a connector that uses FTP/SFTP.

When you create a connection soft coding record, you associate it with an EnterpriseOne user, role, or *PUBLIC. This enables the Orchestrator Studio user who is creating a connector service request to see a list of orchestrations, REST service, or database available through the connection. The user must be authorized to run the originating orchestration—the orchestration on the local system that will call the external orchestration, REST service, or database.

A soft coding record also includes credentials of the user authorized to invoke the external resource, such as an orchestration, REST service, database, or directory on the external system.

10.5.2 Creating a Soft Coding Record for an Orchestrator Connection

1. On the Tools page in the Orchestrator Studio, click the **Connections** icon.
2. On the Connections page, click **New Connection**.
3. On Connection Information, complete these fields:
 - ❑ **Name**. Enter a unique name for the Orchestrator connection.
 - ❑ **Description**. Enter a description to identify the connection. For example, if you are creating more than one soft coding record to connect to different locations, you should include the location in the description.
 - ❑ **Type**. Select **Orchestrator**.
 - ❑ **User/Role**. Enter the user authorized to run the originating orchestration—the orchestration on the local system that will call the external orchestration. The user can be an EnterpriseOne user, role, or *PUBLIC.
 - ❑ **Environment**. Enter the environment where the local orchestrations reside.
4. On the Service Information tab, in the **Endpoint** field, enter the URL to the Orchestrator directory on the AIS Server where orchestrations are stored, for example:

`https://<aisserverdomain>:<port>/jderest/orchestrator`

Important: Oracle strongly recommends that all external calls use the SSL protocol (https) with a certificate from a reputable certificate authority.

5. On the Security tab, enter the credentials for accessing the external system.
 - ❑ **Security Policy**
Select the authentication type used by the external AIS Server: **Basic Authorization** or **User/Password Authorization**.
 - ❑ **User**
 - ❑ **Password**
 - ❑ **Override Environment**. Use this field and the Override Role field to override the environment and role configured for the external AIS Server. For example, if you enter JDV920C1 and ADMIN in these fields for the soft coding record, when the external orchestration call is made (during the execution of an orchestration), the system will sign on to the external AIS Server with environment JDV920C1 and role ADMIN.
 - ❑ **Override Role**
6. If the connection is to an external server, on the Proxy tab, specify a proxy server for accessing the external server:
 - ❑ **Proxy Host**. Enter the URL of the proxy server.
 - ❑ **Proxy Port**. Enter the port number of the proxy server.

- ❑ **Use Proxy.** Toggle right to enable the proxy.
- 7. When complete, click **Save** at the top of the Connections page.

10.5.3 Creating a Soft Coding Record for a REST Connection

This section has been updated in support of OAuth 2.0, which is supported starting with EnterpriseOne Tools 9.2.2. This type of security allows orchestrations to exchange REST calls and data with Oracle Cloud services and other third-party systems.

1. On the Tools page in the Orchestrator Studio, click the **Connections** icon.
2. On the Connections page, click **New Connection**.
3. On Connection Information, complete these fields:
 - ❑ **Name.** Enter a unique name for the connection.
 - ❑ **Descriptions.** Enter a description to identify the connection. For example, if you are creating more than one soft coding record to connect to different locations, you should include the location in the description.
 - ❑ **Type.** Select **External Rest**.
 - ❑ **User/Role.** Enter the user authorized to run the originating orchestration—the orchestration on the local system that will call the external REST service. The user can be an EnterpriseOne user, role, or *PUBLIC.
 - ❑ **Environment.** Enter the environment where the local orchestrations reside.
4. On the Service Information tab, enter the URL for accessing the REST services in the **Endpoint** field.

Important: Oracle strongly recommends that all external calls use the SSL protocol ([https](https://)) with a certificate from a reputable certificate authority.

5. Click the **Security** tab and from the Security Policy drop-down list, select the type of security you want to use and then follow the appropriate steps:
 - ❑ For Basic Authorization, complete these fields:
 - User
 - Password
 - ❑ For OAuth 2.0 Client Credential (starting with EnterpriseOne Tools 9.2.2):
 - a. In the Token Endpoint URL field, enter the address of the server that will provide the OAuth token.
 - b. In the Client Id field, enter ID of the client.
 - c. In the Client Secret field, enter the value for client secret you have received after registering the application.
 - d. If the client requires additional header parameters, in the OAuth Parameters area, click **Add**.
The system adds a new row in the OAuth Parameters table.
6. If the connection is to an external server, on the Proxy tab, specify a proxy server for accessing the external server:
 - ❑ **Proxy Host.** Enter the URL of the proxy server.

- ❑ **Proxy Port.** Enter the port number of the proxy server.
 - ❑ **Use Proxy.** Toggle right to enable the proxy.
7. On the HTTP Headers tab, perform the following steps to add a header record that will be used in the HTTP request to execute the external REST service:
- a. Click the **Add** button.
 - b. In the pop-up box, complete these fields:
 - Name.**
 - Value.**
 - c. Slide the **Encrypt** toggle to the right to enable encryption.
 - d. Click **Add**.
- The Orchestrator Studio adds a row to the grid in the HTTP Headers tab.
- e. Add additional header records as necessary.
8. When complete, click **Save** at the top of the Connections page.

10.5.4 Creating a Soft Coding Record for a Database Connection

Important: For a database connection to work, the JDBC driver for the database must be accessible in the AIS Server classpath.

1. On the Tools page in the Orchestrator Studio, click the **Connections** icon.
2. On the Connections page, click **New Connection**.
3. In the Connection Information area, complete these fields:
 - ❑ **Name.** Enter a unique name for the connection.
 - ❑ **Descriptions.** Enter a description to identify the connection. For example, if you are creating more than one soft coding record to connect to different locations, you should include the location in the description.
 - ❑ **Type.** Select **Database**.
 - ❑ **User/Role.** Enter the user authorized to run the originating orchestration—the orchestration on the local system that will call the external database. The user can be an EnterpriseOne user, role, or *PUBLIC.
 - ❑ **Environment.** Enter the environment where the local orchestrations reside.
4. In the Database Connection Details area, complete these fields:
 - ❑ **Connection.** Enter the URL to the database.

Important: Oracle strongly recommends that all external calls use the SSL protocol (https) with a certificate from a reputable certificate authority.

 - ❑ **User.** Enter the database user.
 - ❑ **Password.** Enter the database user password.
 - ❑ **Driver.** Enter the driver for the type of database you are accessing. You need to make sure the driver is specified on the AIS Server classpath.
5. When complete, click **Save** at the top of the Connections page.

10.5.5 Creating a Soft Coding Record for an FTP Server Connection (Orchestrator Studio 6.1.0)

Create a soft coding record for a connection to a server that supports FTP or SFTP. This enables a business administrator to create connector service requests for transferring files to an FTP server.

1. On the Tools page in the Orchestrator Studio, click the **Connections** icon.
2. On the Connections page, click **New Connection**.
3. On Connection Information, complete these fields:
 - ❑ **Name**. Enter a unique name for the connection.
 - ❑ **Description**. Enter a description to identify the connection. For example, if you are creating more than one soft coding record to connect to different locations, you should include the location in the description.
 - ❑ **Type**. Select **FTP File Transfer**.
 - ❑ **User/Role**. Enter the user authorized to run the originating orchestration—the orchestration on the local system that will use the FTP connection. The user can be an EnterpriseOne user, role, or *PUBLIC.
 - ❑ **Environment**. Enter the environment where the local orchestrations reside.
4. In the FTP Connection Details area, complete the following fields:
 - ❑ **FTP Host**. Enter the URL of the FTP server.
 - ❑ **FTP Port**. Enter the port number of the FTP host server.
 - ❑ **User**. Enter the user authorized to access the directory on the FTP server.
 - ❑ **Password**. Enter the password of the user authorized to access the directory on the FTP server.
 - ❑ **Use Secure FTP (SFTP)**. Enable this toggle if the server uses secure FTP.
 - ❑ **FTP File Path**. Enter the path to the directory on the FTP server where you want the file transferred.
5. Click **Save**.

10.6 Setting Up Orchestration Error and Exception Tracking (Release 9.2.2.4)

Important: With EnterpriseOne Tools 9.2.3, the instructions in [Setting Up Orchestrator Health and Exception Monitoring \(Release 9.2.3\)](#) replace the information in this section.

In Server Manager, you can enable error and exception handling for orchestrations. When enabled, each exception in an orchestration is saved as a single serialized JSON object and stored in a directory on the AIS Server. In addition to storing these files, the directory serves as a buffer for storing orchestration requests if the EnterpriseOne system is down.

When an orchestration exception occurs, the AIS Server log states that the exception was saved to a file and provides the file location. For information on how to enable logging on the AIS Server, see [Enable Debugging on the AIS Server](#).

The following list identifies the types of default errors and exceptions that can occur during orchestration processing:

- ❑ JSON payload parse failure.
- ❑ Any non 200 status response from the HTML Server, which includes connection failures and security errors.
- ❑ Any non 200 status response from external REST calls (includes connection failures).
- ❑ Any failure to connect to an external database.
- ❑ Any failure to find orchestration components due to security errors.
- ❑ Invalid orchestration inputs.
- ❑ Invalid form service request, data request, or cross reference request due to failure to execute.
- ❑ Cross reference or whitelist not found when an orchestration is terminated.
- ❑ Any exception thrown from a Groovy script in a rule or service request.

In addition, you can specify error level handling for form service request responses when you set up exception handling in Server Manager.

To enable exception handling for orchestrations:

1. In Server Manager, under the Advanced configuration settings for the AIS Server, locate the General settings section.
2. In the General settings, click the **Save Orchestration Exceptions** check box.

This enables exception handling. If desired, perform the following steps to set up additional exception handling features.

3. In the "Full path to exceptions directory" fields, you can include a path to a directory on the AIS Server.

If you do not complete this field, files will be saved to a temporary directory on the AIS Server.

4. If you want to specify the level of error handling for form service requests, enter either or both of the following values in the Exception Scenario List field:

- ❑ FSR_ERROR

Enter if you want errors returned by a form request to be handled as a failure.

- ❑ FSR_WARNING

Enter if you want warnings returned by a form request to be handled as a failure. This level of error handling works only for form requests configured with the "Stop on Warnings" option selected in the Orchestrator Studio.

10.6.1 Understanding Orchestration Error Files

The following table describes the attributes in an orchestration error file. The attributes contain details about the orchestration and the error:

Attribute	Description
jsonRequest	A string representation of the incoming JSON request for the orchestration that failed.
requestingAddress	IP address of the host requesting the orchestration.
requestingHost	Host name of the machine requesting the orchestration.
aisURL	AIS URL used to request the orchestration.

Attribute	Description
errorStatusCode	The HTTP status code returned to the caller when the orchestration failed.
httpMethod	HTTP method sent by caller.
orchestrationName	The name of the orchestration requested.
inputs	The inputs passed to the orchestration (assuming properly formed inputs were passed).
detailInputs	If the inputs are an array, a detailed list of the inputs passed in the array.
errorDetails	An object containing details about the error encountered: the name of the exception, a timestamp, and the exception message.
requestedDateString	A date/time string of when the orchestration request was received.
environment	The EnterpriseOne environment specified in the request.
role	The EnterpriseOne role specified in the request.
jasserver	The EnterpriseOne HTML Server specified in the request.
token	The token included in the request.
deviceName	The name of the device included in the request.
username	The username included in the request.

10.7 Setting Up Orchestrator Health and Exception Monitoring (Release 9.2.3)

This section describes the tasks that you need to perform before users can use the Orchestrator Monitor. It contains the following topics:

- ❑ Downloading and Installing the Orchestrator Monitor
- ❑ Enabling Orchestrator Health and Exception Tracking in Server Manager
- ❑ Set Up User Access to the Orchestrator Monitor and Orchestrator Health and Exceptions Program
- ❑ Enable UDO View Security to Monitor Orchestrations, Notifications, and Schedules

10.7.1 Downloading and Installing the Orchestrator Monitor

To access the Orchestrator Monitor download, use the Update Center or Change Assistant to search on the bug number for your release:

- ❑ 28186193 - ORCHESTRATOR MONITOR ENHANCEMENT 9.2
- ❑ 28389735 - ORCHESTRATOR MONITOR ENHANCEMENT 9.1

The download contains the following files:

- ❑ Orchestrator Monitor.

This is delivered as an EnterpriseOne page which is referenced by an external form (P980060X|W980060XB). For the system to properly render the Orchestrator Monitor application, you must publish this page.

- ❑ Health Summary (optional).

This provides another way to view Orchestrator health details outside of the Orchestrator Monitor; it does not contain exception information. It can be published as a composed page in EnterpriseOne, but can also be made available within the My Work List composed page

(recommended). For more information, see "My Work List" in the *JD Edwards EnterpriseOne Tools Foundation Guide*.

- ❑ Exceptions Since Yesterday watchlist (optional).

Implement this watchlist and the following query so users can be alerted to Orchestrator exceptions in EnterpriseOne. For more information, see [Managing Orchestrator Health and Exception Records in EnterpriseOne](#).

- ❑ Exceptions Since Yesterday query (optional).

10.7.2 Enabling Orchestrator Health and Exception Tracking in Server Manager

An administrator must enable performance and exception tracking in the AIS Server configuration settings in Server Manager. When enabled, the system stores health and exception records in the Health (F980061) and Exception (F980060) tables in EnterpriseOne. Data from these records is used to display health and exception details in the Orchestrator Monitor (P980060X).

The AIS Server contains an additional setting to enable management of health and exception records stored in the aforementioned tables through the EnterpriseOne Orchestrator Health and Exceptions program (P980060). Access to these records in P980060 requires providing the credentials of an EnterpriseOne user who has read-write access to these tables.

In addition, in the AIS Server configuration settings, you can specify how the system processes errors and warnings returned from form requests within orchestrations.

Note: If debugging is enabled on the AIS Server, the AIS Server log will indicate that an exception was saved to the database each time the Orchestrator generates an exception. See [Enable Debugging on the AIS Server](#).

Caution: Based on the number of sessions, exception records in the database have the potential to grow rapidly.

To enable exception tracking:

1. In Server Manager, under the Advanced configuration settings for the AIS Server, locate the General Settings section.
2. In the General settings, click the **Save Orchestration Exceptions** check box to enable Orchestrator health and exception tracking.
3. In the "Full path to exceptions directory" field, enter a path to a directory on the AIS Server where exceptions are saved if the system temporarily goes offline.

When the system resumes, exceptions in the file system are automatically written to the F980060 table.

If you do not complete this field, files will be saved to a temporary directory on the AIS Server.

4. To enable access to health and exception records in P980060, enter the credentials of an EnterpriseOne user who has read-write access to F980061 and F980060 tables:
 - ❑ Exception/Health Monitor User
 - ❑ Exception/Health Monitor Password

This makes these records accessible through the EnterpriseOne Orchestrator Health and Orchestrator Exceptions program (P980060).

5. In the Exception Scenario field, enter either or both of the following values if you want errors and warnings returned from form requests to be processed as failures:
 - ❑ FSR_ERROR
Enter if you want **errors** returned by a form request to be processed as a failure.
 - ❑ FSR_WARNING
Enter if you want **warnings** returned by a form request to be processed as a failure.
This level of error handling works only for form requests configured with the "Stop on Warnings" setting in the Orchestrator Studio.

10.7.3 Set Up User Access to the Orchestrator Monitor and Orchestrator Health and Exceptions Program

An administrator must use EnterpriseOne application security to enable user access to the Orchestrator Monitor (P980060X) and the Orchestrator Health and Exceptions program (P980060). See "Managing Application Security" in the *JD Edwards EnterpriseOne Tools Security Administration Guide*.

10.7.4 Enable UDO View Security to Monitor Orchestrations, Notifications, and Schedules

Users must have UDO view security access to the orchestrations, notifications, and schedules that they want to monitor in the Orchestrator Monitor. For more information about UDO view security, see "Managing UDO View Security" in the *JD Edwards EnterpriseOne Tools Security Administration Guide*.

Understanding the AIS Server Discovery Service

The discovery service is a service on the AIS Server that enables any AIS client or consumer of orchestrations, including the Oracle Internet of Things (IoT) Cloud Service, to discover orchestrations available on the AIS Server. The service enables consumers of orchestrations to view the available orchestrations as well as the format and inputs of each orchestration.

Note: For more information on how to use the IoT Cloud Service, see:

<https://cloud.oracle.com/iot>

The URI for the discovery service is:

`http://<ais_server>:<port>/jderest/discover`

You can invoke the service using a GET HTTP method.

The discovery service is secured. You must include basic authentication credentials in the GET request to secure it.

The response is returned in an array, which lists each orchestration separately. The array includes the following details about each orchestration:

- ❑ Name. The name of the orchestration.
- ❑ Description. A description that should describe the task performed by the orchestration and include additional details about other components the orchestration uses such as a white list, rule, or cross reference.
- ❑ Input format. JDE Standard, Generic, or Cloud IoT.
- ❑ Input. The input value.
- ❑ Input type. String, numeric, or date.

Note: Orchestration descriptions are available starting with EnterpriseOne Tools release 9.2.0.3.

Example 11–1 provides an example of the orchestration details in a JSON response.

Example 11–1 Orchestrations Details in the JSON Response from the Discovery Service

```
{  
  "orchestrations": [
```

```

{
  "name": "JDE_ORCH_Sample_UpdateMeterReadings",
  "description": "This orchestration invokes the JD Edwards EnterpriseOne Speed Meter Readings application (P12120U) to update the New Fuel Meter Reading and New Hour Meter Reading for a specified Equipment Number. This orchestration requires that a white list entry for the incoming EquipmentNumber is set up in the JD Edwards EnterpriseOne Cross-Reference application (P952000).",
  "inputFormat": "JDE Standard",
  "inputs": [
    {
      "name": "EquipmentNumber",
      "type": "String"
    },
    {
      "name": "NewFuelMeterReading",
      "type": "Numeric"
    },
    {
      "name": "NewHourMeterReading",
      "type": "Numeric"
    }
  ]
},
{
  "name": "JDE_ORCH_Sample_AddConditionBasedAlert",
  "description": "This orchestration invokes the JD Edwards EnterpriseOne Condition-Based Alerts Revisions application (P1311) to create a condition-based alert (alarm or warning) when a temperature or vibration threshold is exceeded. The temperature and vibration thresholds that trigger warnings and alarms can be modified by editing the orchestration rules. This orchestration requires that two cross-references are set up in the JD Edwards EnterpriseOne Cross-Reference application (P952000): (1) a cross-reference from the incoming sensorID to the JD Edwards Equipment Number and Measument Location; and (2) a cross-reference between the Equipment Number and the alarm and warning recipients who should receive the alerts for that equipment.",
  "inputFormat": "JDE Standard",
  "inputs": [
    {
      "name": "SensorID",
      "type": "String"
    },
    {
      "name": "Date",
      "type": "Milliseconds"
    },
    {
      "name": "Time",
      "type": "String"
    },
    {
      "name": "VibrationReading",
      "type": "Numeric"
    },
    {
      "name": "TemperatureReading",
      "type": "Numeric"
    }
  ]
}

```

```
        "name": "JDE_ORCH_Sample_UpdateEquipmentLocation",
        "description": "This orchestration invokes the JD Edwards EnterpriseOne Work
with Equipment Locations application (P1704) to update the latitude, longitude,
and elevation for a specific Equipment Number. This orchestration requires that a
cross-references is set up in the JD Edwards EnterpriseOne Cross-Reference
application (P952000) to cross-reference the incoming deviceID to the JD Edwards
Equipment Number.",
        "inputFormat": "JDE Standard",
        "inputs": [
            {
                "name": "CustomerNumber",
                "type": "String"
            },
            {
                "name": "SiteNumber",
                "type": "String"
            },
            {
                "name": "DeviceID",
                "type": "String"
            },
            {
                "name": "Latitude",
                "type": "String"
            },
            {
                "name": "Longitude",
                "type": "String"
            }
        ]
    }
}
```


A

Creating Orchestrations with Orchestrator Studio 6.x.x

Note:

This appendix describes how to use Orchestrator Studio 6.x.x that was available starting with EnterpriseOne Tools 9.2.2.4.

To use Orchestrator Studio 7.0.x.0, the latest versions available with EnterpriseOne Tools 9.2.3, see:

- [Chapter 2, "Implementing the Orchestrator Studio"](#) for installation instructions
 - [Chapter 4, "Creating Orchestrations with Orchestrator Studio 7.x.x.x"](#)
-

This chapter contains the following topics:

- [Section A.1, "Understanding the Orchestrator Studio and Components"](#)
- [Section A.2, "Accessing the Orchestrator Studio"](#)
- [Section A.3, "Navigating the Orchestrator Studio"](#)
- [Section A.4, "Creating Service Requests"](#)
- [Section A.5, "Creating Rules"](#)
- [Section A.6, "Creating Cross References"](#)
- [Section A.7, "Creating White Lists"](#)
- [Section A.8, "Creating Orchestrations"](#)
- [Section A.9, "Creating Schedules for Orchestrations \(Orchestrator Studio 6.0.x\)"](#)
- [Section A.10, "Updating Version 1 Orchestrations to Version 2 Orchestrations"](#)
- [Section A.11, "Reloading Orchestrations and Orchestration Components"](#)
- [Section A.12, "Supported Input Message Formats"](#)
- [Section A.13, "Orchestration Security Considerations"](#)
- [Section A.14, "Exporting Orchestration Components from the Orchestrator Studio"](#)
- [Section A.15, "Importing Orchestration Files in the Orchestrator Studio"](#)

A.1 Understanding the Orchestrator Studio and Components

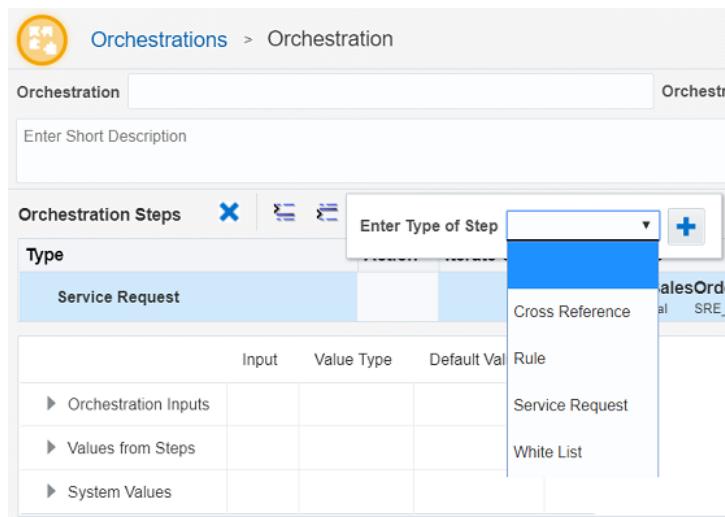
The Orchestrator Studio is a web-based application for creating orchestrations and the components that comprise an orchestration. The JD Edwards EnterpriseOne Orchestrator processes these components when executing an orchestration instance on the AIS Server.

Use the Orchestrator Studio to create the following components:

- ❑ **Orchestrations.** An orchestration is the master component that provides a unique name for an orchestration process. The orchestration is where you define the inputs for the orchestration, the expected incoming data. It also includes orchestration steps, which are invocations to the other components described in this list. When the Orchestrator invokes an orchestration, it processes the steps defined in the orchestration to enable the transfer of data from a third-party to EnterpriseOne.
- ❑ **Service Requests.** A service request contains the instructions that an orchestration uses to:
 - Perform a business transaction or query data in EnterpriseOne.
 - Send a message about a transaction to EnterpriseOne users or external users.
 - Retrieve Watchlist data from EnterpriseOne.
 - Invoke an EnterpriseOne report (Orchestrator Studio 6.1.0)
 - Transfer files between EnterpriseOne and another system (Orchestrator Studio 6.1.0).
 - Execute a custom process using custom Java or Groovy.
 - Invoke a REST service, database, a notification, or another orchestration.
- ❑ **Rules.** A set of conditions that the input to the orchestration is evaluated against to produce a true or false state. With rules, a false outcome or true outcome can invoke further orchestration steps. You can also nest rules, in which an outcome of one rule can invoke a different rule, to produce complex evaluations. You can also use custom Java to define rules.
- ❑ **Cross References.** A set of data relationships that map third-party values to EnterpriseOne values. For example, a device's serial number can be cross-referenced to a JD Edwards EnterpriseOne Equipment Number for use in service requests.
- ❑ **White Lists.** A white list contains an inclusive list of values permitted in the orchestration and terminates the orchestration process if the data is not recognized.
- ❑ **Schedules** (Orchestrator Studio 6.0.1). A schedule defines how often the system executes an orchestration or notification. You can define a schedule using minutes, hours, days, or a Cron string (for example, every Tuesday at 2:00 pm). You can attach the same schedule to multiple components.

Starting with Orchestrator Studio 6.0.1, you can create notifications in the Orchestrator Studio. A notification is a process that can run independent of an orchestration. It enables the system to notify users of business events as they happen. The notification can contain boilerplate text and a shortcut to an EnterpriseOne application and can be configured to execute a Watchlist or an orchestration. To learn how to create notifications, see the *JD Edwards EnterpriseOne Tools Notifications Guide*.

Figure 4–1 shows the drop-down list of the steps you can add to an orchestration. Each step in an orchestration is simply a reference to a cross reference, rule, service request, or white list component.

Figure A-1 *Orchestration Steps in the Orchestrator Studio*

A.1.1 Invoking an Orchestration

When a new orchestration is saved in the Orchestrator Studio, the name of orchestration is used to define an endpoint on the AIS Server. The endpoint URL is:

`http://<server>:<port>/jderest/orchestrator/<orchestrationname>`

To invoke an orchestration, calling applications or devices use a post operation to this URL, where `<orchestrationname>` is the name of your orchestration. The post operation must include security parameters that enable access to the orchestration and any EnterpriseOne application invoked by the orchestration. See [Orchestration Security Considerations](#) for more information.

A.1.2 Reusable Orchestration Components

Orchestration components are reusable. You can include the same component, such as a service request or cross reference, in more than one orchestration. If a component is used as a step in more than one orchestration, you should evaluate how it is used in the other orchestrations before modifying it.

To determine if a component is used by other orchestrations, select the component and then click the "i" button to display a pop-up window that lists the orchestrations where the component is used.

When in doubt, use the "Save As" button to create a new component from an existing one. This enables you to give it a new name and modify it as necessary, and eliminates the risk of breaking other orchestrations where the component is used.

A.1.3 Orchestrations as User Defined Objects

All orchestration components created in the Orchestrator Studio are stored as user defined objects (UDOs) in EnterpriseOne. The Orchestrator Studio includes UDO features for creating, sharing, and modifying orchestration components as UDOs. See [User Defined Object \(UDO\) Features in the Orchestrator Studio](#) for a description of the UDO features.

A.2 Accessing the Orchestrator Studio

The Orchestrator Studio is a web application that runs in a web browser. Ask your system administrator for the URL to the Orchestrator Studio.

Important: Before users can access the Orchestrator Studio, an administrator must set up security to authorize access to the Orchestrator Studio design pages and determine the actions Orchestrator Studio users can perform. See [Chapter 10, "Administering the Orchestrator Studio and Orchestrations"](#) for more information.

To access the Orchestrator Studio:

1. In a web browser, enter the URL to the Orchestrator Studio:

`http://<adf_server>:<port>/OrchestratorStudio/faces/index.jsf`

2. On the Orchestrator Studio Sign In screen, enter your EnterpriseOne User credentials, environment, and role.

Note: It is highly recommended that you enter an EnterpriseOne environment used for testing, not a production environment.

3. Click the **Login** button.

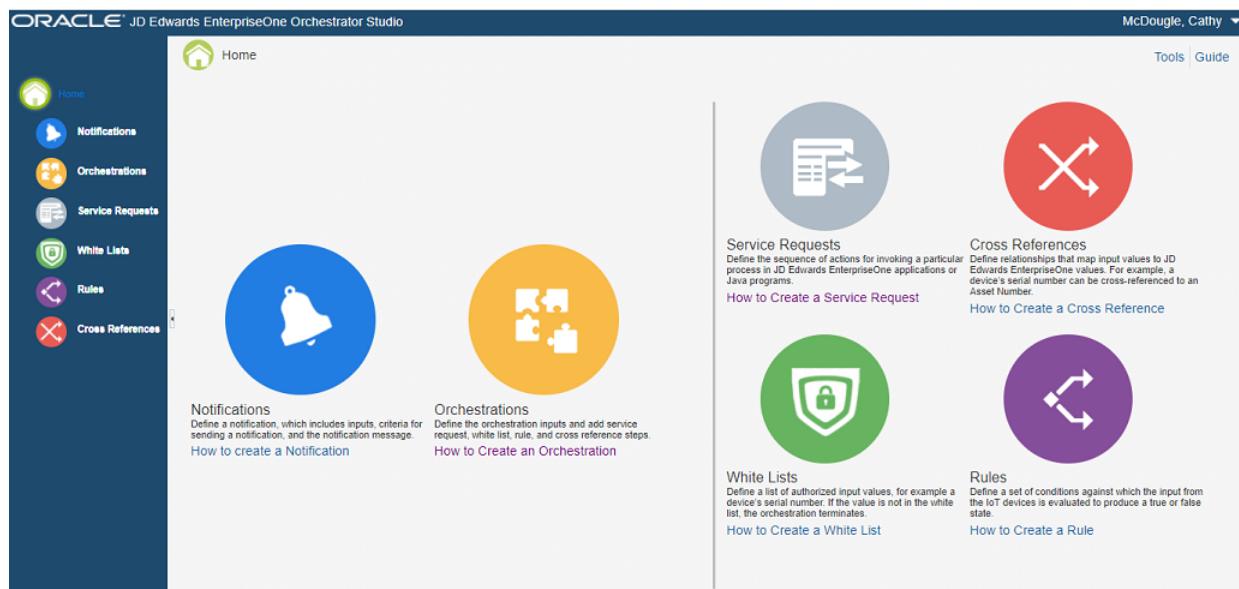
In the Orchestrator Studio, click the drop-down menu in the upper-right corner to view the path to the AIS Server. The drop-down menu also provides a link to log out of the Orchestrator Studio.

A.3 Navigating the Orchestrator Studio

The orchestration component icons on the Orchestrator Studio Home page take you to the design pages for creating and modifying each orchestration component. You can click the Home icon at the top left of the Home page to display a side panel, which provides another way to access the orchestration component design pages. You can also access this side panel within the component design pages for easy navigation between the different design pages.

Figure A–2 shows the Home page with the side panel enabled.

Figure A–2 Orchestrator Studio Home



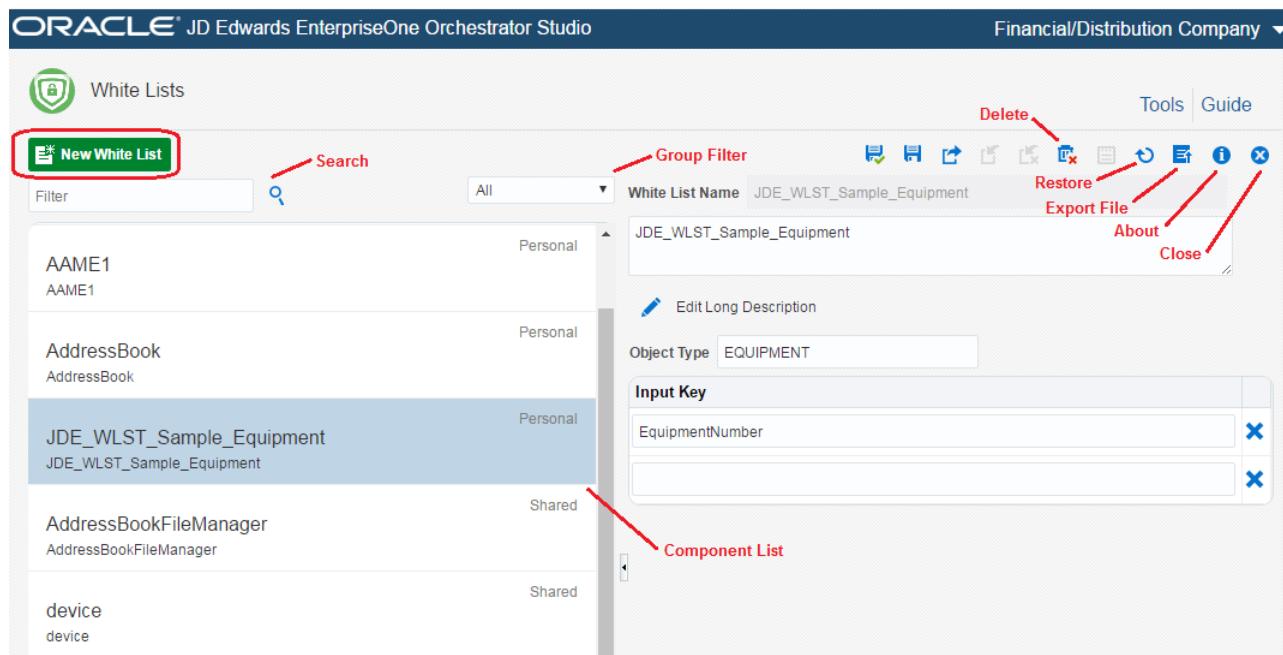
The Tools link in the upper-right corner of the Home page provides access to the Orchestrator Studio Tools page. This page provides links to tools for testing orchestrations, importing orchestration files, creating connection soft coding records for connector service requests, creating schedules, and accessing the JD Edwards EnterpriseOne web client. For more information, see the following topics:

- [Testing an Orchestration in the EnterpriseOne Orchestrator Client](#)
- [Importing Orchestration Files in the Orchestrator Studio](#)
- [Creating Connection Soft Coding Records for Connector Service Requests](#)
- [Creating a Schedule](#)

A.3.1 Orchestrator Studio Design Page Features

All Orchestrator Studio design pages contain the following features, which are highlighted in Figure A-3:

- **Component list.** Displays a list of existing components.
Use the vertical divider next to the component list to adjust the size of the list. You can click the raised tab on the divider to hide or show the component list.
- **Group Filter drop-down list.** Enables you to display components in the component list by UDO status: Personal, Pending Approval, Rework, Reserved, Shared, or All.
- **Search field.** Search for an existing component in the list.
- **New <Component> button.** Create a new component.
- **i (About).** Takes you to the About page which provides the Status, Detail, Description, and Object Name of the selected component. It also shows a list of the orchestrations where the component is used.
- **Restore All or Restore <Component>.** Restore the component to its original state if you made a mistake and do not want to save your changes.
- **Export File.** Export the component file to your local machine, which you should only use to inspect the XML of the component. See [Exporting Orchestration Components from the Orchestrator Studio](#) in this guide for more information.
- **Close.** Exit the design page.

Figure A–3 Orchestrator Studio Design Page Features

A.3.2 User Defined Object (UDO) Features in the Orchestrator Studio

Orchestration components are saved and managed as UDOs in EnterpriseOne. The Orchestrator Studio includes UDO buttons, highlighted in [Figure A–4](#), that enable you to create orchestration components for your own personal use, publish or "share" orchestration components, and modify shared orchestration components created by other users.

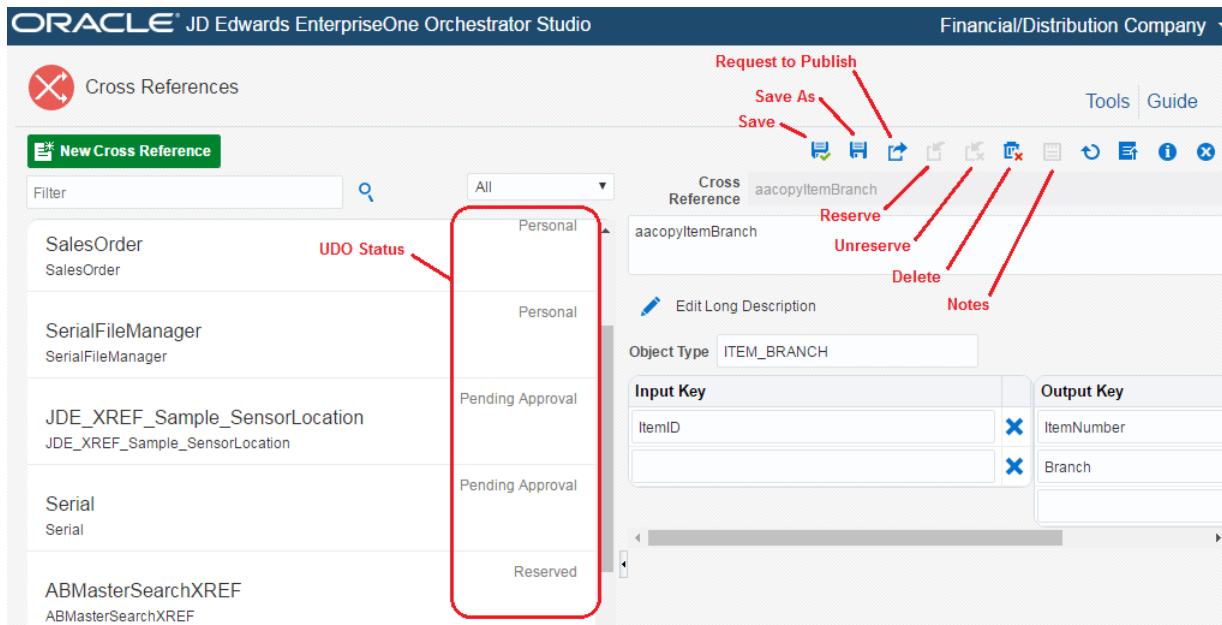
Note: The actions that you are allowed to perform in the Orchestrator Studio depend on the UDO security permissions granted to you by a system administrator. See [Setting Up UDO Security for Orchestrator Studio Users \(Release 9.2.1\)](#) in this guide for more information.

Orchestration components as UDOs enables administrators to use EnterpriseOne administration tools to manage the life cycle of orchestration components. For more information about the life cycle management of UDOs, see "UDO Life Cycle and Statuses" in the *JD Edwards EnterpriseOne Tools Using and Approving User Defined Objects Guide*.

[Table A–1](#) describes the UDO buttons in the Orchestrator Studio design pages and the life cycle status enacted by each UDO action.

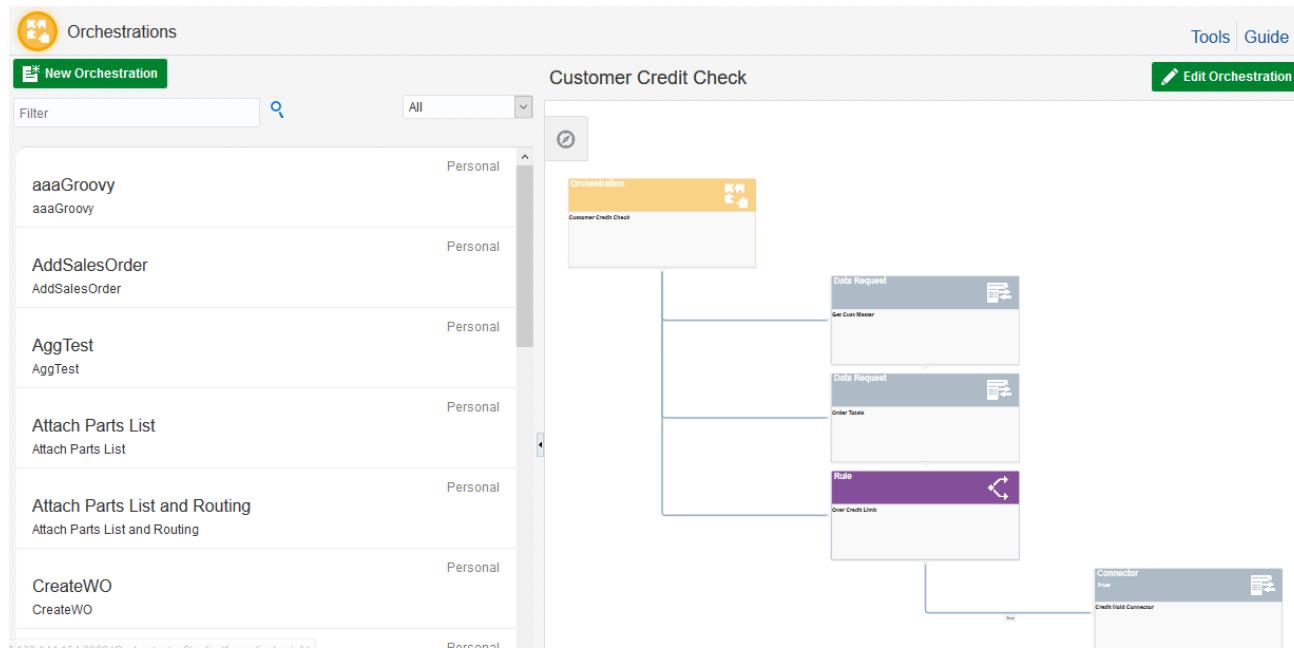
Table A-1 Description of UDO Features in Orchestrator Studio Design Pages

UDO Button	Description
Save and Save As	Saves the orchestration component to a status of "Personal." Components with a status of "Personal" are components that you are developing and have not been shared for publishing to the AIS Server.
Request to Publish	Sends the orchestration component for approval for sharing. An administrator or approver must approve it in order for the UDO to be shared. The component status changes to "Pending Approval" in the component list and then changes to "Shared" once it is approved. If rejected, that status changes to "Rework." At that point, you can edit the component and then use the Request to Publish button to send it for approval again.
Reserve	Reserves a shared UDO so you can modify it. When reserved, no other users can make changes to it. The component status changes to "Reserved."
Unreserve	Cancels the reserved component, which returns the status of the component to "Shared."
Delete	Deletes a "Personal" UDO. You cannot use this button to delete a shared UDO. Shared UDOs can only be deleted by an administrator.
Notes	Available when the component is in the "Pending Approval" status, this button enables you to add an additional note to send to the approver of the UDO. The Notes button is active only if there was a note added the first time the UDO was sent for approval using the "Request to Publish" button. This feature enables you to add an addendum to the original note.

Figure A-4 Design Page UDO Features

A.3.3 Working with the Graphical Representation of an Orchestration

Figure A-5 shows an example of the initial Orchestrations page, which lists all orchestrations that you have access to and provides a graphical representation of an orchestration with all its components.

Figure A–5 Orchestrations Page

The graphic area includes the following features:

- ☒ **Control Panel**

The Control Panel icon in the upper-left corner of the graphic area contains directional controls to pan left, right, up, and down, as well as zoom in or zoom out. "Zoom to Fit" displays the entire graphical representation in the window. The layout buttons change the layout to vertical, horizontal, tree, radial, or circle, which helps to view more complex orchestrations that contain multiple components.

- ☒ **Informational hover help**

Hover your mouse over a component in the graphical area to view an enlarged image of the component. Hovering over the labels on the lines between a rule component and its child components magnify the "True" or "False" label. A "True" label indicates the child component will be invoked if the conditions in the rule are met. A "False" label indicates the child component will be invoked when the condition of the rule is not met.

- ☒ **Isolate and Restore buttons**

The Isolate button on the left side of a component shows only that component in the graphic area. Restore displays all orchestration components.

- ☒ **Access to the design page for editing the component**

When you click a box representing a component, the Orchestrator Studio takes you to the design page for modifying that particular component.

A.4 Creating Service Requests

This section contains the following topics:

- ☒ [Understanding Service Requests](#)
- ☒ [Creating a Service Request](#)
- ☒ [Configuring a Form Request in the Orchestrator Studio](#)

- [Creating a Form Request with the JD Edwards EnterpriseOne Process Recorder \(Tools 9.2.2.4\)](#)
- [Configuring a Data Request](#)
- [Configuring a Message Request](#)
- [Configuring a Connector Service Request](#)
- [Configuring a Watchlist Service Request \(Orchestrator Studio 6.0.1.0\)](#)
- [Configuring a Report Service Request \(Orchestrator Studio 6.1.0.0\)](#)
- [Configuring Form Request and Data Request Processing](#)
- [Configuring a Custom Service Request with Java \(Advanced\)](#)
- [Configuring a Custom Service Request with Groovy \(Advanced\)](#)

A.4.1 Understanding Service Requests

A service request provides the instructions that enable an orchestration to carry out a particular task. You can create the following types of service requests in the Orchestrator Studio:

- **Form Request**

A form request contains the instructions for the orchestration to perform a particular business transaction in EnterpriseOne.
- **Data Request**

Use a data request in an orchestration to query and return values from an EnterpriseOne table or business view. You can also configure a data request to perform an aggregation on data to return aggregate amounts.
- **Message**

Use a message request if you want an orchestration to send a message to an external email address (requires an SMTP server) or EnterpriseOne users through the EnterpriseOne Work Center.
- **Connector**

Use a connector request to invoke an orchestration, notification, REST service or database. For a connector to an orchestration, a connector can invoke a local orchestration or an orchestration on another AIS Server, such as an AIS Server on another EnterpriseOne system in a multisite operation. (Connectors to a notification and database are available starting with Orchestrator Studio 6.0.x.)

Starting with Orchestrator Studio 6.1.0, you can configure a connector to use FTP or a REST call to transfer report output or other files to an external server.
- **Custom**

Use custom Java or Groovy to execute a custom process.
- **Watchlist (Orchestrator Studio 6.0.x)**

Use a Watchlist service request to use the information from Watchlists, such as critical or warning states, threshold levels, and number of records, within an orchestration.
- **Report (Orchestrator Studio 6.1.0)**

Use a report request to invoke a batch version of a report in EnterpriseOne.

Before you create a service request, you must first identify the EnterpriseOne task or business process that you want the service request to perform. See [Identifying the Service Request Information for the Orchestration](#) for more information.

A.4.1.1 Understanding the Application Stack Option in a Form Request

Use the Application Stack option to create a form request that establishes a session for a specific application and maintains that session across calls. With application stack processing, the form request can invoke multiple forms in different applications to complete a business process. Without application stack processing, each form in a form request is opened independently.

If you use the Application Stack option, the form request must include instructions for navigating between forms. Without the navigation, additional forms cannot be called.

Only values from the last form in the application stack can be returned; return controls specified for any form except the last form are ignored.

A.4.2 Creating a Service Request

To create a service request:

1. On the Orchestrator Home page, click the **Service Requests** icon.
The Orchestrator Studio displays the initial Service Requests page.
2. On this page, click **Create Service Request**, and from the drop-down list, select the type of service request that you want to create:
 - ☒ Form Request

Recommended:

Instead of manually creating a form request in the Orchestrator Studio, use the JD Edwards EnterpriseOne Orchestrator Process Recorder to create it. You can access the Process Recorder in EnterpriseOne and record each step or action that you want the form request to perform, which the Process Recorder then saves as a form request. See [Creating a Form Request with the JD Edwards EnterpriseOne Process Recorder \(Tools 9.2.2.4\)](#).

- ☒ Custom
- ☒ Data Request
- ☒ Message
- ☒ Connector
- ☒ Watchlist
- ☒ Report

Alternatively, access the Orchestrations design page and add a Service Request step to an orchestration. At the end of the service request row, click **Edit** (pencil icon) and select the service request type that you want to create.

3. On the Service Requests design page, click the **Product Code** drop-down list to select a product code to associate with the service request.
This gives an administrator the option to manage UDO security for orchestration components by product code.
4. On the *Service Request Type* design page, complete these fields:

- ❑ **Service Request.** Enter a name for the service request. Do **NOT** include special characters in the name. You should include the service request type, such as "form request" or "data request," in the name to distinguish it from other service requests listed in the Service Request design page.
 - ❑ Short description field. In the space provided, enter a description with a maximum of 200 characters. This description will appear below the service request name in the component list.
5. (Optional) Click **Long Description** to provide additional details about the purpose of the service request.
 6. Click **Save**.
- The first time a new service request is saved, it is saved as a "Personal" UDO. After configuring the service request, you can use the UDO buttons described in the [User Defined Object \(UDO\) Features in the Orchestrator Studio](#) section to move the service request to the appropriate status.
7. Refer to the appropriate section for instructions on how to configure the service request type: form request, data request, message request, connector request, Watchlist request, report request, custom Java request, or custom Groovy request.

A.4.3 Configuring a Form Request in the Orchestrator Studio

A form request contains the instructions for an orchestration to perform a particular business transaction in EnterpriseOne.

Create the form request as described in [Creating a Service Request](#), and then perform these tasks:

- ❑ Load the EnterpriseOne Application Form Fields and Controls
- ❑ Configure Form Request Actions
- ❑ Configure the Order of Execution
- ❑ Populating Multiple Grids with Repeating Inputs

> Tutorial:

[Click here to view a recording of this feature.](#)

Recommended:

Instead of using the Orchestrator Studio to create a form request, use the JD Edwards EnterpriseOne Orchestrator Process Recorder. You can access the Process Recorder in EnterpriseOne and record each step or action that you want the form request to perform, which the Process Recorder then saves as a form request. See [Creating a Form Request with the JD Edwards EnterpriseOne Process Recorder \(Tools 9.2.2.4\)](#).

A.4.3.1 Load the EnterpriseOne Application Form Fields and Controls

1. In the Available Actions area, complete the following fields:
 - ❑ **Application.** Enter the application ID.
 - ❑ **Form.** Click the drop-down list to select the form.
 - ❑ **Version.** Enter the version ID.

If you leave Version blank, the Orchestrator will use the default version when it executes the form request.

Starting with Orchestrator Studio 6.1.0, leaving the version field blank with the Application Stack option selected gives you the option to pass in the version from an orchestration input. This enables you to configure an orchestration that includes rules with conditions so that clients can call different versions dynamically.

When you add the form request to an orchestration, click the **Add Inputs to Orchestration** button to add the variable representing the version to the Orchestration Inputs area. The variable is displayed with the application ID followed by _Version, for example P03013_Version.

2. Click the **Form Mode** drop-down list and select the appropriate form mode: **Add, Update, or Inquiry**.

At runtime, the controls that appear on an EnterpriseOne form are dependent on the form mode as specified in Form Design Aid (FDA). The form mode ensures that you can see all of the form controls in the Orchestrator Studio that are available in the form at runtime.

3. Select the following options as appropriate:

- **Application Stack.** Application stack processing enables the form request to establish a session for a specific application and maintains the session across calls. With application stack processing, the form request can invoke multiple forms in different applications to complete a business process. Without the application stack processing, each form in the form request is opened independently.

If you use the Application Stack option, you must configure the form request with the actions to navigate between forms. Without this navigation, additional forms will not be called.

- **Run Synchronously** (Orchestrator Studio 6.1.0). Enabled by default, this ensures that the form request completes all actions before the Orchestrator executes the next step in the orchestration. It also ensures that the events in the EnterpriseOne application are run synchronously. This option is recommended if there are steps that follow the form request in the orchestration or if you configure the form request to return a response.

Caution: If the form request involves invoking a control that launches an asynchronous report or any long running process, it is possible that the Orchestrator could time out if Run Synchronously is enabled.

- **Bypass Form Processing.** This option enables the form request to ignore event rules when loading the form fields and controls in the Available Actions grid. For example, some forms have event rules to perform a find on entry or hide certain fields, which can cause the load to fail or prevent certain fields from loading in the Available Actions grid.

4. Click the **Load Form** button.

The Orchestrator Studio loads the controls and fields in the grid. The name of the form is displayed in the first row in the grid.

If a form request needs to invoke more than one application to complete the desired task, you can load controls and fields for additional application forms as needed.

A.4.3.2 Configure Form Request Actions

Figure A–6 shows the Available Actions area, where you specify the EnterpriseOne fields and controls used to perform the desired business transaction in EnterpriseOne.

Figure A–6 Available Actions Area in the Form Request Design Page

Available Actions		Application Stack <input checked="" type="checkbox"/>		Run Synchronously <input checked="" type="checkbox"/>		Bypass Form Processing <input type="checkbox"/>			
Application	P1311	Form	W1311B - Condition-Based Alerts Revisions	Version		Form Mode		<input type="button" value="Upload"/>	
Description	Mapped Value	Default Value	ID	Version	Form Mode	Return	Variable Name		
Condition-Based Alerts Revisions			P1311_W1311B		<input checked="" type="radio"/>		<input type="button" value="X"/>	<input type="button" value="Edit"/>	
Buttons and Exits									
Cancel			12				<input type="button" value="Edit"/>	<input type="button" value="Delete"/>	
Create W.O.			114				<input type="button" value="Edit"/>	<input type="button" value="Delete"/>	
Failure Analysis			124				<input type="button" value="Edit"/>	<input type="button" value="Delete"/>	
Investigation			120				<input type="button" value="Edit"/>	<input type="button" value="Delete"/>	
Investigation Msg			115				<input type="button" value="Edit"/>	<input type="button" value="Delete"/>	
Notification			119				<input type="button" value="Edit"/>	<input type="button" value="Delete"/>	

Use the following features to configure the actions in the form request. After configuring each action, click the **Add Action** button at the end of each row to add it to the Order of Execution area.

□ **Description** (informational only)

This column displays the controls and fields for each form in a collapsible/expandable parent node named after the EnterpriseOne form. Child nodes categorize other items and include a Buttons and Exits node, a node for displaying the available Query by Example (QBE) fields in a grid (if applicable), and a node for displaying all available columns in a grid.

□ **Mapped Value**

This is an editable column for identifying the inputs to the form request. In each field to which you want to map an orchestration input, enter a variable name. When you add this form request to an orchestration, you map the orchestration input to this variable.

The only fields to which you can map inputs are EnterpriseOne editable fields. The variables names for the inputs in the form request should match the inputs defined in the orchestration to which you add the form request. If they do not match, you can use the "Transformations" feature to map input names after you add the form request to an orchestration. See [Adding Inputs to an Orchestration](#) and [Mapping Orchestration Inputs](#) for more information.

Instead of a mapped value, you can enter a hard-coded value in the Default Value column or you can click the "Text substitution" check box to combine inputs into a text string.

You can also use this field to populate multiple rows in multiple grids. See [Populating Multiple Grids with Repeating Inputs](#) for more information.

□ **Default Value**

- For an editable field to which you want to map an input, use this column to enter a hard-coded value. You can also add a hard-coded value to use if the mapped value returns a null.
- For a check box or a radio button, you can select it to include it. If there are multiple radio buttons grouped together on a form, select only one. If you select more than one, only the last radio button in the Order of Execution list will be used.
- For a combo box control (a list of items in a drop-down menu), the Studio displays a drop-down menu in the Default Value column in which you can select the appropriate item from the list.

□ **Select All Rows** (row) (Orchestrator Studio 6.1.0)

Add this action to select all rows in a grid. Use this action if the form has a button to perform an action on all selected grid rows. For example, if you want to update the PO status for all rows in a grid, add this action to the Order of Execution area and then configure the next action to update the PO status.

- ▣ **Select First Row** (row) or **Select Row** (Orchestrator Studio 6.1.0)

Add this action if the form requires selecting the first row in a grid to perform a particular action.

Starting with Orchestrator Studio 6.1.0, you can enter a variable in the Mapped Value column of this row, which enables you to map an orchestration input to this variable. Or you can enter a value in the Default Value column to select a specific row. To select the first row in the grid, leave these columns blank when adding this action to the Order of Execution.

- ▣ **Row Number for Update** (row)

If the task requires updating a particular row in an input capable grid, then map the appropriate input value containing the row number to the "Row Number for Update" row, or specify the row number in the Default Value column.

Do NOT use this action if the transaction involves adding rows to a grid.

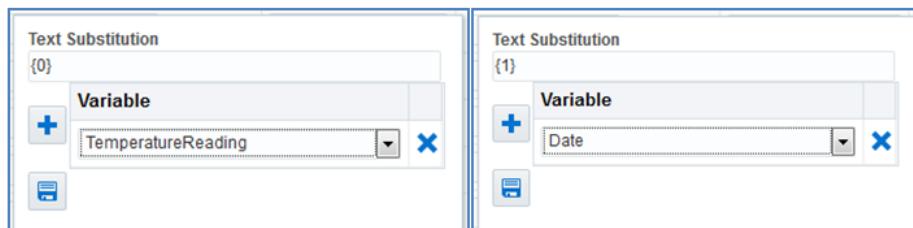
- ▣ **"Text Substitution"** check box column

Use text substitution to combine input values into a text string for an input into an EnterpriseOne field. For example, you can create a text string such as "Temp was {0} at {1}" that contains input values for temperature and time.

Note: If you use text substitution for the input, the field cannot contain a mapped value. The substituted text string becomes the default value for the field when you add it to the Order of Execution list.

1. Click the **Add Text Substitution** button (plus symbol) in the row that contains the field for which you want to substitute the input with text.
2. In the Text Substitution field in the pop-up box, use a variable for the first input by entering {0} (with brackets).
3. In the Variable field, enter the input variable name.
4. Click the **Add** (plus symbol) to add the variable for the text substitution.
5. Add the next variable, {1}, and then in the next blank row below the Variable field, enter the input variable name.
6. Click **Add** to add it to the text substitution.
7. Click **OK** to save the variables.

The following image shows an example of variables added for a temperature input value and a date input value.



- **ID** (informational only)

This column displays the ID of the control or field in EnterpriseOne.

- **Form Mode** (informational only)

If a form mode was selected, this column displays the form mode in the form row.

- **Return**

Select the check box in this column for fields that contain values that you want returned in the orchestration response. If you do not specify any return values, all values on the form will be returned. Return values may be used by customized third-party gateway programs or devices to perform further processing based on the values returned from EnterpriseOne applications.

Caution: If using the Application Stack option:

- Only values from the last form in the application stack are returned; return controls specified for any form except the last form are ignored.
 - If you use a form to add or update a value that upon saving exits to another form, such as a Find/Browse form, the return values will come from the last form that appears after the save. In this scenario, you need to add the last form to the application stack and select the return controls from the last form.
-

- **Variable Name**

Enabled when the Return check box is selected, use this field to enter a variable name for the return value. When you add the form request to an orchestration, this name appears in the orchestration inputs grid, which makes the returned value available for mapping to subsequent steps in the orchestration.

You can also use this field to return a data set from an application grid. See [Retrieving and Passing Data Sets in an Orchestration \(Orchestrator Studio 6.0.x\)](#) for more information.

- **Return Hidden Fields** (left arrow icon in the Return column)

Use this feature to return data from "hidden" fields or grid columns not displayed in the Available Actions area. Hidden fields are fields that appear only after performing a particular action in a form. In the EnterpriseOne web client, use the field-level help (Item Help or F1 key) to identify the control IDs for hidden fields and grid columns. The control ID is displayed in the Advanced Options section in the help pop-up window.

To return a hidden field:

1. In the Available Actions area, in the row of the form that contains the hidden fields, click the left arrow in the Return column.
A dialog box appears displaying the control IDs and associated variable names of any return fields already selected in the Available Actions area.
2. For hidden fields, enter the control ID of the hidden field in the Return Form Ids field. Optionally, you can enter a variable name to represent the return value in the associated Name field.

For multiple controls, use a bar delimiter to separate the values in both fields, making sure the order of the values in each field match.

3. For hidden grid columns, enter the control ID for the grid column in the Return Grid Ids field. Optionally, you can enter a variable name to represent the return value in the associated Name field.

You must enter a variable name for the return value if you want it to appear in the orchestration inputs list in the orchestration.

An example of the notation for multiple grid columns in a grid is: 1[32, 34, 42]

Where 1 represents the first grid in the form (because some forms can have multiple grids), and 32, 34, 42 each represent a column in the grid.

Use a bar delimiter to separate the variable names, making sure the order of the variable names matches the order of the grid control IDs, as shown in the following example:

Value	ID	Version	Form Mode	Return
Return Form Ids 8 16 22				
Form Id Names User ID Address Phone				
Return Grid Ids 1[14,13,11,12]				
Grid Id Names UDC Hard Coded UDC Special Handling Description 01 Description 02				
<input type="button" value="OK"/> <input type="button" value="Cancel"/>				

- ¤ (Optional) "Options" icon (at the end of the first row in the grid)

This option contains settings that determine how the AIS Server processes the form request. See [Configuring Form Request and Data Request Processing](#) for details.

A.4.3.3 Configure the Order of Execution

After actions are added to the Order of Execution grid, you can reorder them using the up and down arrow buttons to the right of each row. You can also delete any actions using the Delete (X) button. [Figure A-7](#) shows the Order of Execution area.

Starting with Orchestrator Studio 6.1.0, you can enter or modify values in the Mapped Value and Default Value columns in the Order of Execution area.

Figure A–7 Order of Execution in the Service Request Design Page

The screenshot shows the 'Service Requests > Form Request' interface. At the top, it displays 'Service Request AddCBM_Alarm' and 'Product Code 55'. Below this is a table titled 'Order of Execution' with columns: Description, Action, Mapped Value, Default Value, and two control columns with up/down arrows and an 'X'.

Description	Action	Mapped Value	Default Value		
Asset Number (edit)	SetControlValue	equipmentNumber		<input type="button" value="^"/>	<input type="button" value="v"/>
Description	SetControlValue	description		<input type="button" value="^"/>	<input type="button" value="v"/>
Alert Level	SetControlValue		2	<input type="button" value="^"/>	<input type="button" value="v"/>
Event Date / Time	SetControlValue	date		<input type="button" value="^"/>	<input type="button" value="v"/>
Event Time Formatted	SetControlValue	time		<input type="button" value="^"/>	<input type="button" value="v"/>
Automated Response Type (edit)	SetControlValue		1	<input type="button" value="^"/>	<input type="button" value="v"/>
OK	DoAction			<input type="button" value="^"/>	<input type="button" value="v"/>

A.4.3.4 Populating Multiple Grids with Repeating Inputs

You can configure a form request to map repeating inputs into one or more grids.

In the Available Actions area, rows for editable grids include a default value of GridData in the Mapped Value column. This value can be used to associate a set of repeating inputs to the grid. If you need to populate more than one grid in an orchestration, change this value to identify each grid. And then make sure that the "name" tag that identifies the repeating inputs in the detail inputs section of the orchestration input matches this value.

A.4.4 Creating a Form Request with the JD Edwards EnterpriseOne Process Recorder (Tools 9.2.2.4)

The JD Edwards EnterpriseOne Orchestrator Process Recorder enables you to record a series of actions in EnterpriseOne and save those actions as a new form request. This simplified method to create a form request is an alternative to manually creating a form request in the Orchestrator Studio.

> **Tutorial:**

[Click here to view a recording of this feature.](#)

After launching and starting the Process Recorder in EnterpriseOne, you access an application and perform each of the steps to complete the business process or transaction that you want the form request to perform.

The Process Recorder automatically adds inputs to the form request for each field in which you enter data during the recording. The input is the name of the field, and the data you enter becomes the default value. For example, if you click a field called **Customer Number** and enter 1001, the Process Recorder records an input called Customer Number with a default value of 1001. You can delete, change, or replace the value with a variable by editing the form request in Orchestrator Studio.

When finished, before you save the process as a form request, you can select any of the controls or grid columns on the form that contain values that you want the form request to return.

After you save the form request in the Process Recorder, you can open the form request in the Orchestrator Studio, modify it as necessary, and add it to an orchestration. If you have an open

Orchestrator Studio session, you will have to close it and sign in again to access the newly created form request.

A.4.4.1 Rule for Creating a Form Request in the Process Recorder

When using the Process Recorder to create a form request, follow these rules:

- ❑ Start the recording and then launch the first application from the carousel, Favorites, a link on an EnterpriseOne page, or Fast Path. Or you can open an application and then start the recording before performing the first action in a form. You cannot start recording after performing an action in an EnterpriseOne application.
- ❑ If the process involves adding multiple records, perform only the steps to add a single record. Later, when you add the form request to an orchestration, you can use the Iterate Over feature in the Orchestrator Studio to configure the form request to add multiple records.
- ❑ Some business processes entail using forms in more than one application. You can record this as long as you access the other forms from a Form or Row menu. However, you cannot return to the EnterpriseOne home page and launch another application. If the business process requires launching a second application from the EnterpriseOne home page, then record it as a separate form request.
- ❑ When recording a process in the Process Recorder, each transaction is saved in EnterpriseOne just as it would be if you were not using the Process Recorder. To ensure that you do not sully production data, it is highly recommended that you use the Process Recorder in an EnterpriseOne test environment. If recording in a production environment, delete any transactions that were created while using the Process Recorder.

A.4.4.2 Prerequisites

The Process Recorder is disabled by default. To make it available in EnterpriseOne, a system administrator must enable both of the following security types:

- ❑ UDO feature security.
A system administrator must enable the RECORDER feature in UDO feature security. See "Managing UDO Feature Security" in the *JD Edwards EnterpriseOne Tools Security Administration Guide*.
- ❑ UDO action security.
A form request is a type of service request, and service requests are saved and managed as a user defined objects (UDOs) in EnterpriseOne. Therefore, users must be authorized to create service requests through UDO action security before they can access and use the Process Recorder to create a form request. See "Managing UDO Action Security" in the *JD Edwards EnterpriseOne Tools Security Administration Guide*.

A.4.4.3 Using the Process Recorder to Create a Form Request

1. On the EnterpriseOne home page, click the user drop-down menu in the upper right corner and select **Record a Process**.
2. In the **Process Recorder** pop-up box, click **Start** to start the recording.
3. Access the first application from the carousel, Favorites, a link on an EnterpriseOne page, or Fast Path.
EnterpriseOne highlights the application title bar and application tab in the carousel (if enabled) of the application that is being recorded.
4. Continue through the application steps that you want the form request to perform.

The Process Recorder automatically adds inputs to the form request for each field in which you entered data during the recording. The input is the name of the field, and the data you enter becomes a default value. You can change the name of the input and delete or change the default value by editing the form request in Orchestrator Studio.

Important: If you return to the EnterpriseOne home page or try launching a separate application from the Carousel while recording, the Process Recorder pauses the recording and will not record those steps. To resume the recording, you can click the **Paused** button and the Process Recorder will resume recording, returning you to the point in the process where you initially paused the recording.

5. After performing the final step, click **Stop**.

If you stopped the recording prematurely, click **Cancel**, and then continue to record the remaining actions. To discard the process and start over, click **Discard**.

6. If you want the form request to return values from the last form, click the controls and columns on the form that contain values you want returned. If you want the form request to return the values of all controls and columns on the last form, leave the Return Controls and Return Columns fields blank.

The Process Recorder adds any controls and columns that you click to the respective Return Controls and Return Columns fields. If you add a control or column by mistake, click it again to remove it from the list.

7. To complete the recording, enter a name, product code, and description for the form request.

8. Click **Save**.

If you need to modify the form request, see [Configuring a Form Request in the Orchestrator Studio](#). To add it to an orchestration, see [Adding Steps to an Orchestration](#).

A.4.5 Configuring a Data Request

You can configure a data request to:

- Query and return data from an EnterpriseOne table or business view.
- Perform an aggregation of data from a table or business view and return aggregate amounts.

In a standard data request that returns data directly from a table or business view, you can use variables to define the return fields. In a data request that performs an aggregation, you can use variables to define the "group by" fields and the returned aggregate amounts.

When you add a data request to an orchestration, any variables used to define the returned data in the data request are automatically added to the orchestration as inputs. This enables you to map the returned data to subsequent steps in an orchestration.

Important: Variables are only supported in the first row of a response. If a query contains multiple rows, only the values from the first row will be available as variables.

A.4.5.1 Configuring a Data Request to Return Field Data

A data request includes filtering criteria and indicates the fields that contain data that you want returned. For example, a business analyst wants to add a data request to an orchestration that returns a customer's credit limit amount. The orchestration has "Customer Number" defined as an input. To configure the data request, the business analyst:

- ❑ Sets up filter criteria with a condition that filters on Address Number.
- ❑ Selects the Credit Limit and Alpha Name fields for the return data.
- ❑ Adds the data request to the orchestration, mapping the Customer Number input in the orchestration to the Address Number in the data request.

When the data request is added to the orchestration, the Credit Limit and Alpha Name fields automatically become additional inputs to the orchestration, which the business analyst can then map to subsequent orchestration steps.

> **Tutorial:**

[Click here to view a recording of this feature.](#)

To configure a data request to return field data:

1. Create a data request as described in [Creating a Service Request](#).
2. In the Available Actions area, enter the name of the table or business view in the Table/View Name field.

If you do not know the table or business view name, but you know the application that uses the table or business view, you can use the JD Edwards EnterpriseOne Cross Reference Facility (P980011) to identify the table or business view associated with the application. From the Cross Reference form, click the **Interactive Applications** tab, then click **Business Views Used by an Interactive Application or Tables Used by an Interactive Application**. If you do not have access to the Cross Reference Facility, ask an administrator to look up this information for you.

3. Click the **Load** button.

The Orchestrator Studio loads all fields from the table or business view into the grid.

Note: You can use the Data Set Variable Name field to configure the data request to return a data set. See [Retrieving and Passing Data Sets in an Orchestration \(Orchestrator Studio 6.0.x\)](#) for more information.

4. Define the filtering criteria for the data request:
 - a. In the Fields grid on the left, click the "**Filter**" icon next to the field or fields that contain data that you want to filter on.

The Orchestrator Studio displays each field in the Filter Criteria area on the right.
 - b. For each field in the Conditions box, select the appropriate operand and in the adjacent field, enter a hard coded value or a variable.

A variable appears in the field by default. If you modify the variable name, make sure the syntax includes the \$ sign and brackets, for example:

```
 ${Address Number 1}
```
 - c. Select the **Match All** or **Match Any** option to determine how the conditions are applied.
- You can also click the **Options** button, and from the Query drop-down list, select a predefined query to use for the filtering criteria. You can use a query instead of or in combination with the filtering criteria defined in a data request. The queries that you can see in this list are based on UDO security permissions.
5. Specify the data you want returned:

- a. In the Fields grid, click the "**Return**" icon next to each field for which you want data returned.

Each field appears in the "Return Fields and Variable Names" on the right.

- b. (Optional) You can use a variable for the return by entering a name for the variable in the adjacent blank field.

Note: Return variables do not need the \${ } notation. This notation is only necessary for input variables to distinguish between a variable and a hard coded value.

When you add a data request to an orchestration, these variables are automatically added to the orchestration as inputs, which you can use to map return data to subsequent orchestration steps.

6. (Optional) For the return data, determine the order by which you want the data returned:
 - a. In the grid on the left, select the **Order By** icon next to any field that was selected for the return data.
The Orchestrator Studio adds the field name to the Order By area on the right.
 - b. In the drop-down list next to the field name, select **Ascending** or **Descending**.
7. (Optional) Click the **Options** button to configure settings that control how the AIS Server processes a data request at runtime. See [Configuring Form Request and Data Request Processing](#).

A.4.5.2 Configuring a Data Request with Data Aggregation

> **Tutorial:**

[Click here to view a recording of this feature.](#)

To configure a data request to return aggregate amounts:

1. Create a data request as described in [Creating a Service Request](#).
2. In the Available Actions area, enter the name of the table or business view in the Table/View Name field.

If you do not know the table or business view name, click the **Get View From Form** button and enter the application and form ID to load all fields from the primary business view associated with the form.

3. Click the **Load** button.

The Orchestrator Studio loads all fields from the table or business view into the grid.

4. Slide the **Aggregation** toggle to the right.

This enables the aggregation features in the Fields grid.

Note: You can use the Data Set Variable Name field to configure the data request to return a data set. See [Retrieving and Passing Data Sets in an Orchestration \(Orchestrator Studio 6.0.x\)](#) for more information.

5. Click the "**Filter**" icon next to the fields that contain the data that you want to filter on.

The Orchestrator Studio displays each field in the Conditions area on the right.

6. Set up conditions for filtering data:
 - a. For each field in the Conditions box, select the appropriate operand and in the adjacent field, enter a hard coded value or a variable.

A variable appears in the field by default. You can modify the variable name, but you must use the \$ sign and brackets in the syntax, for example:

`${Address Number 1}`

- b. Select the **Match All** or **Match Any** option to determine how the conditions are applied.

You can also select the Options button, and from the Query drop-down list, select a predefined query to use for the filtering criteria. You can use a query instead of or in combination with the filtering criteria defined in a data request. The queries that you can see in this list are based on UDO security permissions.

7. Click the "Aggregate" icon next to the fields that contain the data for the aggregation.
8. In the pop-up window, select the type of aggregate that you want to perform. The aggregate options displayed depend on whether the field contains a numeric value or a string.
9. (Optional) In the Aggregations and Variable Names section, click the **Include Count** check box if you want the response to include a count of the records returned. To use the returned count in a subsequent orchestration step, enter a variable name in the adjacent field.
10. (Optional) Use the following features in the Fields grid to further define the data aggregation:

- **"Having" icon.** Click this icon next to a field for which you want to provide additional filtering criteria on an aggregation.

The Data Request design page displays the field in the Having section on the right.

- a. Click the drop-down list next to the field and select the operand.
- b. In the adjacent field, enter a hard coded value or variable.
- **"Group By" icon.** Click this icon next to any field that you want the aggregate results grouped by. In the "Group By and Variable Name" section on the right, you can enter a variable name.

Using "Group By" may result in multiple rows being returned, one for each unique combination of a group. For example, you might return total orders for an entire year for customers, and group the results by customer. In this case, the data request response will return a unique row for each customer with the customer's total orders for the year.

Note: Variables are only supported in the first row of a response. If a query contains multiple rows, only the values from the first row will be available as variables.

- **"Order By" icon.** For any fields used in an aggregate or "group by," click this icon to arrange return data in ascending or descending order.

11. (Optional) Click the **Options** button to configure settings that control how the AIS Server processes a data request at runtime. See [Configuring Form Request and Data Request Processing](#).

A.4.5.3 Viewing and Copying JSON Code of a Data Request

After configuring a data request, you can click the **Preview** button to view and copy the JSON code for a data request. Developers can use this code to develop AIS client applications that can make data request calls directly to the AIS Server rather than through an orchestration. See the *JD Edwards EnterpriseOne Application Interface Services Client Java API Developer's Guide* for more information.

A.4.6 Configuring a Message Request

Add a message request to an orchestration to send emails to external email systems or the EnterpriseOne Work Center email system. The following EnterpriseOne Work Center and workflow features are supported in a message request:

- Workflow distribution lists to send messages to a predefined group of EnterpriseOne users.
- Message templates from the data dictionary that contain boilerplate text and values related to a particular business process.
- Shortcuts to EnterpriseOne applications.

For more information about these workflow features, see the *JD Edwards EnterpriseOne Tools Workflow Tools Guide*.

In a message request, you can include variables to pass data from an orchestration input into a message request. You can include variables in the recipient fields, the subject and body, a message template, and a shortcut.

> Tutorial:

[Click here to view a recording of this feature.](#)

To configure a message request:

1. Create and name the message request as described in [Creating a Service Request](#).
2. To enter recipients (To, Cc, or Bcc), select a recipient type:
 - Select **Address Book** to send a message to a single EnterpriseOne user. Enter the address book number of the EnterpriseOne user.
 - Select **Contact** to send a message to an individual in a user's contact list. Enter the address book number of a user and then the number of the contact.
 - Select **Distribution List** to send a message to the members of an EnterpriseOne distribution list. Enter the address book number of the list in the Parent Address Number field and select its structure type. For more information about structure types, see "Structure Types" in the *JD Edwards EnterpriseOne Tools Workflow Tools Guide*.
 - Select **Email** to enter an email address for the recipient.
3. In the Subject and body fields, enter text, variables, or a combination of both. To insert variables, see step 6.
4. To include boilerplate text from a message template in the data dictionary:
 - a. Expand the Data Dictionary Text section.
 - b. In the Data Item field, enter the name of the message template data item and click **Load**.

- c. If the message template contains variables, use the grid in the right to override the variables with text substitution.
- 5. To include a shortcut to an application:
 - a. Expand the JD Edward EnterpriseOne Shortcut section.
 - b. Complete the Application, Form, and Version fields to specify the form that you want the shortcut to launch.
 - c. Click **Load**. Starting with Orchestrator Studio 6.1.0, this button was removed as it is no longer necessary to "load" the shortcut.
 - d. In the grid, you can use variables to pass in data to the application when the application is launched from the shortcut.
- 6. To include variables in the recipient types, subject, body, message template text, or shortcut:
 - a. Right-click in a field and click **Insert Variable**.
 - b. In the variable that appears, rename variable as appropriate, but make sure the syntax includes the \$ sign and brackets, for example:

`${creditmanager}`

A.4.7 Configuring a Connector Service Request

This section contains the following topics:

- ❑ [Section A.4.7.1, "Understanding Connector Service Requests"](#)
- ❑ [Section A.4.7.2, "Before You Begin"](#)
- ❑ [Section A.4.7.3, "Configuring a Connector Service Request to Invoke an Orchestration or Notification"](#)
- ❑ [Section A.4.7.4, "Configuring a REST Connector to Invoke a REST Service"](#)
- ❑ [Section A.4.7.5, "Configuring a REST Connector to Transfer Files to a REST Service \(Orchestrator Studio 6.1.0\)"](#)
- ❑ [Section A.4.7.6, "Configuring a Database Connector \(Orchestrator Studio 6.0.1\)"](#)

- Section A.4.7.7, "Configuring a Connector to Transfer Files Using File Transfer Protocol (FTP) (Orchestrator Studio 6.1.0)"

A.4.7.1 Understanding Connector Service Requests

In the Orchestrator Studio, you can create a connector service request to enable an orchestration to:

- Invoke another orchestration or invoke a notification (Orchestrator Studio 6.0.1) either on your local system or on an AIS Server on another EnterpriseOne system in a multisite operation.
- Invoke a REST service. This enables outbound REST calls to external systems as an orchestration step. For example, an orchestration could make a REST call to a Cloud service and use the data in the response in subsequent orchestration steps.
- Connect to a database (Orchestrator Studio 6.0.1). This enables orchestrations to read from and write to non-JD Edwards databases using standard SQL protocol. The database must support JDBC. A database connector enables external databases to be the source of input for orchestrations. It also allows data that is not appropriate for transaction tables to be stored for analysis, archive, or integration purposes.
- Retrieve a file or send a file to a known location using FTP or SFTP protocols. (Orchestrator Studio 6.1.0)
- Send a file to a known location using a REST protocol. (Orchestrator Studio 6.1.0)

A.4.7.2 Before You Begin

A connector service request requires a connection soft coding record, which provides the connection that the connector uses to access resources on an external system. Ask your system administrator to create a connection. See [Creating Connection Soft Coding Records for Connector Service Requests](#) for details.

A.4.7.3 Configuring a Connector Service Request to Invoke an Orchestration or Notification

> Tutorial:

[Click here to view a recording of this feature.](#)

1. Create and name the connector service request as described in [Creating a Service Request](#).
2. On the Connector design page, click in the **Connection** field and select a connection under the Orchestrator category.

After you select a connection, the Orchestrator Studio displays the URL to the AIS Server next to the Connection field.

3. Click inside the Orchestration/Notification field and select an orchestration or notification from the drop-down list.

The drop-down list displays a list of available orchestrations followed by a list of available notifications. The orchestrations and notifications are further categorized by UDO status: Personal, Pending Approval, Reserved, or Shared.

Any inputs defined in the called orchestration or notification appear in the Orchestration Input column, with variable names automatically generated in the adjacent Input column.

4. In the Input column, you can modify the variable names as desired or map a hard coded value by entering a value (without the \${ } notation).

5. For an orchestration connector, use the adjacent Output grid to specify values you want returned from the called orchestration. These values must be defined as outputs in the called orchestration. Optionally, enter a variable for the value to make it an orchestration input, which gives you the option to map the value to a subsequent step in an orchestration.

A.4.7.4 Configuring a REST Connector to Invoke a REST Service

Configure a REST connector to invoke a REST service. In the REST connector, you specify an HTTP method and then provide the following additional details required for the connector to complete the request:

- ❑ The path to a particular endpoint in the REST service.
- ❑ Parameters, if required by the endpoint.
- ❑ Key-value pairs for the HTTP header.

In the Orchestrator Studio, you can test the connector and view the JSON response of the REST service call. You can also use Groovy scripting to refine the data in the connector output. For example, you can configure a Groovy script to refine the contents of the REST service response so that the connector output contains only the data required by the consuming device or program.

To configure a connector to invoke a REST service:

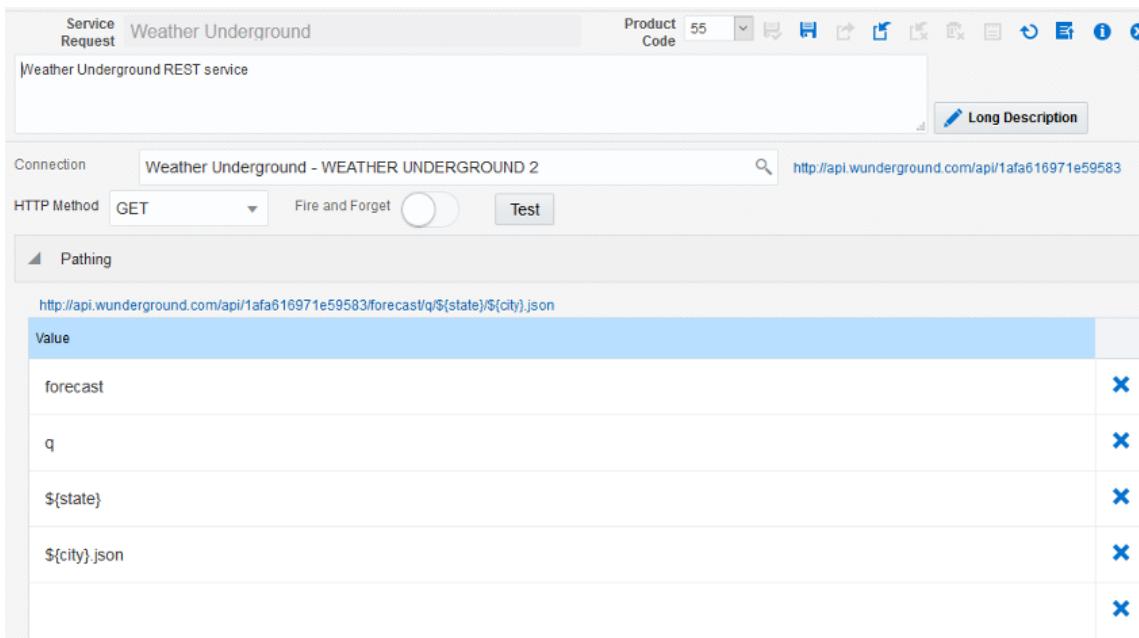
1. Create and name the connector service request as described in [Creating a Service Request](#).
2. On the Connector design page, click the **Connection** field and select a connection under the REST category.

The Orchestrator Studio displays the URL to the REST service available through the connection.

3. Click the **HTTP Method** drop-down list and select the appropriate method.
4. Slide the **Fire and Forget** toggle to the right if you want the orchestration to execute without waiting for a response. If Fire and Forget is enabled, the Output section is hidden because the REST service response is not processed.
5. Expand the Pathing section and use the Value grid to build the path to a REST service endpoint.

Each value you enter in the Value grid is appended to the path, separated by a forward slash (/). You can use variables in the path by adding a value inside \${ }. Click the **Delete** button at the end of a row to delete any values from the path.

The following image shows an example of a path to an endpoint that includes variables.



6. If the REST service takes parameters, use the Parameters section to append parameters to the endpoint.

Parameters are appended to the endpoint after a "?" and are separated by a "&". You can include variables by specifying a value inside \${}.

7. In the Headers section, enter key-value pairs for the header in the Key and Value columns. You can also choose any known values from the drop-down menu in each column.
8. In the Body section (which is not used or displayed if the HTTP method is GET or TRACE), if the HTTP method requires it, specify a payload to send to the REST service.
9. In the Output section, use the following sections to identify the outputs:

Note: The outputs are based on the JSON response of the REST service call. You can see the response after running a successful test of the REST service using the Test button.

- Manipulate Output (advanced). Use Groovy scripting to manipulate the output of the REST call. For example, if you only need certain data in the output, you can use Groovy scripting to refine the data displayed in the output. See [Groovy Template for Manipulating Output from a REST Connector Response](#) for more information.
- Output grid. If the value you want is nested in JSON objects, you can get to the value by separating each JSON object name by a period (.). If the value you want is in an array, you can access it by adding the proper index of the array inside [].

The following image shows an example of outputs defined in the Output section:

Output	Variable Name	
forecast.simpleforecast.forecastday[0].qpf_allday.in	QPFDay0	X
forecast.simpleforecast.forecastday[1].qpf_allday.in	QPFDay1	X
forecast.simpleforecast.forecastday[2].qpf_allday.in	QPFDay2	X

10. To test the connector:
 - a. Click the **Test** button.
 - b. If the connector includes variables, enter values for the variables in the popup box.
If successful, the Orchestrator Studio displays the response.
 - c. Click the **Show Output** button displayed at the end of the response to apply the instructions defined in the Output section, including the Groovy script if specified.
Use the results to validate the path to the output values specified in the Output section.

A.4.7.5 Configuring a REST Connector to Transfer Files to a REST Service (Orchestrator Studio 6.1.0)

You can configure a REST connector to transfer files to a REST service. When added to an orchestration that includes a report service request, a REST connector can transfer the output of the report service request to an external system. Also, if an orchestration includes a custom service request that uses a Groovy script to generate and save a file to the AIS Server, you can create a REST connector to transfer the generated file to an external system.

1. Create and name the connector service request as described in [Creating a Service Request](#).
2. On the Connector design page, click the **Connection** field and select a connection under the REST category.
The Orchestrator Studio displays the URL to the REST service available through the connection.
3. Click the **HTTP Method** drop-down list and select **POST**.
4. Enable the **Fire and Forget** toggle if you want the orchestration to execute without waiting for a response.
If Fire and Forget is enabled, the Output section is hidden because the REST service response is not processed.
5. In the File section of the request, enter the File Part Name, which you can find in the documentation for the REST API you are calling.
6. Select the **Report** or **File** option to specify the type of file you are sending, and then complete the following fields accordingly:
For Report, complete these fields:
 - **Job Number.** Enter `${variablename}` to use a variable for the job number. When you add this connector to an orchestration, you can map the job number of a report returned from a report service request to this variable. Or instead of a variable, you can enter an actual job number.

- **Execution Server.** Enter the server where the report was generated. Enter \${variablename} to use a variable for the server name. When you add this connector to an orchestration, you can map the name of this server from the report service request, which returns the name of this server. Or instead of variable, you can enter the server name.

- **File Type.** Select the file type of the output.

For File, complete these fields:

- **Source File Name.** Enter the name of the file that was created in the custom Groovy service request.
- **Remove File.** Enabled by default, this removes the file from the temporary directory on the AIS Server after the file is transferred.

7. If the REST API being called requires additional information, you can use the table at the bottom of the File section to include additional details.

A.4.7.6 Configuring a Database Connector (Orchestrator Studio 6.0.1)

You must have knowledge of Groovy scripting language to create a database connector in order to route data to a database. The database must support JDBC. The Connector design page provides a Groovy template that you can use as a basis for creating a Groovy script for a database connector.

1. Create and name the connector service request as described in [Creating a Service Request](#).
2. On the Connector design page, click the **Connection** field and select a connection under the Database category.
The Orchestrator Studio displays an edit area that contains a Groovy script template. Use the Find and "Go to Line" fields and Undo and Redo buttons to work with the script.
3. In the Input grid, enter the inputs that you want to pass to the database. You can enter hard coded values or include variable using the \${ } notation, entering the variable name within the brackets.
4. In the Output column, list the fields added to the returnMap in the Groovy script to make those outputs available to the orchestration. Optionally, enter a variable name in the Variable column if you want to make the values available for mapping to a subsequent step in an orchestration.
5. Click **Save**.

A.4.7.7 Configuring a Connector to Transfer Files Using File Transfer Protocol (FTP) (Orchestrator Studio 6.1.0)

You can configure a connector to transfer files using FTP or secure file transfer protocol (SFTP). With this type of connector, referred to as an FTP connector, you can:

- Transfer the output of a report service request to an FTP server.
The connector can transfer the output of a standard EnterpriseOne report or an embedded Oracle BI Publisher report.
- Transfer other types of files from an FTP server to a temporary directory on the AIS Server.
An administrator must define the temporary directory in the basic configuration settings for the AIS Server. For more information, see [Set Up a Temporary Directory on the AIS Server for File Transfers](#).
- Transfer files in the temporary directory on the AIS Server to an FTP server.

Note: You can also use a REST connector to transfer a report or file to an external location. See [Configuring a REST Connector to Transfer Files to a REST Service \(Orchestrator Studio 6.1.0\)](#)

An FTP connector uses a secure connection created through a soft coding record to receive and send a file from an FTP server.

An FTP connector can transfer only one file at a time.

Configure an FTP Connector to Transfer the Output of a Report Service Request

1. Create and name the connector service request as described in [Creating a Service Request](#).
2. On the Connector design page, click the **Connection** field and select a connection under the FTP category.
If there are no connections available for an FTP connector, ask your system administrator to create one. See [Creating a Soft Coding Record for an FTP Server Connection \(Orchestrator Studio 6.1.0\)](#).
3. Click the **Report** option and complete these fields:

- **Job Number.** By default, this field contains a variable. When you add this connector to an orchestration, you can map the job number of a report returned from a report service request to this variable. You can modify the variable name if desired. Or you can delete the variable (including the special characters) and replace it with an actual job number.
- **Execution Server.** Enter the server where the report was generated. By default, this field contains a variable. When you add this connector to an orchestration, you can map the name of this server from the report service request, which returns the name of this server. You can modify the variable name if desired. Or you can delete the variable (including the special characters) and replace it with a server name.
- **File Type.** Select the type of file that the FTP connector will transfer. The default is PDF for standard EnterpriseOne reports. If transferring an embedded BI Publisher report, then select one of the BI Publisher file types.
- **Path Extension.** Enter the name of the sub-directory on the third-party FTP server where you want the report output sent. The FTP Connection already contains information for the base folder.

4. Click **Save**.

Configure an FTP Connector to Transfer a File from an FTP Server to the AIS Server

1. Create and name the connector service request as described in [Creating a Service Request](#).
2. On the Connector design page, click the **Connection** field and under the FTP category, select a connection to the server from which you want to transfer a file.
If there are no FTP connections available, ask your system administrator to create one. See [Creating a Soft Coding Record for an FTP Server Connection \(Orchestrator Studio 6.1.0\)](#).
3. Select the **File** option.
4. Click the **Type** drop-down list and select **Receive**.
5. In the **Source File Name** field, enter the name of the file that you want to retrieve.

6. In the **Target File Name** field, enter the name of the target file if you want it to be saved with a different name than the source file when saved to the temporary directory on the AIS Server.
7. Click the **Use Temporary File Location** check box to save the file to the temporary directory on the AIS Server.
8. Enable or disable the **Remove File** toggle. Enabled by default, this removes the file from the temporary location on the AIS Server after it is transferred.
9. Click **Save**.

Configure an FTP Connector to Transfer a File from the AIS Server to an External Server

1. Create and name the connector service request as described in [Creating a Service Request](#).
2. On the Connector design page, click the **Connection** field and under the FTP category, select a connection to the server to which you want to transfer the file.
If there are no FTP connections available, ask your system administrator to create one. See [Creating a Soft Coding Record for an FTP Server Connection \(Orchestrator Studio 6.1.0\)](#).
3. Select the **File** option.
4. Click the **Type** drop-down list and select **Send**.
5. In the **Source File Name** field, enter the name of the file that you want to transfer.
6. Click the **Use Temporary File Location** check box to save the file to the temporary directory on the AIS Server.
7. In the **Target File Name** field, enter the name of the target file if you want it to be saved with a different name than the source file when saved to the AIS Server directory.
8. Enable or disable the **Remove File** toggle. Enabled by default, this removes the file from the temporary location on the AIS Server after it is transferred to an external server.
9. Click **Save**.

A.4.8 Configuring a Watchlist Service Request (Orchestrator Studio 6.0.1.0)

Use a Watchlist service request to use the information from Watchlists, such as critical or warning states, threshold levels, and number of records, within an orchestration. For more information about the data you can retrieve from a Watchlist, see *JD Edwards EnterpriseOne Applications One View Watchlists Implementation Guide*.

1. Create and name the Watchlist service request as described in [Creating a Service Request](#).
On the Watchlist page, the grid displays all Watchlists that you have been authorized to access through UDO security in EnterpriseOne.
If you need a Watchlist not available in this list, ask your EnterpriseOne administrator to grant you access to the Watchlist.
2. In the Name column in the grid, click the link to the Watchlist that you want this service request to access.
The Watchlist displays information about the Watchlist and areas for selecting the Watchlist details you want returned. To select a different Watchlist, click **Change Watchlist** to return to the list of Watchlists.
3. In the Outputs area, select the Watchlist data you want returned.

4. In the Advanced Outputs area, select any additional details about the Watchlist that you want returned.
5. Click the **Force Watchlist To Update** check box if you want the service request to refresh the Watchlist data in EnterpriseOne before returning Watchlist data. (Recommended)

A.4.9 Configuring a Report Service Request (Orchestrator Studio 6.1.0.0)

Use a report service request to invoke a batch version of an EnterpriseOne report from an orchestration. You can configure a report service request to use the settings defined for the report in the Batch Versions program. Or you can configure it to override the processing options, data selection, and data sequencing of a report.

You can also create a report service request to invoke an embedded BI Publisher report, and configure it to use the default settings or override the settings.

A report service request can launch EnterpriseOne reports designed with report interconnects. Report interconnects are inputs added to a report at design time, which enable a report to run automatically without user interaction. Typically, report interconnects are used to launch a report from a button or exit on an EnterpriseOne form.

In the Report design page, the "Fire and Forget" option enables the report to run asynchronously. When the Orchestrator executes the orchestration, this option enables the orchestration to continue to the next action or step without waiting for the report to complete.

At runtime, when the Orchestrator processes the report service request, it respects the EnterpriseOne authorization security for the report. The user initiating the orchestration must be authorized to run the batch version of the report as defined by an administrator in the Security Workbench. If the report service request includes data selection or processing option overrides, the user initiating the orchestration must be authorized in the Security Workbench to perform these overrides as well.

To configure a report service request:

1. Create and name the report service request as described in [Creating a Service Request](#).
2. In the Report field, enter the ID of the report that you want the report service request to invoke.

The Orchestrator Studio loads all versions associated with the report in the Version drop-down list.

3. Click the **Version** drop-down list and select a version of the report.

The Orchestrator Studio loads any settings defined for the version in EnterpriseOne, including data selection, data sequencing, processing options, and output options.

4. Enable **Fire and Forget** if you want the report to run asynchronously.

If you do not enable this setting, if added to an orchestration that has steps following the report service request, the remaining steps will not execute until the report finishes.

5. Select one of the following options to determine how to run the report:

▪ **Blind Execution.** Select this option if you want the report service request to invoke the batch version using the settings defined in EnterpriseOne for the report version.

You can review the settings defined for the version by expanding the Data Selection, Data Sequencing, and Processing Options sections.

▪ **Report Interconnects.** If a report is defined with report interconnects, select this option and enter the values you want to pass into the report in the available fields. This option is disabled if no report interconnects were defined for the report.

- **Override Version.** Select this option to override the settings in EnterpriseOne for the version of the report, and then perform these steps:
 - a. For Data Selection and Data Sequencing, expand the sections to modify existing criteria, or click the **Add** button to define new criteria.
 - b. Expand the Processing Options section and change the settings as desired.
 - 6. Click the **Queue Name** drop-down list and select the queue to which the reported is submitted. If you leave this field blank, when this report is invoked, the system uses the default queue defined in the Enterprise Server configuration settings.
 - 7. To modify the print properties defined for the report, click **Output Options** and complete the fields as appropriate.
- For more information about report output options, see "Report Output" in the *JD Edwards EnterpriseOne Tools Report Printing Administration Technologies Guide*.
8. To enable logging for the report, click **Output Options** and then enable the **JDE Log** toggle. Enter a value from 0–6 in the Logging Level field to indicate the level of detail to be captured in the logs.

Any value greater than 0 will force the system to write both the JDE and JDEDEBUG logs. When you select a high value to receive more technical information, you also receive all the information for the lower values. For example, when you enter a value of 3 (object level messages), you also receive information for 2 (section level messages), 1 (informative messages), and 0 (error messages).

A.4.10 Configuring Form Request and Data Request Processing

In the Orchestrator Studio, you can configure settings that control how the AIS Server processes a form request or data request in an orchestration at runtime.

For form requests, these settings are available from the Options icon at the end of the first row of each form listed in the Available Actions grid.

For data requests, these settings are available from the Options button on the Data Request design page.

The settings include:

- **Maximum Records.** 0 is for All Records.
- **Query Name.** From this drop-down list, you can select a predefined query to use for the filtering criteria. You can use a query instead of or in combination with the filtering criteria defined in a data request. The queries that you can see in this list are based on UDO security permissions.
- **Output Type.** Select one of the following format types for the format of the JSON response. Formats include:
 - **Default.** The default format is Version 2 output type for version 1 orchestrations (orchestrations created prior to Orchestrator Studio 5.0.1). The default format is Grid Data output type for version 2 orchestrations (orchestrations created in Orchestrator Studio 5.0.1 and higher).
 - **Grid Data.** Returns grid data in simplified name-value pairs. This is the default output type for version 2 orchestrations (orchestrations created in Orchestrator Studio 5.0.1 and higher).
 - **Version 2.** Returns cell-specific information for each grid row as well as information about each column such as column ID, whether it is visible, editable, and so forth.

This is the default output type for orchestrations created prior to Orchestrator Studio 5.0.1, which are referred to as version 1 orchestrations.

- ❑ **Stop on Warning** check box (form requests only). Click this check box if you want the processing to stop if the invoked EnterpriseOne form produces a "Warning" message. Keep this check box clear if you want processing to continue after a warning message.
- ❑ **Turbo Mode** (form requests only). Use this option to reduce processing time of form requests. See "Using Turbo Mode" in the *JD Edwards EnterpriseOne Application Interface Services Client Java API Developer's Guide* for more information.
- ❑ **Caching: Allow**. Click this check box to enable caching of the service request response. If the "Enable Caching by Default" option is enabled in Server Manager, this parameter is ignored and all responses for Read operations will be cached.
- ❑ **Caching: Force Update**. Click this check box to force the system to fetch the data from the database for the service request. If the "Enable Caching by Default" setting is enabled in Server Manager, then this parameter for the individual service request is ignored.
- ❑ **Caching: Time Stored - Milliseconds**. Enter the amount of time in milliseconds for the service request to use the cache for the response. This setting overrides the value in the "Default Read Cache Time To Live" setting in Server Manager.

A.4.11 Configuring a Custom Service Request with Java (Advanced)

You can create a service request that uses custom Java to execute a custom process or to route data into another database. Before you can create a custom Java service request, you must create the custom Java as described in [Chapter 7, "Creating Custom Java for Orchestrations"](#).

1. After naming the custom service request as described in [Creating a Service Request](#), select the **Java** option.
2. In the **Fully Qualified Class** field, enter the Java class.
This is the custom Java class that you created for the service request.
3. In the first grid, complete the following fields:
 - ❑ **Input**. Enter the name of the field in the class.
 - ❑ **Input Value**. Enter the input variable name. If the attribute is a boolean and you pass in "true", then you could use a default value.
 - ❑ **Default Value**. Enter a default, hard-coded value if there is no mapped value. You can also add a hard-coded value to use if the mapped value returns a null.
4. (Optional) In the second grid, enter a variable name for an output if you want to use the output value in subsequent steps in the orchestration or to another orchestration.
5. If you make a mistake, click the **Restore Custom Java** button to return the custom Java to its last saved state.
6. Click the **Save** button.

The Orchestrator Studio saves the custom Java request. You can then access the orchestration in the Orchestration design page and add this custom Java service request as a step in the orchestration.

A.4.12 Configuring a Custom Service Request with Groovy (Advanced)

You can create a service request that uses Groovy scripting to execute a custom process or to route data into another database. To use Groovy for a service request, the Orchestrator Studio provides an edit area that contains a sample Groovy script with instructions on how to modify

the script. You can use the Find and "Go to Line" fields and Undo and Redo buttons to work with the script. Chapter 8, "Using Apache Groovy for Custom Service Requests, Rules, and Manipulating Output (Orchestrator Studio 5.1.0 and Higher)" provides information on how to work with the sample Groovy script.

To configure a customer service request with Groovy:

1. After naming the custom service request as described in [Creating a Service Request](#), select the **Groovy** option.
2. Configure the script to perform the desired action.
3. In the "Input" grid, enter the names of the inputs.

The inputs will be placed in the inputMap of the "main" function and can be retrieved in the script by following the commented out example code.

4. Click the **Load Outputs** button.

This reads the script and adds any values added to the returnMap in the script to the Output grid.

5. (Optional) In the "Output" grid, enter a variable name for an output if you want to use the output value in a subsequent orchestration step.

When you add this service request to an orchestration, this name appears in the orchestration inputs grid, which makes the returned value available for mapping to subsequent steps in the orchestration.

Note: Even if no variables are used, all defined outputs are available for mapping using the Orchestration Output feature. See [Working with Orchestration Output](#) for more information.

6. To test the script:

- a. Enter a value in the Test Value column for one or more inputs.
- b. Click the **Test** button.

Orchestrator Studio executes the script using the inputs you entered. If successful, it populates the results in the Test Output column of the Output grid.

If you included orchAttr.writeWarn or orchAttr.writeDebug statements in the script, a Logging popup displays after execution. At runtime, log statements are included in the AIS Server log, which can be used for debugging script issues.

The following example shows the Logging popup, which you can use to verify the values passed to the inputs:

• Groovy ○ Java

Find Go to Line

Logging

```
55
56
57 24 May 2017 09:54:44.239 [WARN ] DEMO - [CUSTOM] Custom SR Determine Irrigation Operation START
58 24 May 2017 09:54:44.239 [WARN ] DEMO - [CUSTOM] today Wed May 24 09:54:44 MST 2017
59 24 May 2017 09:54:44.239 [WARN ] DEMO - [CUSTOM] after30Mins Wed May 24 10:24:44 MST 2017
60 } now millis 1495644884239
61 } after 30 millis 1495646684239
62 24 May 2017 09:54:44.240 [WARN ] DEMO - [CUSTOM] averagePrecip 2
63 24 May 2017 09:54:44.240 [WARN ] DEMO - [CUSTOM] averageLow
64 24 May 2017 09:54:44.240 [WARN ] DEMO - [CUSTOM] averageHigh
65 24 May 2017 09:54:44.240 [WARN ] DEMO - [CUSTOM] soilMoisture
66 24 May 2017 09:54:44.240 [WARN ] DEMO - [CUSTOM] qpfDay0
67 24 May 2017 09:54:44.240 [WARN ] DEMO - [CUSTOM] qpfDay1
68 orch Close
```

Load Outputs Test

Input	Test Value
averagePrecip	2
averageLow	
averageHigh	
soilMoisture	

7. Click the **Save** button.

The Orchestrator Studio saves the custom Groovy service request. You can then access the orchestration in the Orchestration design page and add this custom Groovy service request as a step in the orchestration.

A.5 Creating Rules

This section contains the following topics:

- ☒ Section A.5.1, "Understanding Rules"
 - ☒ Section A.5.2, "Creating a Rule"
 - ☒ Section A.5.3, "Creating a Custom Rule with Java (Advanced)"
 - ☒ Section A.5.4, "Creating a Custom Rule with Groovy (Advanced)"

A.5.1 Understanding Rules

A rule contains conditional logic that the Orchestrator uses to evaluate conditions, such as true or false conditions that determine how the orchestration processes the incoming data. You can define a rule with a list of conditions or you can define a more complex rule using Groovy or a custom Java class.

An orchestration rule with conditions functions similar to an EnterpriseOne query in that each rule has:

- ❑ A "Match Any" or "Match All" setting.
 - ❑ One or more conditions defined, each being a binary operation on two values.

After creating a rule and adding it to an orchestration, you use the Action column in the Orchestration design page to determine the orchestration step that is invoked when the conditions in the rule are met. See [Defining the Actions Between a Rule and Dependent Components](#) for more information.

A.5.2 Creating a Rule

Create a rule to define conditions for an orchestration.

> Tutorial:

[Click here to view a recording of this feature.](#)

To create a rule:

1. On the Orchestrator Home page, click the **Rules** icon. And then on the Rules design page, click the **New Rule** button.

Alternatively, access the Orchestrations design page and add a Rule step to an orchestration. At the end of row with the Rule step, click **Edit** (pencil icon) and select **Rule** from the pop-up box.

2. On the Rules design page, enter a name for the rule in the **Rule** field. Do **NOT** include special characters in the name.

3. Click the **Product Code** drop-down list to select a product code to associate with the rule.

This gives an administrator the option to manage UDO security for orchestration components by product code.

4. In the space provided, enter a short description with a maximum of 200 characters. This description appears below the rule name in the component list.

5. Click the **Edit Long Description** button to add a long description to provide more detail about the purpose of the component.

6. Select the Match Value drop-down menu and select one of the following values:

- ❑ **Match All.** Select this option if all conditions in the rule must be met.
- ❑ **Match Any.** Select this option if any conditions in the rule can be met.

7. In the first row in the grid, complete the following columns to add a condition:

- ❑ **Rule Type.** Click the drop-down menu and select String, Numeric, or Date depending on the format of the input.

- ❑ **Value 1.** Enter a variable for the input name.

- ❑ **Operator.** In the drop-down menu, select from the list of operands which include: >, <, >=, <=, =, !=, startsWith, endWith, contains, between, inList.

- ❑ **Literal.** Click this check box to indicate a literal value.

- ❑ **Value 2.** If you selected the Literal check box, manually enter a value.

- ❑ **Literal Value Type.** If you selected the Literal check box, click the drop-down menu and select the format of the literal value: string, numeric, data.

8. Add additional conditions to the rule as needed. If you add a condition by mistake, you can delete it by clicking the **Remove** button at the end of the row with the condition.

9. Click the **Save** button.

The first time a new rule is saved, it is saved as a "Personal" UDO. Thereafter, you can use the UDO buttons described in the [User Defined Object \(UDO\) Features in the Orchestrator Studio](#) section to move the rule to the appropriate status.

After adding a rule and a service request to an orchestration, in the Orchestration design page, you must define the action for the rule to invoke the service request. See [Defining the Actions Between a Rule and Dependent Components](#) for more information.

A.5.3 Creating a Custom Rule with Java (Advanced)

You can create complex rule using custom Java. Before you add a custom Java rule in the Orchestrator Studio, you must create a custom Java class to use for the rule as described in Chapter 7, "Creating Custom Java for Orchestrations".

To create a custom Java rule:

1. Click the **Rules** icon on the Orchestrator Studio Home page, and on the Rules page, click the **New Custom** button.

Alternatively, access the Orchestrations design page and add a Rule step to an orchestration. At the end of row with the Rule step, click **Edit** (pencil icon) and select **Custom** from the pop-up box.

2. On the Rules design page, select the **Java** radio button.
3. In the Rule field, enter a name for the custom rule. Do **NOT** include special characters in the name.
4. Click the **Product Code** drop-down list to select a product code to associate with the rule.

This gives an administrator the option to manage UDO security for orchestration components by product code.

5. In the space provided, enter a short description with a maximum of 200 characters. This description appears below the rule name in the component list.
6. Click the **Edit Long Description** button to add a long description to provide more detail about the purpose of the component.
7. In the Fully Qualified Class field, enter the fully qualified path to the Java class, which includes the name of the Java class.

This is the custom Java class that you created to use for the rule.

8. Complete the following fields in the grid:
 - **Attribute.** Enter the name of the field in the class.
 - **Input Value.** Enter a variable for the input name. If the attribute is a boolean and you pass in "true", then you could enter a hard-coded default value.
 - **Default Value.** Enter a hard-coded value if there is no mapped value. You can also add a hard-coded value to use if the mapped value returns a null.
9. If you make a mistake while configuring any of the fields, you can click the **Restore Rule** button which refreshes the custom Java rule to its last saved state.

10. Click the **Save** button.

A.5.4 Creating a Custom Rule with Groovy (Advanced)

Use Groovy to create a custom rule when conditions in a standard rule will not suffice.

To create a custom rule with Groovy:

1. Click the **Rules** icon on the Orchestrator Studio Home page, and on the Rules page, click the **New Custom** button.

Alternatively, access the Orchestrations design page and add a Rule step to an orchestration. At the end of row with the Rule step, click **Edit** (pencil icon) and select **Custom** from the pop-up box.

2. On the Rules design page, enter a name for the custom rule in the **Rule** field. Do **NOT** include special characters in the name.

3. Click the **Product Code** drop-down list to select a product code to associate with the rule. This gives an administrator the option to manage UDO security for orchestration components by product code.
4. In the space provided, enter a short description with a maximum of 200 characters. This description appears below the rule name in the component list.
5. Click the **Edit Long Description** button to add a long description to provide more detail about the purpose of the component.
6. Select the **Groovy** radio button.

The Orchestrator Studio displays an edit area that contains a sample groovy script with instructions on how to work with the script. Use the Find and "Go to Line" fields and Undo and Redo buttons to help edit the script. See [Chapter 8, "Using Apache Groovy for Custom Service Requests, Rules, and Manipulating Output \(Orchestrator Studio 5.1.0 and Higher\)"](#) for more information about the sample Groovy script.

7. Configure the script to perform the desired action.
8. In the "Input" grid, enter the names of the inputs.
9. To test the script:
 - a. Enter a value in the Test Value column for one or more inputs.
 - b. Click the **Test** button.

If you included orchAttr.writeWarn or orchAttr.writeDebug statements in the script, a Logging popup displays after execution. At runtime, log statements are included in the AIS Server log, which can be used for debugging script issues.

A Logging dialog box displays the test results which you can use to verify the values passed to the inputs, as shown in the following example:

The screenshot shows the Orchestrator Studio interface with the Groovy tab selected. At the top, there's a search bar with 'Find' and 'Go to Line' buttons. Below is a code editor with the following Groovy script:

```

9
10    orchAttr.writeWarn("averagePrecip " + (String)inputMap.get("averagePrecip"));
11    orchAttr.writeWarn("averageLow " + (String)inputMap.get("averageLow"));
12    orchAttr.writeWarn("averageHigh " + (String)inputMap.get("averageHigh"));
13    orchAttr.writeWarn("soilMoisture " + (String)inputMap.get("soilMoisture"));

```

Below the code editor is a 'Logging' section. It displays log entries from May 25, 2017, at 11:33:23.883. The log entries are:

- [25 May 2017 11:33:23.883 [WARN] JDE - [CUSTOM] START Vineyard History Rule
- [25 May 2017 11:33:23.883 [WARN] JDE - [CUSTOM] averagePrecip .5
- [25 May 2017 11:33:23.883 [WARN] JDE - [CUSTOM] averageLow 44
- [25 May 2017 11:33:23.883 [WARN] JDE - [CUSTOM] averageHigh 69.6
- [25 May 2017 11:33:23.883 [WARN] JDE - [CUSTOM] soilMoisture .5
- [25 May 2017 11:33:23.883 [WARN] JDE - [CUSTOM] qpfDay0 0.06
- [25 May 2017 11:33:23.883 [WARN] JDE - [CUSTOM] qpfDay1 0.97
- [25 May 2017 11:33:23.883 [WARN] JDE - [CUSTOM] qpfDay2 .5

A red box highlights the log entries for 'averagePrecip', 'averageLow', 'averageHigh', 'soilMoisture', 'qpfDay0', 'qpfDay1', and 'qpfDay2'. Below the logging section is a 'Close' button.

At the bottom, there's a 'Test' button with 'Result: true' and an 'Input' table. The 'Input' table has columns 'Input' and 'Test Value'. The rows are:

Input	Test Value
averagePrecip	.5
averageLow	44
averageHigh	69.6

A red box highlights the 'averagePrecip' and 'averageLow' rows in the table.

10. Click Save.

A.6 Creating Cross References

This section contains the following topics:

- ¤ [Section A.6.1, "Understanding Cross References"](#)
- ¤ [Section A.6.2, "Creating a Cross Reference"](#)

A.6.1 Understanding Cross References

The Orchestrator uses a cross reference component to map incoming data (third-party data) to an EnterpriseOne value. The EnterpriseOne value identified in the cross reference is considered the output of the cross reference. The output from the cross reference becomes the data passed to another orchestration step.

For each cross reference that you create in the Orchestrator Studio, you have to create a cross reference record in P952000 in EnterpriseOne. When defining cross reference records for orchestrations in P952000, you must use the "AIS" cross reference type. For more information on how to create these cross reference records in P952000, see [Chapter 5, "Setting Up Cross References and White Lists in EnterpriseOne \(P952000\)"](#) in this guide.

Note: If a cross reference lookup fails in an orchestration, the orchestration is terminated.

A.6.2 Creating a Cross Reference

You must define the orchestration inputs before you can create a cross reference. See [Adding Inputs to an Orchestration](#) for more information.

To create a cross reference:

1. Access the Cross Reference design page:

- ¤ On the Orchestrator Home page, click the **Cross References** icon. And then on the Cross References design page, click the **New Cross Reference** button.
- OR
- ¤ After adding a Cross Reference step to the grid in the Orchestration design page, click the **Edit** button next to the step.

The Orchestrator Studio displays the Cross Reference design page.

2. In the Cross Reference field, enter a name for the cross reference. Do **NOT** include special characters in the name.
3. In the space provided, enter a short description with a maximum of 200 characters. This description appears below the cross reference name in the component list.
4. Click the **Edit Long Description** button to add a long description to provide more detail about the purpose of the component.
5. In the Object Type field, enter a name for the object type.

This is the object type used to categorize the orchestration cross references in EnterpriseOne. See "Adding Orchestration Cross References" in the *JD Edwards EnterpriseOne Tools Interoperability Guide* for more information.

6. Click the **Product Code** drop-down list to select a product code to associate with the cross reference.

This gives an administrator the option to manage UDO security for orchestration components by product code.

7. Complete the Input Key and Output Key columns to map the inputs to EnterpriseOne fields:
 - ❑ **Input Key.** The inputs can be one or many values. The inputs must correspond to the value or pipe (|) delimited values in the Third Party Value column in P952000. See Chapter 5, "Setting Up Cross References and White Lists in EnterpriseOne (P952000)" for more information.
 - ❑ **Output Key.** The outputs can be one or many values. The outputs must correspond to the value or pipe (|) delimited values in the EOne Value column in P952000. See Chapter 5, "Setting Up Cross References and White Lists in EnterpriseOne (P952000)" for more information.
8. If you enter any values by mistake, you can click the **X** button next to the field to delete the entry.
9. Click the **Save** button, which saves the white list as a "Personal" UDO.

The first time a new cross reference is saved, it is saved as a "Personal" UDO. Thereafter, you can use the UDO buttons described in the [User Defined Object \(UDO\) Features in the Orchestrator Studio](#) section to move the rule to the appropriate status.

The output values in the cross reference are now available for mapping in subsequent orchestration steps.

A.7 Creating White Lists

This section contains the following topics:

- ❑ [Section A.7.1, "Understanding White List"](#)
- ❑ [Section A.7.2, "Creating a White List"](#)

A.7.1 Understanding White List

A white list contains a list of IDs permitted in the Orchestrator. By adding a white list to an orchestration, only inputs with IDs listed in the white list are permitted for processing by the Orchestrator. You add a white list component as a step in the orchestration.

In addition to specifying the permitted IDs in the white list, you must also specify the white list IDs in P952000 in EnterpriseOne. This is the same application that you use to set up and store orchestration cross references. As with cross references, use the "AIS" cross reference type in P952000 for a white list. In P952000, the Third Party App ID should have a value of WHITELIST. The EnterpriseOne value is not used for white lists and will default to NA.

If a white list lookup fails in an orchestration, the orchestration is terminated.

A.7.2 Creating a White List

1. Access the White List design page:

- ❑ On the Orchestrator Home page, click the **White Lists** icon. And then on the White Lists design page, click the **New White List** button.
OR
- ❑ After adding a White List step to the grid in the Orchestration design page, click the **Edit** button next to the step.

The Orchestrator Studio displays the White Lists design page.

2. In the White Lists design page, click the **New White List** button.

3. In the White List, enter a name for the white list. Do **NOT** include special characters in the name.
4. In the space provided, enter a short description with a maximum of 200 characters. This description appears below the white list name in the component list.
5. Click the **Edit Long Description** button to add a long description to provide more detail about the purpose of the component.
6. In the Object Type field, enter a name for the object type.

The value you enter in the Object Type field must match a cross reference object type in the EnterpriseOne Business Services Cross Reference application (P952000).

The cross reference object type is a named group of records in P952000. For example, you may have thousands of records in P952000. You can use cross reference object types, for example "Equipment" or "Alert_Notification_Recipients" to group cross reference records into manageable categories. You define the cross reference object types as needed. See [Chapter 5, "Setting Up Cross References and White Lists in EnterpriseOne \(P952000\)"](#) for more information.

7. Click the **Product Code** drop-down list to select a product code to associate with the white list.

This gives an administrator the option to manage UDO security for orchestration components by product code.

8. In the Input Key column, enter the input that you want to add as a permitted input.
9. Click the **Save** button.

The first time a new white list is saved, it is saved as a "Personal" UDO. Thereafter, you can use the UDO buttons described in the [User Defined Object \(UDO\) Features in the Orchestrator Studio](#) section to move the white list to the appropriate status.

A.8 Creating Orchestration

This section contains the following topics:

- ❑ [Section A.8.1, "Understanding Orchestration"](#)
- ❑ [Section A.8.2, "Creating an Orchestration"](#)
- ❑ [Section A.8.3, "Adding Inputs to an Orchestration"](#)
- ❑ [Section A.8.4, "Adding Steps to an Orchestration"](#)
- ❑ [Section A.8.5, "Mapping Orchestration Inputs"](#)
- ❑ [Section A.8.6, "Retrieving and Passing Data Sets in an Orchestration \(Orchestrator Studio 6.0.x\)"](#)
- ❑ [Section A.8.7, "Working with Orchestration Output"](#)

A.8.1 Understanding Orchestration

Creating an orchestration in the Orchestrator Studio involves:

- ❑ Naming the orchestration and specifying the input format for the orchestration.

The input format can be JDE Standard, Oracle Cloud IoT, or Generic. See [Supported Input Message Formats](#) for more information about which input format to use.

- ❑ Adding inputs to the orchestration.

The inputs define the data passed from the device to the orchestration. This may include an ID that identifies the device as well as other data that the orchestration will receive from a device, such as temperature, date, or any other data you want to capture. Each input has a name and a type which can be string, numeric, or various date formats.

- Adding steps to the orchestration.

Each step is a component of the orchestration: a service request, rule, cross reference, or white list. At a minimum, an orchestration requires only a service request step, which provides the actions or the instructions to perform a particular task in EnterpriseOne.

- Configuring transformations.

You use transformations to map orchestration inputs to inputs defined in each orchestration step, such as inputs in a rule, cross reference, white list, or service request component.

Important: Remember that when you are ready to "request to publish" an orchestration, you need to make sure that you also request to publish all components associated with the orchestration. This enables an administrator to save all required components to the proper directory on the AIS Server for processing by the Orchestrator. If a component is missing, the orchestration process will end in error.

A.8.2 Creating an Orchestration

> **Tutorial:**

See the following tutorials on how to create an orchestration:

[Creating and Testing an Orchestration](#)

[Creating an Orchestration with Multiple Components](#)

To create an orchestration:

1. On the Orchestrator Studio Home page, click the **Orchestrations** icon.
2. On the Orchestrations page, click the **New Orchestration** button.
3. On the Orchestration design page, enter a unique name for the orchestration in the Orchestration field. Do **NOT** include special characters in the name.

All orchestrations created in Orchestrator Studio 5.0.1 or higher are saved as version 2 orchestrations. You cannot change this value in the Orchestrator Version drop-down list. If you want to update a version 1 orchestration created in a previous version of the Orchestrator Studio, see [Updating Version 1 Orchestrations to Version 2 Orchestrations](#).

Note: When you save an orchestration, the orchestration name is used to define an endpoint on the AIS Server. The endpoint URL is:

`http://<server>:<port>/jderest/orchestrator/<orchestrationname>`

To invoke this orchestration, third-party applications or devices use a post operation to this url, where <orchestrationname> is the name of the orchestration.

-
4. Click the **Product Code** drop-down list to select a product code to associate with the orchestration.

This gives an administrator the option to manage UDO security for orchestration components by product code.

5. In the space provided, enter a short description with a maximum of 200 characters. This description appears below the orchestration name in the component list.

6. Click the **Edit Long Description** button to provide more detail about the component.

Use this field to describe the purpose of the orchestration and any details that differentiate the orchestration from other orchestrations that you create.

7. Click the **Input Format** drop-down menu and select the appropriate format:

JDE Standard (default)

Oracle Cloud IoT

Generic

See [Supported Input Message Formats](#) for more information about which input format to use.

8. At this point, you can click **Save** before defining inputs or steps for the orchestration.

The Orchestrator Studio saves the orchestration as a "Personal" UDO. Next, add inputs to the orchestration as described in [Adding Inputs to an Orchestration](#).

You can also click **Save As** and rename an existing orchestration to create a new one.

Caution: If you use Save As to create a copy of an orchestration, only the orchestration is copied. The Orchestrator Studio does NOT create a copy of the components that are associated with the orchestration's steps. That is, both the original orchestration and the new orchestration use the same components that comprise the orchestration. Therefore, in the new orchestration, do NOT modify the components in any way that would break other orchestrations that use the same components.

A.8.3 Adding Inputs to an Orchestration

Orchestration inputs identify the data an orchestration consumes from a calling custom program, a third-party application, an IoT device, or Cloud service. For example, you could have an orchestration consuming data from an IoT device, with a "SensorID" input to pass a sensor ID value to an orchestration and a "TemperatureReading" input to pass a temperature value to an orchestration.

Each component that you add to an orchestration—a service request, rule, white list, or cross reference—also includes one or more inputs. For example, inputs in a rule evaluate data received from orchestration inputs to determine the next step in an orchestration. When you assemble an orchestration, you map the orchestration inputs to the inputs defined in the components. See [Mapping Orchestration Inputs](#).

In the Orchestration design page, there are three different types of inputs you can add to an orchestration: (**Note:** Starting with Orchestrator Studio 6.1.0, the inputs are organized in three separate expandable grids.)

- Orchestration Inputs.**

You can manually enter orchestration inputs, including arrays (Orchestrator Studio 6.1.0), to identify the data an orchestration consumes from third-party applications or devices.

Starting with Orchestrator Studio 6.1.0, after adding a component as a step in the orchestration, you can use the **Add Inputs to Orchestration** button at the end of the step

to automatically add inputs defined in a component as inputs to the orchestration. You can remove any inputs as needed, and then map the orchestration inputs to the appropriate inputs in the orchestration steps.

- **Value from Steps.**

When you add a form request, data request, or cross reference configured to return values from EnterpriseOne, the variables defined to represent the return values appear as additional orchestration inputs. This gives you the option to map data returned from one component to a subsequent component in an orchestration.

Also, if you add a service request step that generates output, the Orchestrator Studio automatically adds a <servicerequestname>.output as an orchestration input. This is a variable that represents the JSON output of the service request, which gives you the option to map the JSON to a subsequent orchestration step.

- **System Values.**

Starting with Orchestrator Studio 6.0.x, new orchestrations include the following default inputs: User Address Book Number and User Name. These inputs represent the originator of the orchestration when executed at runtime. This gives you the option to map these inputs to an orchestration step. Orchestrator Studio 6.1.0 contains an additional default system value called System Date.

To manually add orchestration inputs in the Orchestration Inputs grid:

1. In the first row in the inputs area, enter the name of the input in the Input column.
2. In the Value Type column, select the input value type. Valid values are:
 - String
 - Numeric
 - Array (Orchestrator Studio 6.1.0)

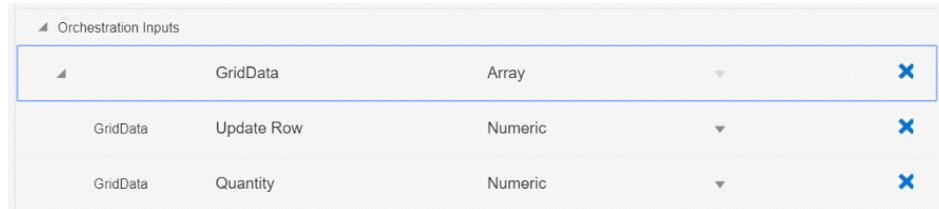
If the input is a date, you can use any of the following date formats:

- dd/MM/yyyy
- dd/MM/yy
- yyyy/MM/dd
- MM/dd/yyyy
- MM/dd/yy
- yy/MM/dd

You can also use the following date formats, which create additional inputs derived from the passed value:

- Milliseconds
 - yyyy-MM-dd 'T' HH:mm:ss.SSSZ
3. Optionally, use the Default Value column to enter a default value that you want to pass through the orchestration input.
 4. If you add an array for an input, the grid expands so you can define the inputs in the array.

As shown in the following example, a user added an array named GridData to the highlighted row. And then entered the inputs "Update Row" and "Quantity" for the array in the subsequent rows.



5. Click **Save** to save your changes.

To generate orchestration inputs from an orchestration component (Orchestrator Studio 6.1.0):

1. In the row with the orchestration step, click the **Add Inputs to Orchestration** button. The inputs defined in the component appear as inputs to the orchestration.
2. Use the **X** button at the end of each row to remove any unnecessary inputs.
3. Modify the **Value Type** and **Default Value** columns as needed.
4. Click **Save**, and then see [Mapping Orchestration Inputs](#) to map these inputs to the appropriate component inputs.

A.8.4 Adding Steps to an Orchestration

Each component (service request, rule, cross reference, white list) that you add to an orchestration is an orchestration step. It is recommended that you create the component that you want to add to an orchestration before you add it as an orchestration step.

[Figure A–8](#) shows an orchestration that contains multiple rule, cross reference, and service request steps:

Figure A–8 Steps in the AddConditionBased Alert Sample Orchestration in the Orchestrator Studio

Type	Action	Iterate Over	Name
Cross Reference			JDE_XREF_Sample_SensorLocation Personal XRE_1611280001CUST
Cross Reference			JDE_XREF_Sample_AlertNotificationRecipients Personal XRE_1611280002CUST
Rule			JDE_RULE_Sample_CBMAalarm_1 Personal RUL_1611280002CUST
Form Request	True		JDE_SREQ_Sample_AdcdBArt_Alarm Personal SRE_1611280004CUST
Rule	False		JDE_RULE_Sample_CBMAalarm_2 Personal RUL_1611280003CUST
Form Request	True		JDE_SREQ_Sample_AdcdBArt_Alarm Personal SRE_1611280004CUST
Rule	False		JDE_RULE_Sample_CBMAalarm_3 Personal RUL_1611280001CUST

Each row in the Orchestration Steps grid represents a step in the orchestration. The grid displays the following information about each step:

- ❑ **Type.** Displays the component type: Rule, Service Request, Cross Reference, or White List.

■ **Action.** Use this column to define which subsequent step is invoked based on the criteria in the preceding rule. See [Defining the Actions Between a Rule and Dependent Components](#) for details.

■ **Iterate Over.** Use this column to pass a data set retrieved from a form request or data request to a subsequent orchestration step. See [Passing a Data Set to a Subsequent Orchestration Step](#) for more information.

■ **Name.** Displays the name of the component.

Use the "Insert Step Before" and "Insert Step After" buttons to add a step before or after another step in the orchestration. For example, if you determine that you need to add a white list to an existing orchestration, you can insert the white list as an initial step before the other steps.

To add steps to an orchestration:

1. To add the initial step to an orchestration, click the **Add Step** button (+ symbol).
2. In the "Enter Type of Step" pop-up field, select one of the following steps to add to the orchestration, and then click the **Ok** button.

■ **Cross Reference**

■ **Rule**

■ **Service Request**

■ **White List**

The Orchestrator Studio displays the first step in the grid.

3. Define the component for the step:

- a. To use an existing component for the new step, click the drop-down menu in the Name column and select a component.
- b. To create a new component for the step, click the **Edit** button (pencil icon) at the end of the row to access the design page for creating the new component.

After creating the component, when you return to the Orchestration design page, you can select the component from the drop-down list in the orchestration steps grid to add it.

See the appropriate topics in this chapter on how to create each type of orchestration component.

4. In the Transformations area on the right side of the page, map the orchestration inputs to inputs in the orchestration steps. See [Mapping Orchestration Inputs](#).
5. To add additional steps to an orchestration:
 - a. Select a step in the grid, and then click either the **Insert Step Before** or **Insert Step After** button to add an additional step before or after the selected step.
 - b. In the "Enter Type of Step" pop-up box, select the step that you want to add.
 - c. Click **Ok**.

To remove a step from an orchestration:

Caution: Before you delete a step, make sure the step is not used by any other component in the orchestration.

1. Select the step that you want to remove.

- Above the grid with the steps, click the **Remove Step** button (the X button).

If you make a mistake when updating an orchestration, click the **Restore All** button in the upper-right corner of the design page to restore the orchestration to its last saved version, which erases any changes made since the last save.

A.8.4.1 Defining the Actions Between a Rule and Dependent Components

The Orchestration design page contains an Action column for defining the actions between a rule step and other orchestration steps in an orchestration. After you create a rule with conditions and add it as a step to an orchestration, you need to define the action the orchestration takes if the condition in the rule is met. For example, if a rule contains a condition that when met should invoke a service request step, then in the row with the service request step, you set the Action column to **True**.

You can also define a **False** action to invoke a different orchestration step when a condition in the initial rule is NOT met. A **False** action can invoke a different Service Request step or another rule step that contains additional conditions for which the incoming data is evaluated against. Thus, you can design an orchestration with as many rules and service requests as your business requirements necessitate.

Figure A–9 shows an example of an orchestration with two rule steps and two service request steps, with the following defined actions:

- For the first service request step, the action is set to **True** to instruct the orchestration to invoke this service request step when the condition in the first rule step is met.
- The action for the second (nested) rule step is set to **False** to instruct the orchestration to invoke this rule when the condition in the first rule is NOT met.
- The action for the second service request step is set to **True** to instruct the orchestration to invoke this service request when the condition in the second rule is met.

Figure A–9 Defining Orchestration Actions

Type	Action	Iterate Over	Name
Cross Reference			JDE_XREF_Sample_SensorLocation Personal XRE_1611280001CUST
Cross Reference			JDE_XREF_Sample_AlertNotificationRecipients Personal XRE_1611280002CUST
Rule			JDE_RULE_Sample_CBMAccess_1 Personal RUL_1611280002CUST
Form Request	True		JDE_SREQ_Sample_AddCBAlert_Alarm Personal SRE_1611280004CUST
Rule	False		JDE_RULE_Sample_CBMAccess_2 Personal RUL_1611280003CUST
Form Request	True		JDE_SREQ_Sample_AddCBAlert_Alarm Personal SRE_1611280004CUST

To set the Action column to True or False:

- In the Orchestration design page, in the appropriate Rule or Service Request step row, click in the Action column.
- Select either **True** or **False** as appropriate.
- Click **Save**.

After completing the orchestration, you can use the Orchestrator Client to test the orchestration in a test environment. You can access the Orchestrator Client from the drop-down menu in the upper-right corner of the Orchestrator Studio. See [Chapter 9, "Testing Orchestrations in the EnterpriseOne Orchestrator Client"](#) for more information.

A.8.5 Mapping Orchestration Inputs

On the Orchestration design page, use the Transformations area to map orchestration inputs to inputs defined in an orchestration step, such as inputs in a rule, cross reference, white list, or service request component. The Transformations area includes an Auto Map button for automatically mapping any matching inputs in the orchestration and orchestration step.

If an orchestration and an orchestration component use different input names to identify the same data, you can use the grid in the Transformations area to map the inputs. This facilitates the reuse of components, so you can avoid having to create a new component simply to match the input names (from a third-party application or device) in a new orchestration.

To better understand how to map inputs, see the example depicted in [Figure A–10](#). In this example, two orchestration inputs have the same name as the inputs in the service request. These inputs were automatically mapped using the Transformations Auto Map button. The orchestration also contains a SerialNumber input to identify equipment, which is represented in EnterpriseOne as "EquipmentNumber." To map these inputs in the Transformations area, "SerialNumber" was selected in the Orchestration Input drop-down list (renamed to Available Values in Orchestrator Studio 6.1.0) next to the EquipmentNumber service request input.

Figure A–10 Transformations Example

Type	Action	Iterate Over	Name	Service Request Input	Available Values
Form Request			IDT_UpdateEquipmentLocation Personal SRE_1705040005CUST	Equipment Number	SerialNumber
Form Request			IDT_UpdateSuppData_Location Personal SRE_1705080002CUST	Latitude	Latitude
Rule			IDT_GeoFence Personal RUL_1705040003CUST	Longitude	Longitude

Input	Value Type	Default Value
Orchestration Inputs		
SerialNumber	Numeric	X
Latitude	Numeric	X
Longitude	Numeric	X
Equipment Manager	String	X
Date	yyyy-MM-dd'T'HH:mm:ss	X

Initially, when you have a small number of orchestrations in your system, it is recommended to keep all input names consistent among the orchestration and all components (orchestration steps) used by orchestration. But as your library of orchestrations and orchestration components increases, instead of creating new components with input names that match the orchestration inputs, you can use transformations to map the orchestration inputs to an existing component.

To map inputs:

1. In the Orchestration design page, select an orchestration step to which you want to map orchestration inputs.
2. Click the Transformations **Auto Map** button to map matching inputs. The Studio automatically maps inputs with the same name; the matching orchestration inputs appear in the Orchestration Input column (named Available Values in Orchestrator Studio 6.1.0) next to the matching input in the "*Orchestration Step Input*" column.
3. To map an orchestration input to an orchestration step input that does not have the same name:

- a. In the Transformations table, click the drop-down menu in the Orchestration Input column next to the orchestration step input.
 - b. Select the orchestration input to map to the orchestration step input.
- For example, in [Figure A–10](#), SerialNumber is mapped to EquipmentNumber in the Service Request Input.
4. You can map a literal value to the orchestration step input by entering a value in the Default Value column.
 5. (Orchestrator 6.1.0) If you are mapping values from an array, you can configure the orchestration to iterate over the orchestration step so the step is called once for each row in the array:
 - a. After defining the array in the Orchestration Inputs section, click the **Iterate Over** drop-down list in the Transformations area and select the name of the array input.
 - b. Make sure to map the orchestration inputs for the array to the inputs defined in the orchestration step, as shown in the following example:

Type	Action	Iterate Over	Name
Service Request	GridData	GridData	Update SO Personal SRE_1711270008CUST

Input	Value Type	Default Value
GridData	Array	
GridData	Row Numb	Numeric
GridData	Quantity	Numeric

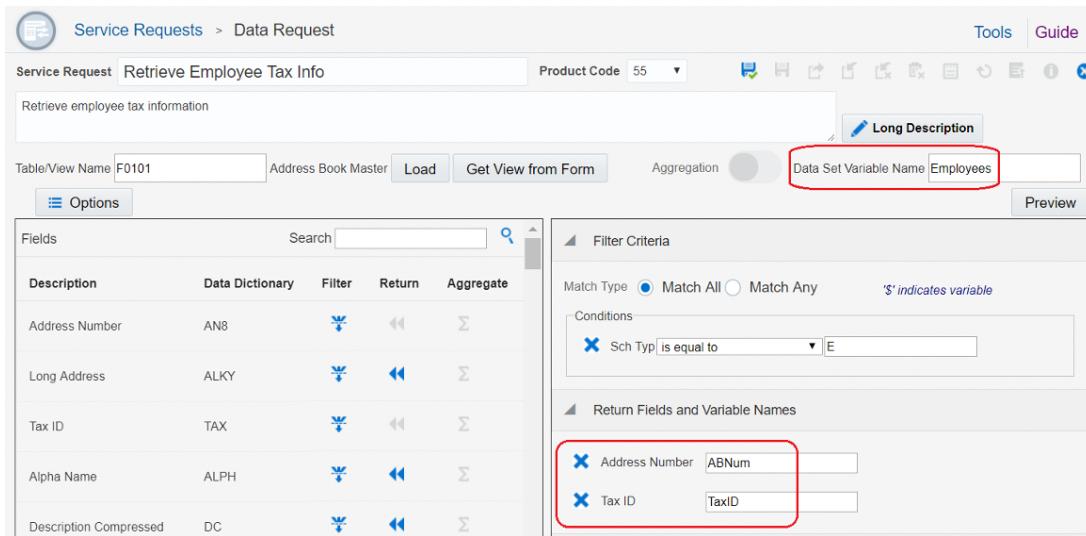
Service Request Input	Available Values
Row Numb	Row Numb
Quantity	Quantity
P4210_Version	

Notice that in the Orchestration Steps grid, the name of the array was automatically inserted into the Iterate Over column in the Service Request step.

A.8.6 Retrieving and Passing Data Sets in an Orchestration (Orchestrator Studio 6.0.x)

You can configure a form request or a data request to return a data set. You can also configure an orchestration to pass a data set returned from a form request or a data request to another step in the orchestration. At runtime, the orchestration executes the form request or data request once per row in the data set.

[Figure A–11](#) shows an example of a data request configured with a data set named "Employees" that will retrieve the address book number and tax ID of all employee records in EnterpriseOne.

Figure A–11 Data Request Configured to Retrieve a Data Set

A.8.6.1 Configuring a Data Request to Retrieve a Data Set

1. On the Data Request page, enter a name for the data set in the **Data Set Variable Name** field.
2. Define filtering criteria for the data set.
3. In the grid, click the "**Return**" icon next to the fields that you want the data request to return in a data set.
4. In the "Return Fields and Variable Names" area, enter a variable name for each return field.

When you add this data request to an orchestration, you can use these variables to map the data in the data set to a subsequent orchestration step.

For more information about configuring a data request, see [Configuring a Data Request](#).

A.8.6.2 Configuring a Form Request to Retrieve a Data Set

Note: Data sets from grids on a power form are not supported.

1. In the form request, locate the row with the *Form Name*-Grid node.
2. In the *Form Name*-Grid row, enter a name for the data set in the Variable Name column. If you configure the orchestration to pass the data in the data set to a subsequent orchestration step, you enter this variable name in the Iterate Over column of the orchestration step to which you want to pass the data.
3. For each column that you want to include in the data set, click **Return**.
4. For each column that you specified to include in the data set, enter a variable name so that the returned column can be mapped to a subsequent orchestration step.

For more information about configuring a form request, see [Configuring a Form Request in the Orchestrator Studio](#).

A.8.6.3 Passing a Data Set to a Subsequent Orchestration Step

You can configure an orchestration to pass each row of a data set to an orchestration step.

1. Add the form request or data request defined with a data set to the orchestration.
2. In the **Iterate Over** column of the orchestration step to which you want to pass the data set, enter the data set name defined in the form request or data request.
3. Map the data set fields to the inputs in the orchestration step. See [Mapping Orchestration Inputs](#).
4. Click **Save**.

A.8.7 Working with Orchestration Output

Use the Orchestration Output page to view the JSON representation of orchestration output. Viewing the output enables you to validate the fields that contain the data you want returned in the orchestration output for output mapping.

Orchestration output can include values from fields and grid columns that you specify as returns when setting up a form request, data request, or cross reference in an orchestration. It can also include the response from a connector, custom service request, or the result of a rule (true or false).

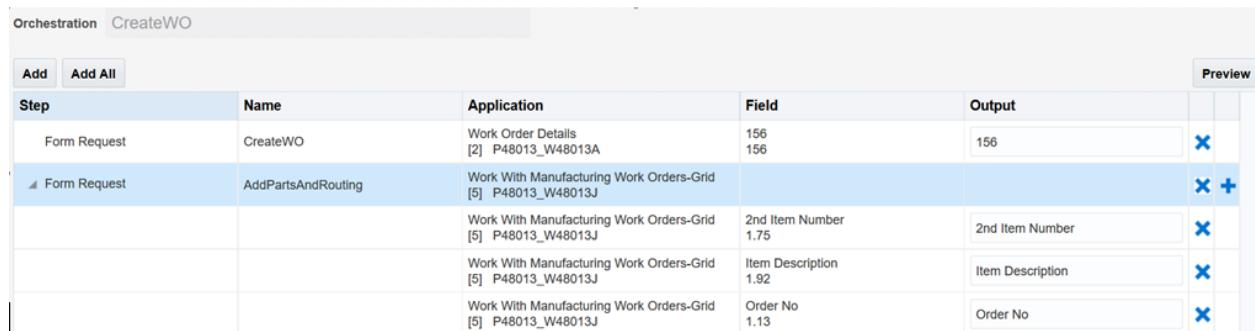
On the Orchestration Output page, you can use Groovy scripting to manipulate the contents of the response to refine the orchestration output as required by the parameters in the consuming device or program.

Note: Orchestration output is available only with version 2 orchestrations created in Orchestrator Studio 5.0.1 or higher. For orchestrations created prior to Orchestrator Studio 5.0.1, you must update them to version 2 orchestrations to view orchestration output.

To work with orchestration output:

1. On the Orchestration design page, click the **Orchestration Output** button.
2. If you want to view the JSON code of all return fields in an orchestration, click **Add All**.

The grid displays all of the components or steps in the orchestration that contain return fields, as shown in the following example:



The screenshot shows the 'Orchestration Output' grid for an orchestration named 'CreateWO'. The grid has columns for Step, Name, Application, Field, and Output. The 'Add' and 'Add All' buttons are at the top left, and a 'Preview' button is at the top right. The 'Output' column contains a 'Remove' icon (X) and a 'Select' icon (+). The 'Field' column lists specific fields like '156', '2nd Item Number', 'Item Description', and 'Order No'. The 'Application' column lists components like 'Work Order Details' and 'Work With Manufacturing Work Orders-Grid'. The 'Name' column lists steps like 'CreateWO' and 'AddPartsAndRouting'.

Step	Name	Application	Field	Output
Form Request	CreateWO	Work Order Details [2] P48013_W48013A	156 156	156 X
Form Request	AddPartsAndRouting	Work With Manufacturing Work Orders-Grid [5] P48013_W48013J		X +
		Work With Manufacturing Work Orders-Grid [5] P48013_W48013J	2nd Item Number 1.75	2nd Item Number X
		Work With Manufacturing Work Orders-Grid [5] P48013_W48013J	Item Description 1.92	Item Description X
		Work With Manufacturing Work Orders-Grid [5] P48013_W48013J	Order No 1.13	Order No X

3. Instead of Add All, you can individually select the return fields for which you want to preview the JSON code:
 - a. Click **Add** to view a list of all steps (orchestration components) in the orchestration that contain return fields.

You can expand each step to see the return fields. The Output Field column displays the variable name defined for a return field.

- b.** In the Select column, click the check box next to any return field that you want to add to your orchestration output, and then click **OK**.

The field will be returned in JSON format, which you can also preview on the design page.

- 4.** To modify the return fields for which you want to see the JSON representation:

- ❑ You can change the variable name for the return field in the Output column.
- ❑ Click **Delete (X)** at the end of row to remove a return field from the JSON preview.
- ❑ Click **Add (+)** to add any additional grid controls not currently selected for output.

- 5.** Click the **Preview** button.

The following image shows an example of the JSON code for three grid return fields: 2nd Item Number, Item Description, and Order Number:

Add	Add All			
Step	Name	Application	Field	Output
Form Request	AddPartsAndRouting	Work With Manufacturing Work Orders [5] P48013_W48013J		
		Work With Manufacturing Work Orders [5] P48013_W48013J	2nd Item Number 1.75	2nd Item Number
		Work With Manufacturing Work Orders [5] P48013_W48013J	Item Description 1.92	Item Description
		Work With Manufacturing Work Orders [5] P48013_W48013J	OrderNo 1.13	Order No

```
{
  "Grds": [
    {
      "Grid ID": "1",
      "Title": "Work With Manufacturing Work Orders",
      "Row Set": [
        {
          "2nd Item Number": "XXXXX",
          "Item Description": "XXXXX",
          "Order No": "XXXXX"
        }
      ]
    }
  ]
}
```

- 6.** In the Preview area, you can click the **Copy to Clipboard** button so you can paste the JSON code into your program.
- 7.** (Advanced) In Orchestrator Studio, edit the provided Groovy script template in the Manipulate Output area to manipulate the contents of the response to refine the orchestration output.

See [Groovy Template for Manipulating Output from an Orchestration Response](#) for more information.

- 8.** Click the **Test** button to view the output.

The Orchestrator Studio displays a preview of the output manipulated by the Groovy script.

- 9.** To save the orchestration outputs, return to the Orchestration design page and click **Save**.

A.9 Creating Schedules for Orchestrations (Orchestrator Studio 6.0.x)

This section contains the following topics:

- ❑ [Section A.9.1, "Understanding Schedules"](#)
- ❑ [Section A.9.2, "Creating a Schedule"](#)
- ❑ [Section A.9.3, "Adding a Schedule to an Orchestration"](#)

A.9.1 Understanding Schedules

The Orchestrator Studio gives you the option to create and assign a schedule to an orchestration. A schedule determines how often the Orchestrator executes an orchestration. For

example, with a schedule, you can create an orchestration that regularly checks for a Watchlist threshold, that when exceeded, sends a notification to the appropriate users.

You define a schedule using minutes, hours, days, or a Cron string. Cron is a scheduling utility using cryptic strings to define an interval. Then you associate a schedule to an orchestration to determine how often it runs. You can attach the same schedule to multiple orchestrations.

You must have been granted proper UDO security to create schedules. Schedules are managed as UDOs, which means you need the proper UDO security in order to publish and share your schedules for others to use, or use schedules that others have published.

The scheduler is a process on the Application Interface Services (AIS) server that starts, stops, and manages schedules. An administrator uses the AIS Server REST API to start, stop, and manage schedules through the scheduler.

A.9.2 Creating a Schedule

Create a schedule to define how often the Orchestrator should execute an orchestration. You can define a schedule using minutes, hours, days, or a Cron string. Cron is a time-based job scheduler that can be used to schedule jobs to run periodically at fixed times, dates, or intervals (for example, every Tuesday and Friday at 9:00 am).

To create a schedule:

1. On the Orchestrator Studio Home page, click the **Tools** link in the upper-right corner.
2. On the Tools page, click the **Schedules** icon.
The Orchestrator Studio displays the Schedules design page.
3. Click the **New Schedule** button.
4. In the Schedules field, enter a name for the schedule. Do **NOT** include special characters in the name.
5. Click the **Product Code** drop-down list to select a product code to associate with the schedule.

This gives an administrator the option to manage UDO security for orchestration components by product code.

6. In the space provided, enter a short description with a maximum of 200 characters. This description should clearly describe the frequency of the schedule so that it can be attached to notification as needed.
7. Click the **Edit Long Description** button to add a long description to provide more detail about the purpose of the component.
8. Do one of the following:
 - ▀ In the **Schedule to Run** section, select a number of minutes, hours, or days to define how often you want the schedule to run.
If you select minutes, you cannot run more often than every five minutes.
 - ▀ In the **Or Enter a Cron String** section, enter a Cron string to define the schedule.
Cron is a time-based job scheduler that can be used to schedule jobs to run periodically at fixed times, dates, or intervals. There are many third-party Cron expression generators available that can help you create a Cron string.
9. Click the **Save** or **Save As** icon in the upper-right corner.

The first time a new schedule is saved, it is saved as a "Personal" UDO. Thereafter, you can use the UDO buttons described in the User Defined Object (UDO) Features section to move the schedule to the appropriate status.

A.9.3 Adding a Schedule to an Orchestration

Adding a schedule to an orchestration does not invoke the orchestration as scheduled. Starting the scheduler is a separate step. After you add a schedule, ask an administrator to start and administer the schedule using REST API services.

For more information about REST API services for starting, stopping, and managing the Scheduler, see *JD Edwards EnterpriseOne Tools REST API for the Application Interface Services Server*.

Note: In order to use the Scheduler with an orchestration, it must be a version 2 orchestration. If you want to update a version 1 orchestration created in a previous version of the Orchestrator Studio, see [Updating Version 1 Orchestration to Version 2 Orchestration](#).

To add a schedule to an orchestration:

1. Select the orchestration to which you want to add a schedule and open it in the Orchestration design page.
2. On the Orchestration design page, click the **Schedule** drop-down list and select a schedule.
3. Click **Save** to save your changes.

A.10 Updating Version 1 Orchestration to Version 2 Orchestration

When executed by the Orchestrator on the AIS Server, orchestrations programmatically call AIS services on the AIS Server to perform specific actions in EnterpriseOne.

Starting with EnterpriseOne Tools 9.2.1.2 is the availability of version 2 AIS services. Version 2 AIS services include all services originally available on the AIS Server (referred to as version 1 services), plus additional services that enable you to create an orchestration that includes the following types of service requests:

- Data request
- Message
- Connector

All orchestrations that you create with Orchestrator Studio 5.0.1 or higher use version 2 AIS services and are referred to as version 2 orchestrations.

All orchestrations created prior to Orchestrator Studio 5.0.1 use version 1 AIS services and are referred to as version 1 orchestrations.

In Orchestrator Studio 5.0.1 or higher, you can update version 1 orchestrations. However, if you add any of the components in the preceding list that rely on version 2 AIS services, you must save the orchestration as version 2 by selecting **Version 2** from the Orchestrator Version drop-down list.

Caution:

If you save a version 1 orchestration as version 2, the following parameters are used for the service request by default:

- ❑ Output Type = grid data
- ❑ Turbo Mode = low

This changes the format of the response, which will affect any device consuming the output returned from the orchestration.

A.11 Reloading Orchestrations and Orchestration Components

Reload Files enables you to reload orchestrations and orchestration components to the state in which they were last saved in the Orchestrator Studio. Use this feature if you do not want to save any modifications that you made.

To reload all files, click the drop-down menu in the upper-right corner of the Orchestrator Studio and then click the **Reload Files** link.

Each individual orchestration component panel also contains a Restore button that you can use to restore an individual component.

A.12 Supported Input Message Formats

Orchestrator supports three input message formats for orchestrations: a standard JD Edwards EnterpriseOne format, a generic format, and Oracle Cloud IoT format. [Example A-1](#) and [Example A-2](#) show an example of the code for each format.

Example A-1 Standard JD Edwards EnterpriseOne Input Message Format

```
{  
    "inputs": [  
        {  
            "name": "equipmentNumber",  
            "value": "41419"  
        },  
        {  
            "name": "description",  
            "value": "test"  
        },  
        {  
            "name": "date",  
            "value": "1427774400000"  
        },  
        {  
            "name": "time",  
            "value": "12:15:15"  
        },  
        {  
            "name": "temperature",  
            "value": "99"  
        }  
    ]  
}
```

Example A-2 Generic or Oracle Cloud IoT Input Message Format

```
{
    "equipmentNumber": "41419",
    "description": "test",
    "date": "1427774400000",
    "time": "12:15:15",
    "temperature": "99"
}
```

A.12.1 Additional Supported Input Message Formats for the Generic Input Format

Additional formats are supported when using the generic input format, as long as the orchestration input values are defined using the full path to the elements used. You may have a deeper JSON structure like this.

```
{
    "equipmentNumber": "41419",
    "equipementDetail": {
        "type": "thermometer",
        "readingDetail": {
            "temperature": 200,
            "units": "F"
        }
    }
}
```

To reference the temperature within the orchestration, you can use the full path delimited by periods, for example:

```
equipmentDetail.readingDetail.temperature
```

A.12.2 Differences Between Input Message Formats

The following list describes the differences between the input message formats:

- The iterateOver attribute for the orchestrationStep element is supported only by the standard JD Edwards EnterpriseOne input message format.
- When using the "detail" form action type with the standard format, it will automatically iterate over all detailInputs and repeatingInputs in order to add multiple rows to a grid. If the generic format is used, only a single grid row can be added.

As shown in [Example A-3](#), "detailInputs" would correspond to grid data; "repeatingInputs" would correspond to individual rows that contain "inputs" that correspond to columns.

If you have a power form with two grids, you could populate both grids using two "detailInputs" structures with different "name" values that correspond to the different grids. "repeatingInputs" would contain a list of rows and for each row you would define a list of "inputs".

You could also define a CrossReforchestration step with iterateOver="GridData" that converts the item value into an EnterpriseOne specific item number. For example, if A123=220 and A124=230, the single orchestration step would convert both.

Example A-3

```
{
    "inputs": [
        {

```

```
        "name": "BranchPlant",
        "value": "30"
    },
{
    "name": "customer",
    "value": "4242"
}
],
"detailInputs": [
{
    "name": "GridData",
    "repeatingInputs": [
        {
            "inputs": [
                {
                    "name": "item",
                    "value": "A123"
                },
                {
                    "name": "Quantity",
                    "value": "3"
                }
            ]
        },
        {
            "inputs": [
                {
                    "name": "item",
                    "value": "A124"
                }
            ]
        }
    ]
}
]
```

A.13 Orchestration Security Considerations

Before the EnterpriseOne Orchestrator can process an orchestration, authentication of the JD Edwards EnterpriseOne user ID and password must take place. It is the responsibility of the originator of the service request to tell the orchestration about the user. The user's credentials must be supplied in a basic authorization header or in the JSON body of the request. The user must also have authorized access to the EnterpriseOne application in which the resulting transaction takes place. The following code is an example of credentials in the JSON body of the request:

```
{
    "username": "JDE",
    "password": "JDE",
    "environment": "JDV900",
    "role": "*ALL"
}
```

The AIS service used with orchestrations is stateless; each call passes credentials to establish a separate EnterpriseOne session. After the transaction is complete, the session closes.

In addition to passing credentials for authentication, you can employ a second level of security for the Orchestrator through whitelisting. Whitelisting enables an initial rudimentary pass/fail

check of the incoming device signature against a predefined list of signatures. If a value passed to the Orchestrator is not a valid value included in the orchestration's white list, the Orchestrator rejects the input. For more information, see [Creating White Lists](#) in this guide.

A.13.1 Restricting Access to Exposed Orchestrations

If you expose orchestrations for business partners or customers to invoke, it is recommended to use an http proxy to allow access to only the endpoints required to execute the orchestration. Configure the proxy to restrict all endpoints except /orchestrator, /discover, /tokenrequest, and /tokenrequest/logout. This allows external users set up with the proper UDO security to discover and call orchestrations.

A.13.2 How to Maintain a Single Session for Multiple Calls to an Orchestration

You can maintain a single AIS Server session for multiple calls to an orchestration by first performing a token request to get a session token. Otherwise, each call to an orchestration on the AIS Server establishes a separate session, which can decrease AIS Server performance.

When performing a token request, a session is established and the AIS Server returns an AIS token in the response. The token is then used for all orchestration calls in the same session.

The amount of time the session remains open is established through the session lifetime settings in the AIS Server. The third party client is responsible for ensuring a valid token is available or re-requesting a new token once a previous token has timed out. If a valid token is not available, the client will receive an error in the response stating the token is invalid.

A.14 Exporting Orchestration Components from the Orchestrator Studio

Caution: Do not use the Export and Import tools to share orchestration component files with other users. With orchestration components stored as UDOs in EnterpriseOne, the management of orchestration components, including the sharing and modifying of shared orchestration components and promoting of orchestration components to the appropriate AIS Server directory for test or production purposes, is administered through the life cycle management tools in EnterpriseOne.

You can export any of the orchestration component types from the Orchestrator Studio to view the source XML files of the components. This is only recommended for advanced users for troubleshooting purposes or for making manual customizations. The Orchestrator Studio exports the source XML files in a zip file.

To export an *orchestration* component, the Orchestrator Studio gives you the option to export only the selected orchestration component or to export the orchestration and all components associated with it. For example, if an orchestration has a service request component and a rule component associated with it, if you export all components, the zip file will contain XML files for the orchestration, service request, and rule.

To export orchestration components:

1. On a component design page, select the component to export and then click the **Export File** button in the upper-right corner.

If you are exporting an orchestration, a dialog box appears in which you can click **All** or **Orchestration Only**.

2. Follow the instructions in the browser to save the zip file to a location on your local machine.

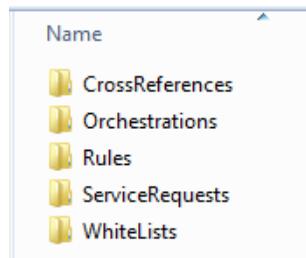
A.15 Importing Orchestration Files in the Orchestrator Studio

Caution: Do not use the Export and Import tools to share orchestration component files with other users. With orchestration components stored as UDOs in EnterpriseOne, the sharing and modifying of shared orchestration components, as well as the promoting of orchestration components to the appropriate AIS Server directory for test or production purposes, is administered through the life cycle management tools in EnterpriseOne.

You might need to import orchestration component XML files that were exported from the Orchestrator Studio for advanced troubleshooting or customization purposes. The Orchestrator Studio Import tool enables you to import individual orchestration XML files or a zip file containing XML files.

Caution: You cannot import orchestration component files that were exported from the EnterpriseOne Object Management Workbench - Web application. The zip file created from an OMW - Web export stores the XML files in a structure that cannot be read by the Orchestrator Studio Import tool.

If importing a zip file that contains orchestration XML files and dependent orchestration component XML files, the zip should contain the following folders with each of the XML files stored in the appropriate folder:



If you import an XML file that has the same name as a current orchestration component (UDO) in the Orchestrator Studio, you have the following options:

- ❑ If the UDO is at a status of "Personal" or "Reserved," you can overwrite the UDO with the XML file.
- ❑ If the UDO is in a "Shared" status, you cannot overwrite it. But you are given the option to import it as a new UDO with a status of "Personal."

To import files:

1. On the Orchestrator Studio Home page, click the **Tools** link in the upper-right corner.
2. On the Tools page, click the **Import Files** icon.
3. On Import Files, click the **Choose File** button.

4. Locate the orchestration component XML files or zip file that contains the orchestration component files that you want to import.

After selecting the files to import, the Import tool checks the XML files that you selected against the current UDOs in the Orchestrator Studio and displays the options that are available for importing the files, as shown in the example in [Figure A–12](#).

Figure A–12 Orchestrator Studio Import Tool

The screenshot shows the 'Select File for Import' screen. At the top, it displays 'Orchestrations-1461866371912.zip' and an 'Update...' button. Below this, it shows 'Number of Files 9' and 'Number of Files Existing on Server 2'. On the right, there is a 'Submit' button. The main area is divided into sections: 'Cross Reference', 'Service Requests', 'Rules', and 'Orchestrations'. Each section lists XML files with import options:

- Cross Reference:** JDE_XREF_Sample_SensorLocation.xml (No update allowed at current status, checkbox unchecked)
- Service Requests:** JDE_SREQ_Sample_AddCBAAlert_Warning.xml (Overwrite existing file, checkbox checked) and JDE_SREQ_Sample_AddCBAAlert_Alarm.xml (File is ready to submit, checkbox checked)
- Rules:** JDE_RULE_Sample_CBMAAlarm_3.xml (File is ready to submit, checkbox checked), JDE_RULE_Sample_CBMAAlarm_1.xml (File is ready to submit, checkbox checked), JDE_RULE_Sample_CBMAAlarm_2.xml (File is ready to submit, checkbox checked), and JDE_RULE_Sample_CBMWarning.xml (File is ready to submit, checkbox checked)
- Orchestrations:** JDE_ORCH_Sample_AddConditionBasedAlert_Generic.xml (File is ready to submit, checkbox checked)

At the bottom right is another 'Submit' button.

5. Review the import option for each file to determine how to proceed with the import. The options are:

- ❑ No update allowed at current status.

The XML file name is the same as an existing component, which has a status of "Pending Approval," so it cannot be overwritten.

- ❑ Select to add or else reserve record and re-import to update.

A component with the same name exists in the Orchestrator Studio in a "Shared" status. Click the check box if you want to import the file as a new UDO. A new name will be generated for the new component upon import.

If you want to overwrite the current component in the Orchestrator Studio, clear the check box. And then change the status of the current component to

"Reserved" and reimport the XML file to overwrite the component.

- ❑ File already exists.

The XML file matches a component that is in a "Personal" or "Reserved" status. Click the check box to import it and overwrite the current component.

- ❑ File is ready to submit.

The XML file is unique and does not already exist in the Orchestrator Studio. Click the check box to import it.

6. You can also click the **Select All** check box to import all XML files that are available for importing.
7. Click the **Submit** button.

The Orchestrator Studio imports the selected components into the components list on the respective design page.

B

Creating Orchestrations with Orchestrator Studio 5.1.0

Note:

This appendix describes how to use Orchestrator Studio 5.1.0 that was available starting with EnterpriseOne Tools 9.2.1.4.

To use Orchestrator Studio 7.0.x.0, the latest version available with EnterpriseOne Tools 9.2.3, see:

- [Chapter 2, "Implementing the Orchestrator Studio"](#) for installation instructions
- [Chapter 4, "Creating Orchestrations with Orchestrator Studio 7.x.x.x"](#)

This appendix contains the following topics:

- [Section B.1, "Understanding the Orchestrator Studio and Orchestrations"](#)
- [Section B.2, "Accessing the Orchestrator Studio"](#)
- [Section B.3, "Navigating the Orchestrator Studio"](#)
- [Section B.4, "Working with Orchestrator Studio Design Pages"](#)
- [Section B.5, "Creating Service Requests"](#)
- [Section B.6, "Creating Rules"](#)
- [Section B.7, "Creating Cross References"](#)
- [Section B.8, "Creating White Lists"](#)
- [Section B.9, "Creating Orchestrations"](#)
- [Section B.10, "Reloading Orchestrations and Orchestration Components"](#)
- [Section B.11, "Supported Input Message Formats"](#)
- [Section B.12, "Setting Up Orchestration Security"](#)
- [Section B.13, "Exporting Orchestration Components from the Orchestrator Studio"](#)
- [Section B.14, "Importing Orchestration Files in the Orchestrator Studio"](#)

B.1 Understanding the Orchestrator Studio and Orchestration

The Orchestrator Studio is a web-based application for creating orchestrations and the components that comprise an orchestration. The Orchestrator uses these components to process a single orchestration instance on the AIS Server.

Use the Orchestrator Studio to create the following components:

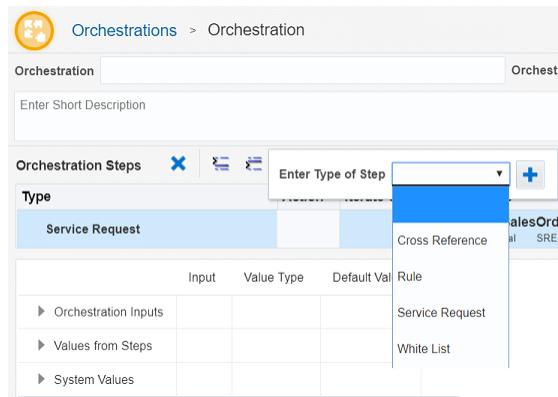
- ❑ **Orchestrations.** An orchestration is the master component that provides a unique name for an orchestration process. The orchestration is where you define the inputs for the orchestration, the expected incoming data. The orchestration also includes orchestration steps, which are invocations to the other components described in this list. When the Orchestrator invokes an orchestration, it processes the steps defined in the orchestration to enable the transfer of data from a third-party to EnterpriseOne.
- ❑ **Service Requests.** A service request can contain instructions to perform a business transaction or query data in EnterpriseOne, send a message to EnterpriseOne users or external users, execute a custom process such as routing data into another database, or invoke another orchestration or REST service through a "connector" service request.
- ❑ **Rules.** A set of conditions that the input to the orchestration is evaluated against to produce a true or false state. With rules, a false outcome or true outcome can invoke further orchestration steps. You can also nest rules, in which an outcome of one rule can invoke a different rule, to produce complex evaluations. You can also use custom Java to define rules.
- ❑ **Cross References.** A set of data relationships that map third-party values to EnterpriseOne values. For example, a device's serial number can be cross-referenced to a JD Edwards EnterpriseOne Equipment Number for use in service requests.
- ❑ **White Lists.** A white list contains an inclusive list of values permitted in the orchestration and terminates the orchestration process if the data is not recognized.

A simple orchestration requires at a minimum an orchestration and a service request to run. When you save an orchestration in the Orchestrator Studio, the name of orchestration is used to define an endpoint on the AIS Server. The endpoint URL is:

`http://<server>:<port>/jderest/orchestrator/<orchestrationname>`

To invoke an orchestration, third-party applications or devices use a post operation to this url, where <orchestrationname> is the name of your orchestration.

Figure B-1 shows the drop-down list of steps that you can add to an orchestration. Each step in an orchestration is simply a reference to a cross reference, rule, service request, or white list component.

Figure B-1 *Orchestration Steps in the Orchestrator Studio*

B.1.1 Reusable Orchestration Components

Orchestration components are reusable. You can include the same component, such as a rule or cross reference, in more than one orchestration. If a component is used as a step in more than one orchestration, you should evaluate how it is used in the other orchestrations before modifying it.

To determine if a component is used by other orchestrations, select the component and then click the "i" button to display a pop-up window that lists the orchestrations where the component is used.

When in doubt, use the "Save As" button to create a new component from an existing one. This enables you to give it a new name and modify it as necessary, and eliminates the risk of breaking other orchestrations where the component is used.

B.1.2 Orchestrations as User Defined Objects

All orchestration components created in the Orchestrator Studio are stored as user defined objects (UDOs) in EnterpriseOne. The Orchestrator Studio includes UDO features for creating, sharing, and modifying orchestration components as UDOs. See [User Defined Object \(UDO\) Features](#) for a description of the UDO features.

B.2 Accessing the Orchestrator Studio

The Orchestrator Studio is a web application that runs in a web browser. Ask your system administrator for the URL to the Orchestrator Studio.

Important: Before users can access the Orchestrator Studio, an administrator must set up security to authorize access to the Orchestrator Studio design pages and determine the actions Orchestrator Studio users can perform. See [Chapter 10, "Administering the Orchestrator Studio and Orchestrations"](#) for more information.

To access the Orchestrator Studio:

1. In a web browser, enter the URL to the Orchestrator Studio:
`http://<adf_server>:<port>/OrchestratorStudio/faces/index.jsf`
2. On the Orchestrator Studio Sign In screen, enter your EnterpriseOne User credentials, environment, and role.

Note: It is highly recommended that you enter an EnterpriseOne environment used for testing, not a production environment.

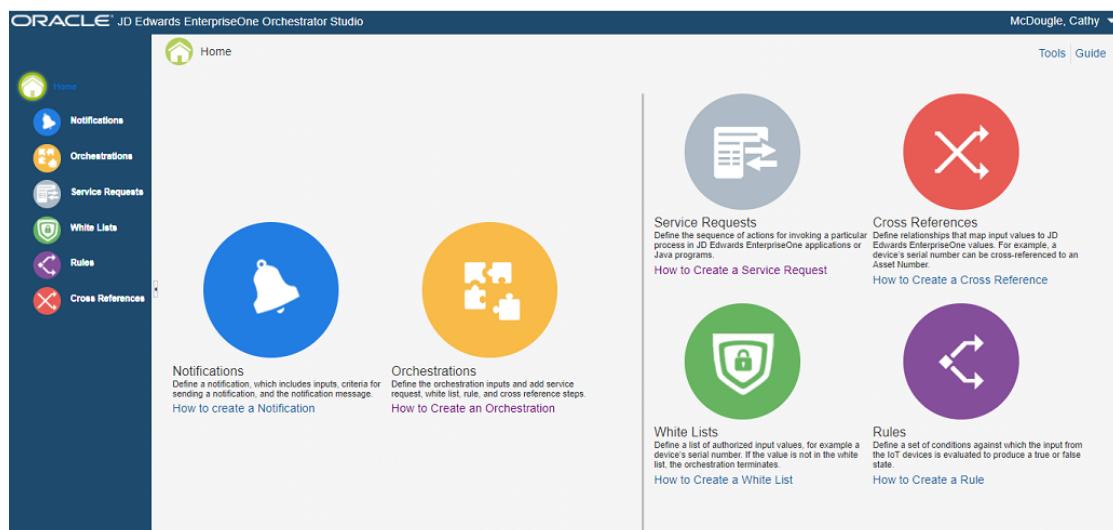
3. Click the **Login** button.

In the Orchestrator Studio, click the drop-down menu in the upper-right corner to view the path to the AIS Server. The drop-down menu also provides a link to log out of the Orchestrator Studio.

B.3 Navigating the Orchestrator Studio

The orchestration component icons on the Orchestrator Studio Home page take you to the design pages for creating and modifying each orchestration component. You can click the **Home** icon at the top left of the Home page to display a side panel, which provides another way to access the orchestration component design pages. You can also access this side panel within the component design pages for easy navigation between the different design pages. Figure B–2 shows the Home page with the side panel enabled.

Figure B–2 Orchestrator Studio Home



The Tools link in the upper-right corner of the Home page provides access to the Orchestrator Studio Tools page. This page provides links to the Orchestrator Client for testing orchestrations, the Import tool for importing orchestration files, and the JD Edwards EnterpriseOne web client. For more information, see the following topics:

- ❑ [Testing Orchestrations in the EnterpriseOne Orchestrator Client](#)
- ❑ [Importing Orchestration Files in the Orchestrator Studio](#)

B.4 Working with Orchestrator Studio Design Pages

This section describes:

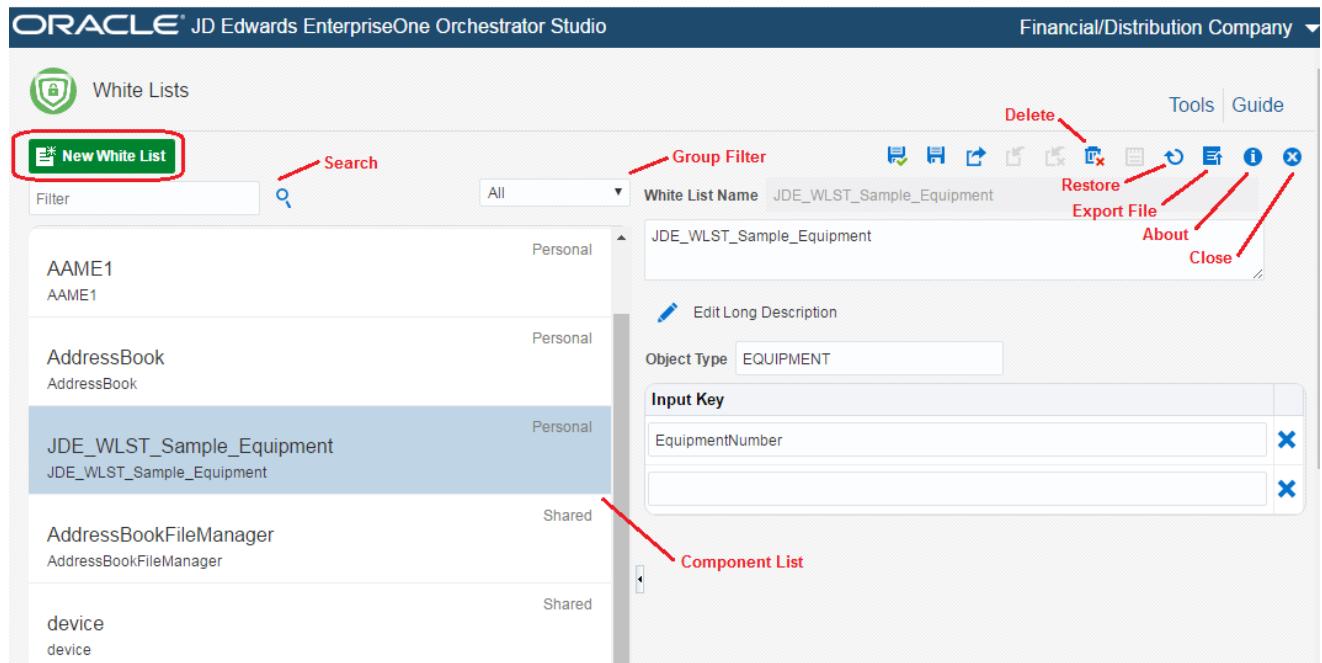
- ❑ [Design Page Features](#)
- ❑ [User Defined Object \(UDO\) Features](#)
- ❑ [Working with the Graphical Representation of an Orchestration](#)

B.4.1 Design Page Features

All Orchestrator Studio design pages contain the following features, which are highlighted in Figure B-3:

- **Component list.** Displays a list of existing components.
- Use the vertical divider next to the component list to adjust the size of the list. You can click the raised tab on the divider to hide or show the component list.
- **Group Filter drop-down list.** Enables you to display components in the component list by UDO status: Personal, Pending Approval, Rework, Reserved, Shared, or All.
- **Search field.** Search for an existing component in the list.
- **New <Component> button.** Create a new component.
- **i (About).** Takes you to the About page which provides the Status, Detail, Description, and Object Name of the selected component. It also shows a list of the orchestrations where the component is used.
- **Restore All or Restore <Component>.** Restore the component to its original state if you made a mistake and do not want to save your changes.
- **Export File.** Export the component file to your local machine, which you should use only to inspect the XML of the component. See [Exporting Orchestration Components from the Orchestrator Studio](#) in this guide for more information.
- **Close.** Exit the design page.

Figure B-3 Orchestrator Studio Design Page Features



B.4.2 User Defined Object (UDO) Features

Orchestration components are saved and managed as UDOs in EnterpriseOne. As such, the Orchestrator Studio contains UDO features, highlighted in Figure B-4, to create orchestration components for your own personal use, publish or "share" orchestration components, and modify shared orchestration components created by other users.

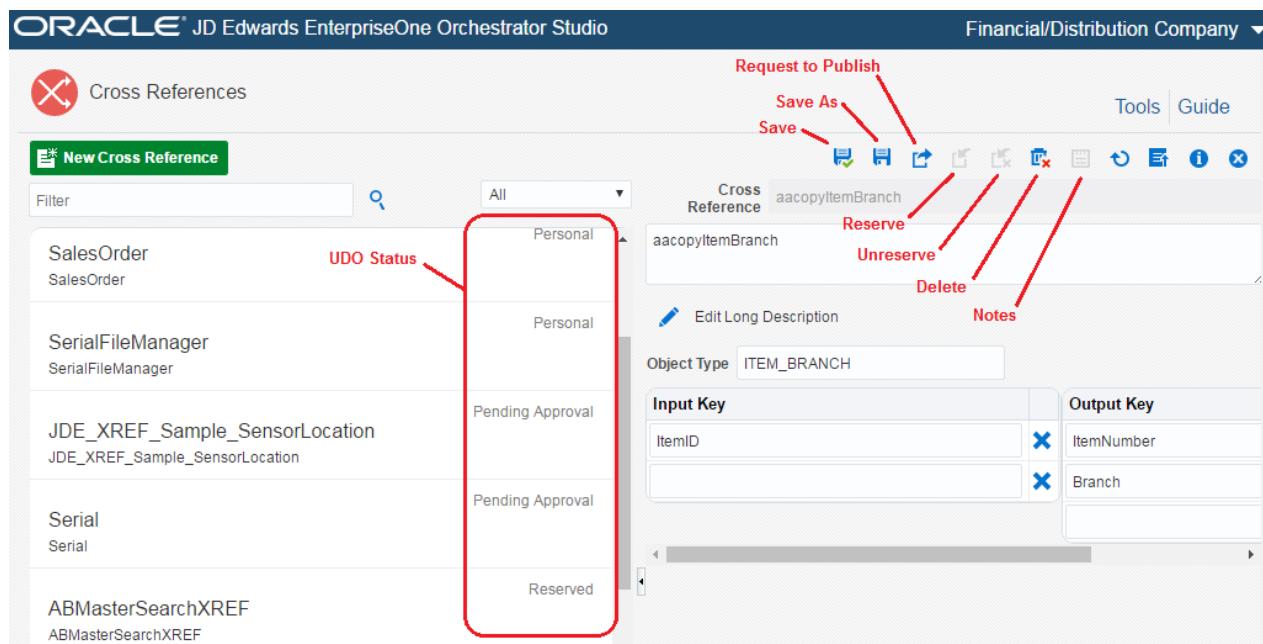
Note: The actions that you are allowed to perform in the Orchestrator Studio depend on the UDO security permissions granted to you by a system administrator. See [Setting Up UDO Security for Orchestrator Studio Users \(Release 9.2.1\)](#) in this guide for more information.

Orchestration components as UDOs enables administrators to use EnterpriseOne administration tools to manage the life cycle of orchestration components. For more information about the life cycle management of UDOs, see "UDO Life Cycle and Statuses" in the *JD Edwards EnterpriseOne Tools Using and Approving User Defined Objects Guide*.

Table B-1 describes the UDO buttons in the Orchestrator Studio design pages and the life cycle status enacted by each UDO action.

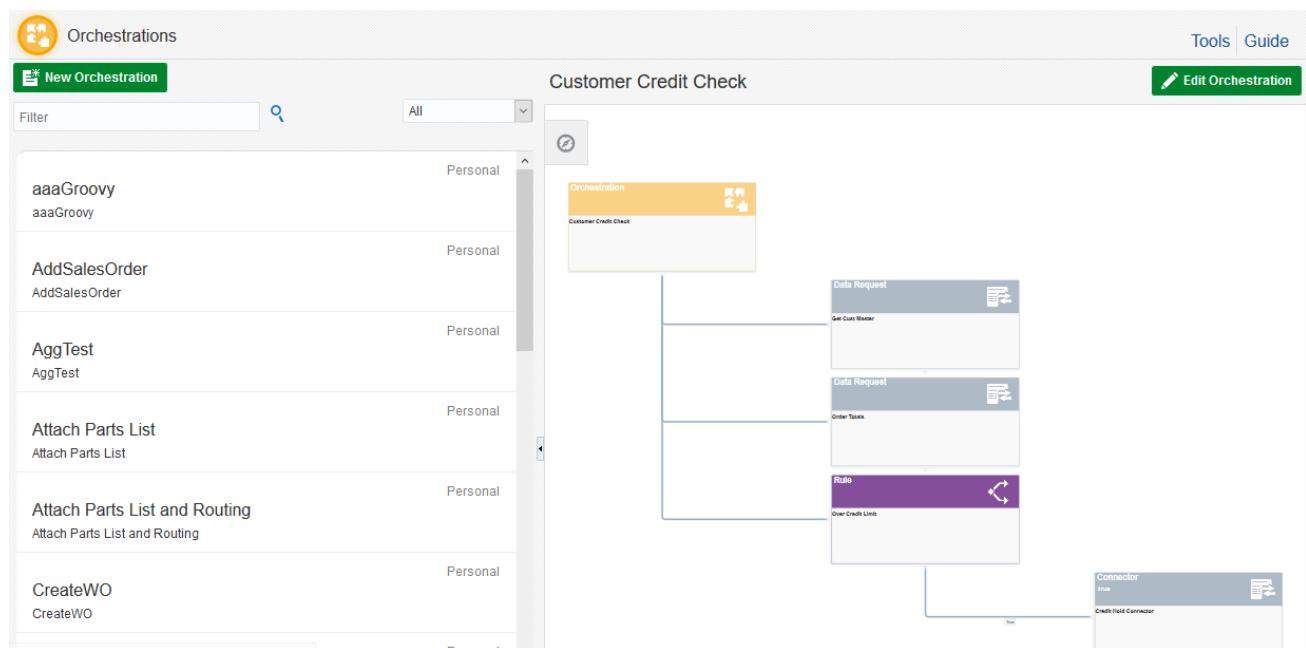
Table B-1 Description of UDO Features in Orchestrator Studio Design Pages

UDO Button	Description
Save and Save As	Saves the orchestration component to a status of "Personal." This means that the component has not been shared. Components with a status of "Personal" are components that you are developing and have not been shared for publishing to the AIS Server.
Request to Publish	Sends the orchestration component for approval for sharing. An administrator or approver must approve it in order for the UDO to be shared. The component status changes to "Pending Approval" in the component list and then changes to "Shared" once it is approved. If rejected, that status changes to "Rework." At that point, you can edit the component and then use the Request to Publish button to send it for approval again.
Reserve	Reserves a shared UDO so you can modify it. When reserved, no other users can make changes to it. The component status changes to "Reserved."
Unreserve	Cancels the reserved component, which returns the status of the component to "Shared."
Delete	Deletes a "Personal" UDO. You cannot use this button to delete a shared UDO. Shared UDOs can only be deleted by an administrator.
Notes	Available when the component is in the "Pending Approval" status, this button enables you to add an additional note to send to the approver of the UDO. The Notes button is active only if there was a note added the first time the UDO was sent for approval using the "Request to Publish" button. This feature enables you to add an addendum to the original note.

Figure B–4 Design Page UDO Features

B.4.3 Working with the Graphical Representation of an Orchestration

The initial Orchestrations design page shows a graphical representation of an orchestration with all its components, as shown in Figure B–5.

Figure B–5 Initial Orchestrations Design Page

The graphic area includes the following features:

- ❑ Navigation toolbox

Click the Control Panel icon in the upper-left corner of the graphic area to display a toolbox for navigating. Use the directional controls to pan left, right, up, and down, as well

as zoom in or zoom out. Click "Zoom to Fit" to display the entire graphical representation in the window. Use the layout buttons to change the layout to vertical, horizontal, tree, radial, or circle. This helps to view more complex orchestrations that contain multiple components.

- **Informational hover help**

Hover your mouse over a component in the graphical area to view an enlarged image of the component. Hovering over the labels on the lines between a rule component and its child components magnify the "True" or "False" label. A "True" label indicates the child component will be invoked if the conditions in the rule are met. A "False" label indicates the child component will be invoked when the condition of the rule is not met.

- **Isolate and Restore buttons**

Click the Isolate button on the left side of a component to show only that component in the graphic area. Click Restore to display all orchestration components.

- **Access to the design page for editing the component**

When you click a box representing a component, the Orchestrator Studio takes you to the design page for modifying that particular component.

B.5 Creating Service Requests

This section contains the following topics:

- [Section B.5.1, "Understanding Service Requests"](#)
- [Section B.5.2, "Creating a Service Request"](#)
- [Section B.5.3, "Configuring a Form Request"](#)
- [Section B.5.4, "Configuring a Custom Service Request with Java \(Advanced\)"](#)
- [Section B.5.5, "Configuring a Custom Service Request with Groovy \(Advanced\) \(Studio 5.1.0\)"](#)
- [Section B.5.6, "Configuring a Data Request"](#)
- [Section B.5.7, "Configuring a Message Request"](#)
- [Section B.5.8, "Configuring a Connector"](#)

B.5.1 Understanding Service Requests

A service request component provides the instructions the Orchestrator uses to invoke and perform a particular task in EnterpriseOne. You can configure a service request to perform a business transaction, query data, send a message to EnterpriseOne users or external users, execute a custom process such as routing data into another database, or invoke another orchestration or REST service.

The following list describes the service request types you can create in the Orchestrator Studio:

- **Form Request**

A form request contains the instructions for the orchestration to perform a particular task in EnterpriseOne.

- **Custom**

Use custom Java or Groovy to execute a custom process or to route data into another database.

- **Data Request**

Use a data request in an orchestration to query and return values from an EnterpriseOne table or business view. You can also configure a data request to perform an aggregation on data to return aggregate amounts.

- ❑ Message

Use a message request if you want an orchestration to send a message to an email address or EnterpriseOne users through the EnterpriseOne Work Center.

- ❑ Connector

Use a connector request to invoke an orchestration or a REST service. For a connector to an orchestration, a connector can invoke a local orchestration or an orchestration on another AIS Server, such as an AIS Server on another EnterpriseOne system in a multisite operation.

Before you create a service request, you must first identify the EnterpriseOne task or business process that you want the service request to perform. See [Identifying the Service Request Information for the Orchestration](#) for more information.

B.5.1.1 Understanding the Application Stack Option in a Form Request

Use the Application Stack option to create a form request that establishes a session for a specific application and maintains that session across calls. With application stack processing, the form request can invoke multiple forms in different applications to complete a business process. Without application stack processing, each form in a form request is opened independently.

If you use the Application Stack option, the form request must include instructions for navigating between forms. Without the navigation, additional forms cannot be called.

Only values from the last form in the application stack can be returned; return controls specified for any form except the last form are ignored.

B.5.2 Creating a Service Request

To create a service request:

1. Access the Service Request design page:

- ❑ On the Orchestrator Home page, click the **Service Requests** icon.

OR

- ❑ After adding a Service Request step to an orchestration, click the **Service Request** step row.

The Orchestrator Studio displays the initial Service Requests design page.

2. On this page, click **Create Service Request**, and from the drop-down list, select the type of service request that you want to create:

- ❑ Form Request

- ❑ Custom

- ❑ Data Request

- ❑ Message

- ❑ Connector

3. (Studio 5.1.0) Click the **Product Code** drop-down list to select a product code to associate with the service request.

This gives an administrator the option to manage UDO security for orchestration components by product code.

4. On the *Service Request Type* design page, complete these fields:
 - ❑ **Service Request.** Enter a name for the service request. You should include the service request type, such as "form request" or "data request," in the name to distinguish it from other service requests listed in the Service Request design page.
 - ❑ **Short Description.** In the space provided, enter a description with a maximum of 200 characters. This description will appear below the service request name in the component list.
 5. (Optional) Click **Long Description** to provide additional details about the purpose of the service request.
 6. Click **Save**.
- The first time a new service request is saved, it is saved as a "Personal" UDO. After configuring the service request, you can use the UDO buttons described in the [User Defined Object \(UDO\) Features](#) section to move the service request to the appropriate status.
7. Refer to the appropriate section for instructions on how to configure the service request type: form request, custom Java, data request, message, or connector.

B.5.3 Configuring a Form Request

After creating the form request as described in [Creating a Service Request](#), perform these tasks:

- ❑ Load the EnterpriseOne Application Form Fields and Controls
- ❑ Configure Form Request Actions
- ❑ Configure the Order of Execution

Load the EnterpriseOne Application Form Fields and Controls

1. In the Available Actions area, complete the following fields:
 - ❑ Application
 - ❑ Form
 - ❑ Version
2. Click the **Form Mode** drop-down list and select the appropriate form mode: **Add**, **Update**, or **Inquiry**.

At runtime, the controls that appear on an EnterpriseOne form are dependent on the form mode as specified in Form Design Aid (FDA). The form mode ensures that you can see all of the form controls in the Orchestrator Studio that are available in the form at runtime.

3. (Optional) Click the **Application Stack** check box to enable application stack processing.

Use this option to create a form request that establishes a session for a specific application and maintains that session across calls. With application stack processing, the form request can invoke multiple forms in different applications to complete a business process. Without the application stack processing, each form in the form request is opened independently.

If you use the Application Stack option, the form request must include instructions for navigating between forms. Without the navigation, additional forms will not be called.
4. Click the **Load Form** button.

The Orchestrator Studio loads the controls and fields in the grid. The name of the form is displayed in the first row in the grid.

If a form request needs to invoke more than one application to complete the desired task, you can load controls and fields for additional application forms as needed.

Configure Form Request Actions

In the Available Actions area, shown in [Figure B–6](#), specify the application fields to which you want to map orchestration inputs, and specify the controls used to carry out the desired task or business process in EnterpriseOne.

Figure B–6 Available Actions Area in the Form Request Design Page

Available Actions		Application Stack <input checked="" type="checkbox"/>	Run Synchronously <input checked="" type="checkbox"/>	Bypass Form Processing <input type="checkbox"/>
Application	Form	Version	Version	Form Mode
Description	Mapped Value	Default Value	ID	
Condition-Based Alerts Revisions			P1311_W1311B	
Buttons and Exits				
Cancel			12	
Create W.O.			114	
Failure Analysis			124	
Investigation			120	
Investigation Msg			115	
Notification			119	

Use the following features in the Available Actions area to configure the fields and controls for a form request. After configuring each item, click the Add Action button at the end of each row to add it to the Order of Execution area.

- **Description** (informational only)

This column displays the controls and fields for each form in a collapsible/expandable parent node named after the EnterpriseOne form. Child nodes categorize other items and include a Buttons and Exits node, a node for displaying the available Query by Example (QBE) fields in a grid (if applicable), and a node for displaying all available columns in a grid.

- **Mapped Value**

Use this column to map inputs from the orchestration to fields in the EnterpriseOne form. In this column, enter a variable name for the input.

The only fields to which you can map inputs are EnterpriseOne editable fields. The variables names for the inputs in the form request should match the inputs defined in the orchestration to which you add the form request. If they do not match, you can use the "Transformations" feature to map input names after you add the form request to an orchestration. See [Adding Inputs to an Orchestration](#) and [Mapping Inputs in the Transformations Area](#) for more information.

Instead of a mapped value, you can enter a hard-coded value in the Default Value column or you can click the "Text substitution" check box to combine inputs into a text string.

This field is also available if you want to populate multiple rows in multiple grids. See [Populating Multiple Grids with Repeating Inputs](#) for more information.

□ **Default Value**

- For an editable field to which you want to map an input, use this column to enter a hard-coded value. You can also add a hard-coded value to use if the mapped value returns a null.
- For a check box or a radio button, you can select it to include it. If there are multiple radio buttons grouped together on a form, select only one. If you select more than one, only the last radio button in the Order of Execution list will be used.
- For a combo box control (a list of items in a drop-down menu), the Studio displays a drop-down menu in the Default Value column in which you can select the appropriate item from the list.

□ **Row Number for Update** (row)

If the task requires updating a particular row in an input capable grid, then map the appropriate input value containing the row number to the "Row Number for Update" row, or specify the row number in the Default Value column.

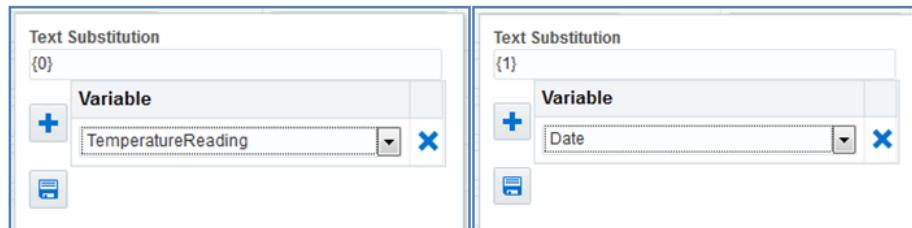
□ **"Text Substitution"** check box column

Use text substitution to combine input values into a text string for an input into an EnterpriseOne field. For example, you can create a text string such as "Temp was {0} at {1}" that contains input values for temperature and time.

Note: If you use text substitution for the input, the field cannot contain a mapped value. The substituted text string becomes the default value for the field when you add it to the Order of Execution list.

1. Click the **Add Text Substitution** button (plus symbol) in the row that contains the field for which you want to substitute the input with text.
2. In the Text Substitution field in the pop-up box, use a variable for the first input by entering {0} (with brackets).
3. In the Variable field, enter the input variable name.
4. Click the **Add** (plus symbol) to add the variable for the text substitution.
5. Add the next variable, {1}, and then in the next blank row below the Variable field, enter the input variable name.
6. Click **Add** to add it to the text substitution.
7. Click **OK** to save the variables.

The following image shows an example of variables added for a temperature input value and a date input value:



□ **ID** (informational only)

This column displays the ID of the control or field in EnterpriseOne.

☒ **Form Mode** (informational only)

If a form mode was selected, this column displays the form mode in the form row.

☒ **Return**

Select the check box in this column for fields that contain values that you want returned in the orchestration response. If you do not specify any return values, all values on the form will be returned. Return values may be used by customized third-party gateway programs or devices to do further processing based on the values returned from EnterpriseOne applications.

Caution: If using the Application Stack option:

- ☒ Only values from the last form in the application stack are returned; return controls specified for any form except the last form are ignored.
- ☒ If you use a form to add or update a value that upon saving exits to another form, such as a Find/Browse form, the return values will come from the last form that appears after the save. In this scenario, you need to add the last form to the application stack and select the return controls from the last form.

☒ **Variable Name**

Enabled after selecting the Return check box, use this field to enter a variable name for the return value. When you add the service request to an orchestration, this name appears in the orchestration inputs grid, which makes the returned value available for mapping to subsequent steps in the orchestration.

☒ **Return Hidden Fields** (left arrow icon in the Return column)

Use this feature to return data from "hidden" fields or grid columns not displayed in the Available Actions area. Hidden fields are fields that appear only after performing a particular action in a form. In the EnterpriseOne web client, use the field-level help (Item Help or F1 key) to identify the control IDs for hidden fields and grid columns. The control ID is displayed in the Advanced Options section in the help pop-up window.

To return a hidden field:

1. In the Available Actions area, in the row of the form that contains the hidden fields, click the left arrow in the Return column.

A dialog box appears displaying the control IDs and associated variable names of any return fields already selected in the Available Actions area.

2. For hidden fields, enter the control ID of the hidden field in the Return Form Ids field. Optionally, you can enter a variable name to represent the return value in the associated Name field.

For multiple controls, use a bar delimiter to separate the values in both fields, making sure the order of the values in each field match.

3. For hidden grid columns, enter the control ID for the grid column in the Return Grid Ids field. Optionally, you can enter a variable name to represent the return value in the associated Name field.

You must enter a variable name for the return value if you want it to appear in the orchestration inputs list in the orchestration.

An example of the notation for multiple grid columns in a grid is: 1[32, 34, 42]

Where 1 represents the first grid in the form (because some forms can have multiple grids), and 32, 34, 42 each represent a column in the grid.

Use a bar delimiter to separate the variable names, making sure the order of the variable names matches the order of the grid control IDs, as shown in the following example:

Value	ID	Version	Form Mode	Return
Return Form Ids				
8 16 22				
Form Id Names				
User ID Address Phone				
Return Grid Ids				
1[14,13,11,12]				
Grid Id Names				
UDC Hard Coded UDC Special Handling Description 01 Description 02				
<input type="button" value="OK"/> <input type="button" value="Cancel"/>				

□ Select First Row

For forms that require selecting the first row in a grid to perform a particular action, the grid displays a row named "Select First Row." Add this row to the Order of Execution if required for the desired task.

□ (Optional) "Options" icon (at the end of the first row in the grid)

This option contains settings that determine how the AIS Server processes the form request. See [Configuring Form Request and Data Request Processing](#) for details.

Configure the Order of Execution

After actions are added to the Order of Execution grid, you can reorder them using the up and down arrow buttons to the right of each row. You can also delete any actions using the Delete (X) button. [Figure B–7](#) shows the Order of Execution area.

Figure B–7 Order of Execution in the Service Request Design Page

Order of Execution			
Description	Action	Mapped Value	Default Value
Odometer	SetCheckboxValue	on	
Fuel Meter	SetCheckboxValue	on	
Hour Meter	SetCheckboxValue	on	
	SetCheckboxValue	on	
Subledge	SetCheckboxValue	on	
	SetCheckboxValue	on	

B.5.3.1 Populating Multiple Grids with Repeating Inputs

You can configure a form request to map repeating inputs into one or more grids.

In the Available Actions area, all editable grids have an editable Mapped Value column that will default to GridData. This value can be used to associate a set of repeating inputs to the grid. If you need to populate more than one grid in an orchestration, change this value to identify each grid. And then make sure that the "name" tag that identifies the repeating inputs in the detail inputs section of the orchestration input matches this value.

B.5.4 Configuring a Custom Service Request with Java (Advanced)

You can create a service request that uses custom Java to execute a custom process or to route data into another database. Before you can create a custom Java service request, you must create the custom Java as described in [Chapter 7, "Creating Custom Java for Orchestrations"](#).

(These steps have been updated in support of Orchestrator Studio 5.1.0.)

1. After naming the custom service request as described in [Creating a Service Request](#), select the **Java** option.
2. In the **Fully Qualified Class** field, enter the Java class.
This is the custom Java class that you created for the service request.
3. In the first grid, complete the following fields:
 - **Input**. Enter the name of the field in the class.
 - **Input Value**. Enter the input variable name. If the attribute is a boolean and you pass in "true", then you could use a default value.
 - **Default Value**. Enter a default, hard-coded value if there is no mapped value. You can also add a hard-coded value to use if the mapped value returns a null.
4. (Optional) In the second grid, enter a variable name for an output if you want to use the output value in subsequent steps in the orchestration or to another orchestration.
5. If you make a mistake, click the **Restore Custom Java** button to return the custom Java to its last saved state.
6. Click the **Save** button.

The Orchestrator Studio saves the custom Java request. You can then access the orchestration in the Orchestration design page and add this custom Java service request as a step in the orchestration.

B.5.5 Configuring a Custom Service Request with Groovy (Advanced) (Studio 5.1.0)

You can create a service request that uses Groovy scripting to execute a custom process or to route data into another database. To use Groovy for a service request, the Orchestrator Studio provides an edit area that contains a sample Groovy script with instructions on how to modify the script. You can use the Find and "Go to Line" fields and Undo and Redo buttons to work with the script. [Chapter 8, "Using Apache Groovy for Custom Service Requests, Rules, and Manipulating Output \(Orchestrator Studio 5.1.0 and Higher\)"](#) provides information on how to work with the sample Groovy script.

To configure a customer service request with Groovy:

1. After naming the custom service request as described in [Creating a Service Request](#), select the **Groovy** option.
2. Configure the script to perform the desired action.

3. In the "Input" grid, enter the names of the inputs.

The inputs will be placed in the inputMap of the "main" function and can be retrieved in the script by following the commented out example code.

4. Click the **Load Outputs** button.

This reads the script and adds any values added to the returnMap in the script to the Output grid.

5. (Optional) In the "Output" grid, enter a variable name for an output if you want to use the output value in a subsequent orchestration step.

When you add this service request to an orchestration, this name appears in the orchestration inputs grid, which makes the returned value available for mapping to subsequent steps in the orchestration.

Note: Even if no variables are used, all defined outputs are available for mapping using the Orchestration Output feature. See [Working with Orchestration Output](#) for more information.

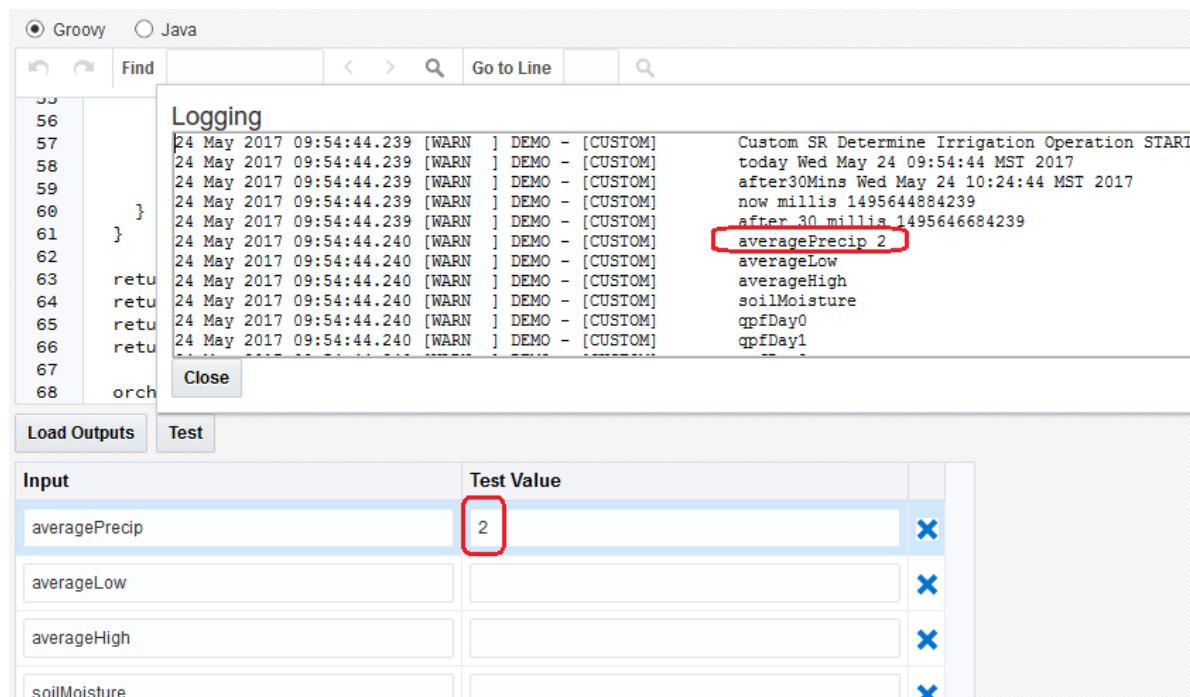
6. To test the script:

- a. Enter a value in the Test Value column for one or more inputs.
- b. Click the **Test** button.

Orchestrator Studio executes the script using the inputs you entered. If successful, it populates the results in the Test Output column of the Output grid.

If you included orchAttr.writeWarn or orchAttr.writeDebug statements in the script, a Logging popup displays after execution. At runtime, log statements are included in the AIS Server log, which can be used for debugging script issues.

The following example shows the Logging popup, which you can use to verify the values passed to the inputs:



The screenshot shows the JD Edwards EnterpriseOne Tools Orchestrator Studio interface. At the top, there are tabs for 'Groovy' and 'Java'. Below the tabs is a toolbar with icons for back, forward, find, go to line, and search. A 'Logging' window is open, displaying a log of messages from May 24, 2017, at 09:54:44.239. The log includes several 'WARN' entries and some custom log statements. One entry for 'averagePrecip' is highlighted with a red box. Below the Logging window is an 'Input' grid. The grid has two columns: 'Input' and 'Test Value'. The 'Input' column lists 'averagePrecip', 'averageLow', 'averageHigh', and 'soilMoisture'. The 'Test Value' column contains the value '2' for 'averagePrecip', and empty fields for the other three inputs. There are also 'X' buttons in the grid for deleting rows.

Input	Test Value
averagePrecip	2
averageLow	
averageHigh	
soilMoisture	

7. Click the **Save** button.

The Orchestrator Studio saves the custom Groovy service request. You can then access the orchestration in the Orchestration design page and add this custom Groovy service request as a step in the orchestration.

B.5.6 Configuring a Data Request

You can configure a data request to:

- Query and return data from an EnterpriseOne table or business view.
- Perform an aggregation of data from a table or business view and return aggregate amounts.

In a standard data request that returns data directly from a table or business view, you can use variables to define the return fields. In a data request that performs an aggregation, you can use variables to define the "group by" fields and the returned aggregate amounts. You define a variable by typing a name next to the field or item that represents the returned value.

When you add a data request to an orchestration, any variables that you define are automatically added to the orchestration as inputs. This enables you to map the returned data to subsequent steps in an orchestration.

Important: Variables are only supported in the first row of a response. If a query contains multiple rows, only the values from the first row will be available as variables.

After creating a data request as described in [Creating a Service Request](#), perform these tasks:

- [Load the Fields from a Table or Business View](#)
- [Configure a Data Request to Return Field Data](#)
- or
- [Configure a Data Request to Perform an Aggregation](#)

Load the Fields from a Table or Business View

You need to know the name of the table or business view that contains the data you want returned. If you do not know the table or business view name, but you know the application that uses the table or business view, you can use the JD Edwards EnterpriseOne Cross Reference Facility (P980011) to identify the table or business view associated with the application. From the Cross Reference form, click the "Interactive Applications" tab, then click the "Business Views Used by an Interactive Application" or "Tables Used by an Interactive Application." If you do not have access to the Cross Reference Facility, ask an administrator to look up this information for you.

1. In the Available Actions area, enter the name of the table or business view in the Table/View Name field.
2. Click the **Load** button.

The Orchestrator Studio loads all fields from the table or business view into the grid.

Configure a Data Request to Return Field Data

After loading fields from a table or business view, set up filtering criteria and specify the fields that contain data that you want returned.

For example, a business analyst wants to add a data request to an orchestration that returns a customer's credit limit amount. The orchestration has "Customer Number" defined for one of its inputs. To configure the data request, the business analyst:

- ❑ Sets up filter criteria with a condition that filters on Address Number.
- ❑ Selects the Credit Limit and Alpha Name fields for the return data.
- ❑ Adds the data request to the orchestration, mapping the Customer Number input in the orchestration to the Address Number in the data request.

When the data request is added to the orchestration, the Credit Limit and Alpha Name fields automatically become additional inputs to the orchestration, which the business analyst can then map to subsequent orchestration steps.

To configure a data request to return field data:

1. Define the filtering criteria for the data request:
 - a. In the Fields grid on the left, click the "**Filter**" icon next to the field or fields that contain data that you want to filter on.
The Orchestrator Studio displays each field in the Filter Criteria area on the right.
 - b. For each field in the Conditions box, select the appropriate operand and in the adjacent field, enter a hard coded value or a variable.

A variable appears in the field by default. If you modify the variable name, make sure the syntax includes the \$ sign and brackets, for example:

`${Address Number 1}`

- c. Select the **Match All** or **Match Any** option to determine how the conditions are applied.

You can also click the **Options** button, and from the Query drop-down list, select a predefined query to use for the filtering criteria. You can use a query instead of or in combination with the filtering criteria defined in a data request. The queries that you can see in this list are based on UDO security permissions.

2. Specify the data you want returned:
 - a. In the Fields grid, click the "**Return**" icon next to each field for which you want data returned.
Each field appears in the "Return Fields and Variable Names" on the right.
 - b. (Optional) You can use a variable for the return by entering a name for the variable in the adjacent blank field.

Note: Return variables do not need the \${ } notation. This notation is only necessary for input variables to distinguish between a variable and a hard coded value.

When you add a data request to an orchestration, these variables are automatically added to the orchestration as inputs, which you can use to map return data to subsequent orchestration steps.

3. (Optional) For the return data, determine the order by which you want the data returned:
 - a. In the grid on the left, select the **Order By** icon next to any field that was selected for the return data.
The Orchestrator Studio adds the field name to the Order By area on the right.

- b. In the drop-down list next to the field name, select **Ascending** or **Descending**.
- 4. (Optional) Click the **Options** button to configure settings that control how the AIS Server processes a data request at runtime. See [Configuring Form Request and Data Request Processing](#).

Configure a Data Request to Perform an Aggregation

To configure a data request to return aggregate amounts:

1. After loading the fields from a table or business view, slide the **Aggregation** toggle to the right.

This enables the aggregation features in the Fields grid.

2. Click the "**Filter**" icon next to the fields that contain the data that you want to filter on.

The Orchestrator Studio displays each field in the Conditions area on the right.

3. Set up conditions for filtering data:

- a. For each field in the Conditions box, select the appropriate operand and in the adjacent field, enter a hard coded value or a variable.

A variable appears in the field by default. You can modify the variable name, but you must use the \$ sign and brackets in the syntax, for example:

`${Address Number 1}`

- b. Select the **Match All** or **Match Any** option to determine how the conditions are applied.

You can also select the Options button, and from the Query drop-down list, select a predefined query to use for the filtering criteria. You can use a query instead of or in combination with the filtering criteria defined in a data request. The queries that you can see in this list are based on UDO security permissions.

4. Click the "**Aggregate**" icon next to the fields that contain the data for the aggregation.

5. In the pop-up window, select the type of aggregate that you want to perform.

The aggregate options displayed depend on whether the field contains a numeric value or a string.

6. (Optional) In the Aggregations and Variable Names section, click the **Include Count** check box if you want the response to include a count of the records returned. To use the returned count in a subsequent orchestration step, enter a variable name in the adjacent field.

7. (Optional) Use the following features in the Fields grid to further define the data aggregation:

- **"Having" icon.** Click this icon next to a field for which you want to provide additional filtering criteria on an aggregation.

The Data Request design page displays the field in the Having section on the right.

- a. Click the drop-down list next to the field and select the operand.

- b. In the adjacent field, enter a hard coded value or variable.

- **"Group By" icon.** Click this icon next to any field that you want the aggregate results grouped by. In the "Group By and Variable Name" section on the right, you can enter a variable name.

Using "Group By" may result in multiple rows being returned, one for each unique combination of a group. For example, you might return total orders for an entire year

for customers, and group the results by customer. In this case, the data request response will return a unique row for each customer with the customer's total orders for the year.

Note: Variables are only supported in the first row of a response. If a query contains multiple rows, only the values from the first row will be available as variables.

- **"Order By" icon.** For any fields used in an aggregate or "group by," click this icon to arrange return data in ascending or descending order.
- 8. (Optional) Click the **Options** button to configure settings that control how the AIS Server processes a data request at runtime. See [Configuring Form Request and Data Request Processing](#).

B.5.6.1 Viewing and Copying JSON Code of a Data Request

After configuring a data request, you can click the **Preview** button to view and copy the JSON code for a data request. Developers can use this code to develop AIS client applications that can make data request calls directly to the AIS Server rather than through an orchestration. See the *JD Edwards EnterpriseOne Application Interface Services Client Java API Developer's Guide* for more information.

B.5.7 Configuring a Message Request

Add a message request to an orchestration to send emails to external email systems or the EnterpriseOne Work Center email system. The following EnterpriseOne Work Center and workflow features are supported in a message request:

- Workflow distribution lists to send messages to a predefined group of EnterpriseOne users.
- Message templates from the data dictionary that contain boilerplate text and values related to a particular business process.
- Shortcuts to EnterpriseOne applications.

For more information about these workflow features, see the *JD Edwards EnterpriseOne Tools Workflow Tools Guide*.

In a message request, you can include variables to pass data from an orchestration input into a message request. You can include variables in the recipient fields, the subject and body, a message template, and a shortcut.

To configure a message request:

1. Create and name the message request as described in [Creating a Service Request](#).
2. To enter recipients (To, Cc, or Bcc), select a recipient type:
 - Select **Address Book** to send a message to a single EnterpriseOne user. Enter the address book number of the EnterpriseOne user.
 - Select **Contact** to send a message to an individual in a user's contact list. Enter the address book number of a user and then the number of the contact.
 - Select **Distribution List** to send a message to the members of an EnterpriseOne distribution list. Enter the address book number of the list in the Parent Address Number field and select its structure type. For more information about structure types, see "Structure Types" in the *JD Edwards EnterpriseOne Tools Workflow Tools Guide*.
 - Select **Email** to enter an email address for the recipient.

3. In the Subject and body fields, enter text, variables, or a combination of both. To insert variables, see step 6.
4. To include boilerplate text from a message template in the data dictionary:
 - a. Expand the Data Dictionary Text section.
 - b. In the Data Item field, enter the name of the message template data item and click **Load**.
 - c. If the message template contains variables, use the grid in the right to override the variables with text substitution.
5. To include a shortcut to an application:
 - a. Expand the JD Edward EnterpriseOne Shortcut section.
 - b. Complete the Application, Form, and Version fields to specify the form that you want the shortcut to launch.
 - c. Click **Load**.
 - d. In the grid, you can use variables to pass in data to the application when the application is launched from the shortcut.
6. To include variables in the recipient types, subject, body, message template text, or shortcut:
 - a. Right-click in a field and click **Insert Variable**.
 - b. In the variable that appears, rename variable as appropriate, but make sure the syntax includes the \$ sign and brackets, for example:
 \${creditmanager}

B.5.8 Configuring a Connector

In the Orchestrator Studio, you can configure a connector service request to enable an orchestration to:

- Invoke another orchestration on your local system or on an AIS Server on another EnterpriseOne system in a multisite operation.

- ❑ Invoke a REST service. (Orchestrator Studio 5.1.0) This enables outbound REST calls to external systems as an orchestration step. For example, an orchestration could make a REST call to a Cloud service and use the data in the response in subsequent orchestration steps.

When you configure a connector, you select a **connection** to the system where the orchestrations or REST service resides. This provides a list of orchestrations or the REST service that you can configure a connector service request to invoke.

Before you can create a connector service request, an administrator must set up a connection to the system with the orchestrations or REST service. See [Creating Connection Soft Coding Records for Connector Service Requests](#) for details.

B.5.8.1 Configuring a Connector to Invoke an Orchestration

1. Create and name the connector service request as described in [Creating a Service Request](#).
2. On the Connector design page, click in the **Connection** field and select an Orchestrator connection.

Connections to orchestrations appear under "Orchestrator." After you select a connection, the Orchestrator Studio displays the URL to the AIS Server next to the Connection field.

3. You can click the **Test Connection** button to test the connection to the AIS Server.
4. Select an orchestration from the **Orchestration** drop-down list.

The inputs in the orchestration appear in the Orchestration Input column. A variable for each input is automatically generated in the adjacent Input column.

5. You can modify the variable names as desired or map a hard coded value by entering the value (without the \${ } notation).

B.5.8.2 Configuring a Connector to Invoke a REST Service (Studio 5.1.0)

To configure a connector to a REST service, you specify an HTTP method and then provide the following additional details to complete the request:

- ❑ The path to a particular endpoint in the REST service.
- ❑ Parameters, if required by the endpoint.
- ❑ Key-value pairs for the HTTP header.

In the Orchestrator Studio, you can test the connector to view the JSON response of the REST service call. You can also use Groovy scripting to refine the data in the connector output. For example, you can configure a Groovy script to refine the contents of the REST service response so that the connector output contains only the data required by the consuming device or program.

To configure a connector to invoke a REST service:

1. Create and name the connector service request as described in [Creating a Service Request](#).
2. On the Connector design page, click the **Connection** field and select a connection under the REST category.

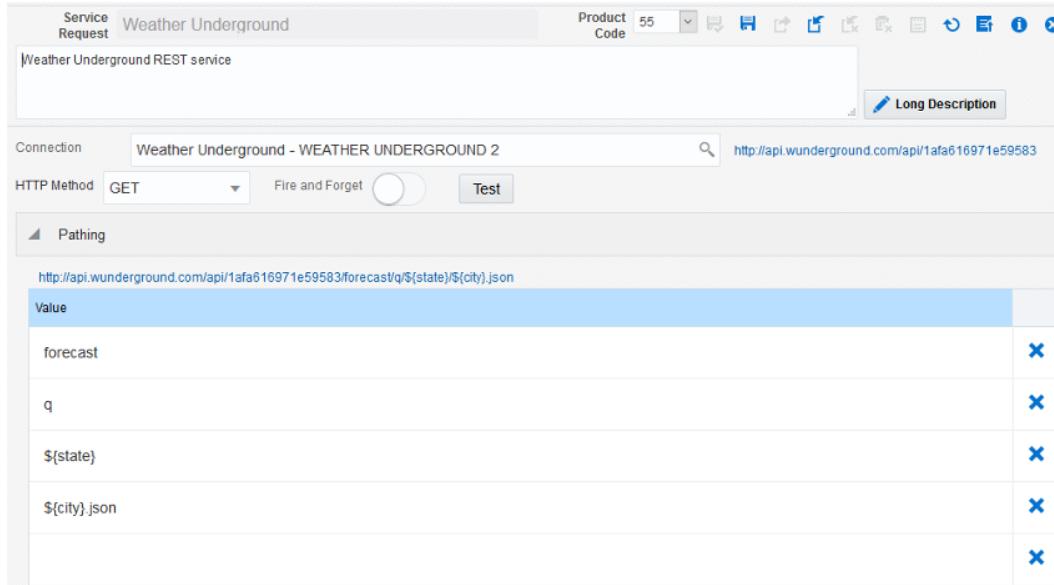
The Orchestrator Studio displays the URL to the REST service available through the connection.

3. Click the **HTTP Method** drop-down list and select the appropriate method.

4. Slide the **Fire and Forget** toggle to the right if you want the orchestration to execute without waiting for a response. If Fire and Forget is enabled, the Output section is hidden because the REST service response is not processed.
5. Expand the Pathing section and use the Value grid to build the path to a REST service endpoint.

Each value you enter in the Value grid is appended to the path, separated by a forward slash (/). You can use variables in the path by adding a value inside \${ }. Click the **Delete** button at the end of a row to delete any values from the path.

The following image shows an example of a path to an endpoint that includes variables.



6. If the REST service takes parameters, use the Parameters section to append parameters to the endpoint.

Parameters are appended to the endpoint after a "?" and are separated by a "&". You can include variables by specifying a value inside \${ }.

7. In the Headers section, enter key-value pairs for the header in the Key and Value columns. You can also choose any known values from the drop-down menu in each column.
8. In the Body section (which is not used or displayed if the HTTP method is GET or TRACE), if the HTTP method requires it, specify a payload to send to the REST service.
9. In the Output section, use the following sections to identify the outputs:

Note: The outputs are based on the JSON response of the REST service call. You can see the response after running a successful test of the REST service using the Test button.

- Manipulate Output (advanced). Use Groovy scripting to manipulate the output of the REST call. For example, if you only need certain data in the output, you can use Groovy scripting to refine the data displayed in the output. See [Groovy Template for Manipulating Output from a REST Connector Response](#) for more information.
- Output grid. If the value you want is nested in JSON objects, you can get to the value by separating each JSON object name by a period (.). If the value you want is in an array, you can access it by adding the proper index of the array inside [].

The following image shows an example of outputs defined in the Output section:

Output	Variable Name	
forecast.simpleforecast.forecastday[0].qpf_allday.in	QPFDay0	X
forecast.simpleforecast.forecastday[1].qpf_allday.in	QPFDay1	X
forecast.simpleforecast.forecastday[2].qpf_allday.in	QPFDay2	X

10. To test the connector:

- Click the **Test** button.
- If the connector includes variables, enter values for the variables in the popup box.
If successful, the Orchestrator Studio displays the response.
- Click the **Show Output** button displayed at the end of the response to apply the instructions defined in the Output section, including the Groovy script if specified.
Use the results to validate the path to the output values specified in the Output section.

B.5.9 Configuring Form Request and Data Request Processing

In the Orchestrator Studio, you can configure settings that control how the AIS Server processes a form request or data request in an orchestration at runtime.

For form requests, these settings are available from the Options icon at the end of the first row of each form listed in the Available Actions grid.

For data requests, these settings are available from the Options button on the Data Request design page.

The settings include:

- **Maximum Records.** 0 is for All Records.
- **Query Name.** From this drop-down list, you can select a predefined query to use for the filtering criteria. You can use a query instead of or in combination with the filtering criteria defined in a data request. The queries that you can see in this list are based on UDO security permissions.
- **Output Type.** Select one of the following format types for the format of the JSON response. Formats include:
 - **Default.** The default format is Version 2 output type for version 1 orchestrations (orchestrations created prior to Orchestrator Studio 5.0.1). The default format is Grid Data output type for version 2 orchestrations (orchestrations created in Orchestrator Studio 5.0.1 and higher).
 - **Grid Data.** Returns grid data in simplified name-value pairs. This is the default output type for version 2 orchestrations (orchestrations created in Orchestrator Studio 5.0.1 and higher).
 - **Version 2.** Returns cell-specific information for each grid row as well as information about each column such as column ID, whether it is visible, editable, and so forth.

This is the default output type for orchestrations created prior to Orchestrator Studio 5.0.1, which are referred to as version 1 orchestrations.

- **Stop on Warning** check box (form requests only). Click this check box if you want the processing to stop if the invoked EnterpriseOne form produces a "Warning" message. Keep this check box clear if you want processing to continue after a warning message.
- **Turbo Mode** (form requests only). Use this option to reduce processing time of form requests. See "Using Turbo Mode" in the *JD Edwards EnterpriseOne Application Interface Services Client Java API Developer's Guide* for more information.
- **Caching: Allow**. Click this check box to enable caching of the service request response. If the "Enable Caching by Default" option is enabled in Server Manager, this parameter is ignored and all responses for Read operations will be cached.
- **Caching: Force Update**. Click this check box to force the system to fetch the data from the database for the service request. If the "Enable Caching by Default" setting is enabled in Server Manager, then this parameter for the individual service request is ignored.
- **Caching: Time Stored - Milliseconds**. Enter the amount of time in milliseconds for the service request to use the cache for the response. This setting overrides the value in the "Default Read Cache Time To Live" setting in Server Manager.

B.6 Creating Rules

This section contains the following topics:

- [Section B.6.1, "Understanding Rules"](#)
- [Section B.6.2, "Creating a Rule"](#)
- [Section B.6.3, "Creating a Custom Rule with Java \(Advanced\)"](#)
- [Section B.6.4, "Creating a Custom Rule with Groovy \(Advanced\) \(Studio 5.1.0\)"](#)

B.6.1 Understanding Rules

A rule contains conditional logic that the Orchestrator uses to evaluate conditions, such as true or false conditions that determine how the orchestration processes the incoming data. You can define a rule with a list of conditions or you can define a more complex rule using Groovy (Studio 5.1.0) or a custom Java class.

An orchestration rule with conditions functions similar to an EnterpriseOne query in that each rule has:

- A "Match Any" or "Match All" setting.
- One or more conditions defined, each being a binary operation on two values.

After creating a rule and adding it to an orchestration, you use the Action column in the Orchestration design page to determine the orchestration step that is invoked when the conditions in the rule are met. See [Defining the Actions Between a Rule and Dependent Components](#) for more information.

B.6.2 Creating a Rule

Create a rule component to define the conditions for an orchestration.

To create a rule:

1. Access the Rules design page:

- On the Orchestrator Home page, click the **Rules** icon. And then on the Rules design page, click the **New Rule** button.

OR

- After adding a Rule step to the grid in the Orchestration design page, click the **Edit** button next to the step.

The Orchestrator Studio displays the Rule design page.

2. In the Rule field, enter a name for the rule.
3. (Studio 5.1.0) Click the **Product Code** drop-down list to select a product code to associate with the rule.

This gives an administrator the option to manage UDO security for orchestration components by product code.

4. In the space provided, enter a short description with a maximum of 200 characters. This description appears below the rule name in the component list.
5. Click the **Edit Long Description** button to add a long description to provide more detail about the purpose of the component.
6. Select the Match Value drop-down menu and select one of the following values:
 - **Match All**. Select this option if all conditions in the rule must be met.
 - **Match Any**. Select this option if any conditions in the rule can be met.
7. In the first row in the grid, complete the following columns to add a condition:
 - **Rule Type**. Click the drop-down menu and select String, Numeric, or Date depending on the format of the input.
 - **Value 1**. Enter a variable for the input name.
 - **Operator**. In the drop-down menu, select from the list of operands which include: >, <, >=, <=, =, !=, startsWith, endsWith, contains, between, inList.
 - **Literal**. Click this check box to indicate a literal value.
 - **Value 2**. If you selected the Literal check box, manually enter a value.
 - **Literal Value Type**. If you selected the Literal check box, click the drop-down menu and select the format of the literal value: string, numeric, data.
8. Add additional conditions to the rule as needed. If you add a condition by mistake, you can delete it by clicking the **Remove** button at the end of the row with the condition.
9. Click the **Save** button.

The first time a new rule is saved, it is saved as a "Personal" UDO. Thereafter, you can use the UDO buttons described in the [User Defined Object \(UDO\) Features](#) section to move the rule to the appropriate status.

After adding a rule and a service request to an orchestration, in the Orchestration design page, you must define the action for the rule to invoke the service request. See [Defining the Actions Between a Rule and Dependent Components](#) for more information.

B.6.3 Creating a Custom Rule with Java (Advanced)

You can create complex rule using custom Java. Before you add a custom Java rule in the Orchestrator Studio, you must create a custom Java class to use for the rule as described in [Chapter 7, "Creating Custom Java for Orchestrations"](#).

To create a custom Java rule:

1. Click the **Rules** icon on the Orchestrator Studio Home page.
2. On the Rules design page, click the **New Custom** button.
3. Select the **Java** radio button.
4. In the Rule field, enter a name for the custom rule.
5. (Studio 5.1.0) Click the **Product Code** drop-down list to select a product code to associate with the rule.

This gives an administrator the option to manage UDO security for orchestration components by product code.

6. In the space provided, enter a short description with a maximum of 200 characters. This description appears below the rule name in the component list.
7. Click the **Edit Long Description** button to add a long description to provide more detail about the purpose of the component.
8. In the Fully Qualified Class field, enter the fully qualified path to the Java class, which includes the name of the Java class.

This is the custom Java class that you created to use for the rule.

9. Complete the following fields in the grid:
 - **Attribute.** Enter the name of the field in the class.
 - **Input Value.** Enter a variable for the input name. If the attribute is a boolean and you pass in "true", then you could enter a hard-coded default value.
 - **Default Value.** Enter a hard-coded value if there is no mapped value. You can also add a hard-coded value to use if the mapped value returns a null.
10. If you make a mistake while configuring any of the fields, you can click the **Restore Rule** button which refreshes the custom Java rule to its last saved state.
11. Click the **Save** button.

B.6.4 Creating a Custom Rule with Groovy (Advanced) (Studio 5.1.0)

Use Groovy to create a custom rule when conditions in a standard rule will not suffice.

To create a custom rule with Groovy:

1. Click the **Rules** icon on the Orchestrator Studio Home page.
2. On the Rules design page, click the **New Custom** button.
3. In the Rule field, enter a name for the custom rule.
4. Click the **Product Code** drop-down list to select a product code to associate with the rule.
5. In the space provided, enter a short description with a maximum of 200 characters. This description appears below the rule name in the component list.
6. Click the **Edit Long Description** button to add a long description to provide more detail about the purpose of the component.
7. Select the **Groovy** radio button.

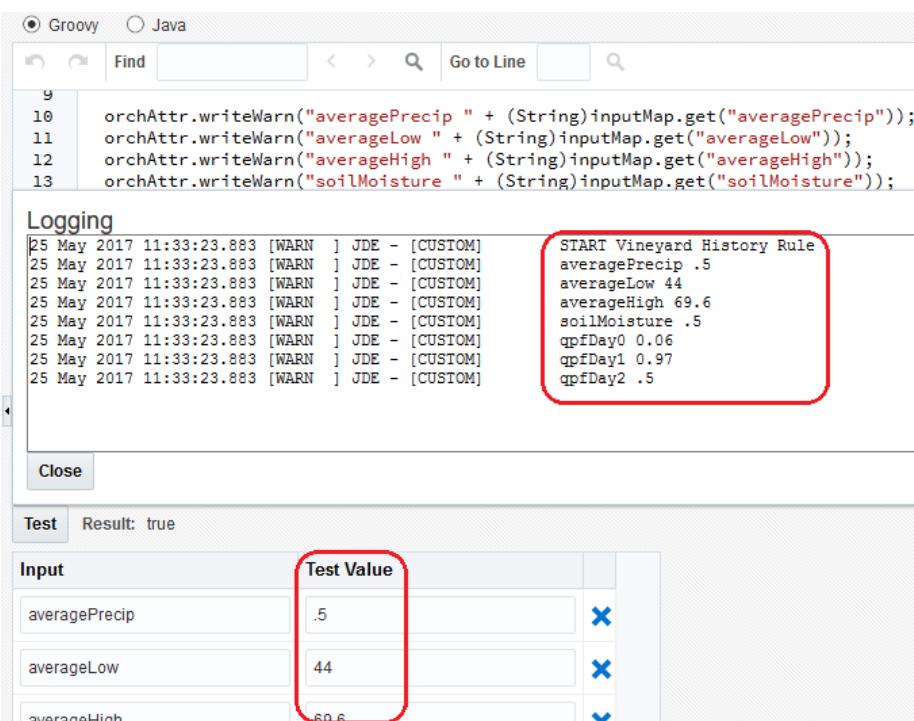
The Orchestrator Studio displays an edit area that contains a sample groovy script with instructions on how to work with the script. Use the Find and "Go to Line" fields and Undo and Redo buttons to help edit the script. See [Chapter 8, "Using Apache Groovy for Custom](#)

[Service Requests, Rules, and Manipulating Output \(Orchestrator Studio 5.1.0 and Higher\)](#)" for more information about the sample Groovy script.

8. Configure the script to perform the desired action.
9. In the "Input" grid, enter the names of the inputs.
10. To test the script:
 - a. Enter a value in the Test Value column for one or more inputs.
 - b. Click the **Test** button.

If you included orchAttr.writeWarn or orchAttr.writeDebug statements in the script, a Logging popup displays after execution. At runtime, log statements are included in the AIS Server log, which can be used for debugging script issues.

A Logging dialog box displays the test results which you can use to verify the values passed to the inputs, as shown in the following example:



11. Click the **Save** button.

B.7 Creating Cross References

This section contains the following topics:

- ❑ [Section B.7.1, "Understanding Cross References"](#)
- ❑ [Section B.7.2, "Creating a Cross Reference"](#)

B.7.1 Understanding Cross References

The Orchestrator uses a cross reference component to map incoming data (third-party data) to an EnterpriseOne value. The EnterpriseOne value identified in the cross reference is considered the output of the cross reference. The output from the cross reference becomes the data passed to another orchestration step.

For each cross reference that you create in the Orchestrator Studio, you have to create a cross reference record in P952000 in EnterpriseOne. When defining cross reference records for orchestrations in P952000, you must use the "AIS" cross reference type. For more information on how to create these cross reference records in P952000, see [Chapter 5, "Setting Up Cross References and White Lists in EnterpriseOne \(P952000\)"](#) in this guide.

Note: If a cross reference lookup fails in an orchestration, the orchestration is terminated.

B.7.2 Creating a Cross Reference

You must define the orchestration inputs before you can create a cross reference. See [Adding Inputs to an Orchestration](#) for more information.

To create a cross reference:

1. Access the Cross Reference design page:

- On the Orchestrator Home page, click the **Cross References** icon. And then on the Cross References design page, click the **New Cross Reference** button.
- OR
- After adding a Cross Reference step to the grid in the Orchestration design page, click the **Edit** button next to the step.

The Orchestrator Studio displays the Cross Reference design page.

2. In the Cross Reference field, enter a name for the cross reference.
3. In the space provided, enter a short description with a maximum of 200 characters. This description appears below the cross reference name in the component list.
4. Click the **Edit Long Description** button to add a long description to provide more detail about the purpose of the component.
5. In the Object Type field, enter a name for the object type.

This is the object type used to categorize the orchestration cross references in EnterpriseOne. See "Adding Orchestration Cross References" in the *JD Edwards EnterpriseOne Tools Interoperability Guide* for more information.

6. (Studio 5.1.0) Click the **Product Code** drop-down list to select a product code to associate with the cross reference.

This gives an administrator the option to manage UDO security for orchestration components by product code.

7. Complete the Input Key and Output Key columns to map the inputs to EnterpriseOne fields:
 - **Input Key.** The inputs can be one or many values. The inputs must correspond to the value or pipe (|) delimited values in the Third Party Value column in P952000. See ["Chapter 5, "Setting Up Cross References and White Lists in EnterpriseOne \(P952000\)"](#) for more information.
 - **Output Key.** The outputs can be one or many values. The outputs must correspond to the value or pipe (|) delimited values in the EOne Value column in P952000. See ["Chapter 5, "Setting Up Cross References and White Lists in EnterpriseOne \(P952000\)"](#) for more information.
8. If you enter any values by mistake, you can click the **X** button next to the field to delete the entry.

9. Click the **Save** button, which saves the white list as a "Personal" UDO.

The first time a new cross reference is saved, it is saved as a "Personal" UDO. Thereafter, you can use the UDO buttons described in the [User Defined Object \(UDO\) Features](#) section to move the rule to the appropriate status.

The output values in the cross reference are now available for mapping in subsequent orchestration steps.

B.8 Creating White Lists

This section contains the following topics:

- ¤ [Section B.8.1, "Understanding White Lists"](#)
- ¤ [Section B.8.2, "Creating a White List"](#)

B.8.1 Understanding White Lists

A white list contains a list of IDs permitted in the Orchestrator. By adding a white list to an orchestration, only inputs with IDs listed in the white list are permitted for processing by the Orchestrator. You add a white list component as a step in the orchestration.

In addition to specifying the permitted IDs in the white list, you must also specify the white list IDs in P952000 in EnterpriseOne. This is the same application that you use to set up and store orchestration cross references. As with cross references, use the "AIS" cross reference type in P952000 for a white list. In P952000, the Third Party App ID should have a value of WHITELIST. The EnterpriseOne value is not used for white lists and will default to NA.

If a white list lookup fails in an orchestration, the orchestration is terminated.

B.8.2 Creating a White List

1. Access the White List design page:

- ¤ On the Orchestrator Home page, click the **White Lists** icon. And then on the White Lists design page, click the **New White List** button.

OR

- ¤ After adding a White List step to the grid in the Orchestration design page, click the **Edit** button next to the step.

The Orchestrator Studio displays the White Lists design page.

2. In the White Lists design page, click the **New White List** button.
3. In the White List, enter a name for the white list.
4. In the space provided, enter a short description with a maximum of 200 characters. This description appears below the white list name in the component list.
5. Click the **Edit Long Description** button to add a long description to provide more detail about the purpose of the component.
6. In the Object Type field, enter a name for the object type.

The value you enter in the Object Type field must match a cross reference object type in the EnterpriseOne Business Services Cross Reference application (P952000).

The cross reference object type is a named group of records in P952000. For example, you may have thousands of records in P952000. You can use cross reference object types, for example "Equipment" or "Alert_Notification_Recipients" to group cross reference records into manageable categories. You define the cross reference object types as needed. See

[Chapter 5, "Setting Up Cross References and White Lists in EnterpriseOne \(P952000\)"](#) for more information.

7. (Studio 5.1.0) Click the **Product Code** drop-down list to select a product code to associate with the white list.

This gives an administrator the option to manage UDO security for orchestration components by product code.

8. In the Input Key column, enter the input that you want to add as a permitted input.
9. Click the **Save** button.

The first time a new white list is saved, it is saved as a "Personal" UDO. Thereafter, you can use the UDO buttons described in the [User Defined Object \(UDO\) Features](#) section to move the white list to the appropriate status.

B.9 Creating Orchestrations

This section contains the following topics:

- [Section B.9.1, "Understanding Orchestrations"](#)
- [Section B.9.2, "Creating an Orchestration"](#)
- [Section B.9.3, "Adding Inputs to an Orchestration"](#)
- [Section B.9.4, "Adding Steps to an Orchestration"](#)
- [Section B.9.5, "Mapping Inputs in the Transformations Area"](#)
- [Section B.9.6, "Updating Version 1 Orchestrations to Version 2 Orchestrations"](#)
- [Section B.9.7, "Working with Orchestration Output"](#)

B.9.1 Understanding Orchestrations

Creating an orchestration in the Orchestrator Studio involves:

- Naming the orchestration and specifying the input format for the orchestration. The input format can be JDE Standard, Oracle Cloud IoT, or Generic. See [Supported Input Message Formats](#) for more information about which input format to use.
- Adding inputs to the orchestration.

The inputs define the data passed from the device to the orchestration. This may include an ID that identifies the device as well as other data that the orchestration will receive from a device, such as temperature, date, or any other data you want to capture. Each input has a name and a type which can be string, numeric, or various date formats.

- Adding steps to the orchestration. Each step is a component of the orchestration: a service request, rule, cross reference, or white list. At a minimum, an orchestration requires only a service request step, which provides the actions or the instructions to perform a particular task in EnterpriseOne.
- Configuring transformations.

You use transformations to map orchestration inputs to inputs defined in each orchestration step, such as inputs in a rule, cross reference, white list, or service request component.

Important: Remember that when you are ready to "request to publish" an orchestration, you need to make sure that you also request to publish all components associated with the orchestration. This enables an administrator to save all required components to the proper directory on the AIS Server for processing by the Orchestrator. If a component is missing, the orchestration process will end in error.

B.9.2 Creating an Orchestration

To create an orchestration:

1. On the Orchestrator Studio Home page, click the **Orchestrations** icon.
2. On the Orchestrations page, click the **New Orchestration** button.
3. On the Orchestration design page, enter a unique name for the orchestration in the Orchestration field.

All orchestrations created in Orchestrator Studio 5.0.1 or higher are saved as version 2 orchestrations. You cannot change this value in the Orchestrator Version drop-down list. If you want to update a version 1 orchestration created in a previous version of the Orchestrator Studio, see [Updating Version 1 Orchestrations to Version 2 Orchestrations](#).

Note: When you save an orchestration, the orchestration name is used to define an endpoint on the AIS Server. The endpoint URL is:

`http://<server>:<port>/jderest/orchestrator/<orchestrationname>`

To invoke this orchestration, third-party applications or devices use a post operation to this url, where <orchestrationname> is the name of the orchestration.

4. Click the **Product Code** drop-down list to select a product code to associate with the orchestration. (Studio 5.1.0)

This gives an administrator the option to manage UDO security for orchestration components by product code.

5. In the space provided, enter a short description with a maximum of 200 characters. This description appears below the orchestration name in the component list.

6. Click the **Edit Long Description** button to provide more detail about the component.

Use this field to describe the purpose of the orchestration and any details that differentiate the orchestration from other orchestrations that you create.

7. Click the **Input Format** drop-down menu and select the appropriate format:

- JDE Standard** (default)
- Oracle Cloud IoT**
- Generic**

See [Supported Input Message Formats](#) for more information about which input format to use.

8. At this point, you can click **Save** before defining inputs or steps for the orchestration.

The Orchestrator Studio saves the orchestration as a "Personal" UDO. Next, add inputs to the orchestration as described in the [Adding Inputs to an Orchestration](#) section.

You can also use the Save As button and rename an existing orchestration to create a new one.

Caution: If you use Save As to create a copy of an orchestration, only the orchestration is copied. The Orchestrator Studio does NOT create a copy of the components that are associated with the orchestration's steps. That is, both the original orchestration and the new orchestration use the same components that comprise the orchestration. Therefore, in the new orchestration, do NOT modify the components in any way that would break other orchestrations that use the same components.

B.9.3 Adding Inputs to an Orchestration

Orchestration inputs identify the data an orchestration consumes from external devices. In the orchestration, you enter names for the inputs. For example you can enter "SensorID" to pass a sensor ID value to an orchestration and you can enter "TemperatureReading" to pass a temperature value to the orchestration.

You also use these orchestration inputs to configure other components used by the orchestration, such as a service request, rule, white list, or cross reference. For example, if the orchestration requires a rule, you use the orchestration inputs to define the conditions for the rule. If an incoming value from a device does not match an EnterpriseOne value, you can create a cross reference that uses the orchestration input to map third-party data to data in EnterpriseOne.

To add the orchestration inputs:

1. In the first empty row in the Orchestration grid, enter the name of the input in the Input column.
2. In the Value Type column, select the input value type. Valid values are:
 - ❑ String
 - ❑ Numeric

If the input is a date, you can use any of the following date formats:

- ❑ dd/MM/yyyy
- ❑ dd/MM/yy
- ❑ yyyy/MM/dd
- ❑ MM/dd/yyyy
- ❑ MM/dd/yy
- ❑ yy/MM/dd

You can also use the following date formats, which create additional inputs derived from the passed value:

- ❑ Milliseconds
 - ❑ yyyy-MM-dd 'T' HH:mm:ss .SSSZ
3. Click **Save** to save your changes.

B.9.4 Adding Steps to an Orchestration

When you add a service request, rule, cross reference, or white list to an orchestration, you add it as a step in the orchestration. It is recommended that you create the component before adding it as a step in an orchestration.

The Orchestrator Studio provides "Insert Step Before" and "Insert Step After" buttons so you can add a step before or after another step in the orchestration. Therefore, you do not have to determine whether or not you need to add a particular step, such as a white list, to an orchestration before you add other steps.

Figure B–8 shows an orchestration that contains multiple rules, cross references, and service requests, which are displayed in the grid on the Orchestration design page:

Figure B–8 Steps in the AddConditionBased Alert Sample Orchestration in the Orchestrator Studio

Type	Action	Name		
Rule	True	JDE_RULE_Sample_VibrationAlarm Personal RUL_1608250018CUST	✓	edit
Cross Reference	True	JDE_XREF_Sample_SensorLocation Pending Approval XRE_1608250006CUST	✓	edit
Cross Reference	True	JDE_XREF_Sample_AlertNotificationRecipients Shared XRE_1608250008CUST	✓	edit
Service Request	True	JDE_SREQ_Sample_AddCBAlert_Alarm Personal SRE_1609270001CUST	✓	edit
Rule	False	JDE_RULE_Sample_VibrationWarning Personal RUL_1608250017CUST	✓	edit
Cross Reference	True	JDE_XREF_Sample_SensorLocation Pending Approval XRE_1608250006CUST	✓	edit
Cross Reference	True	JDE_XREF_Sample_AlertNotificationRecipients Shared XRE_1608250008CUST	✓	edit
Service Request	True	JDE_SREQ_Sample_AddCBAlert_Warning Personal SRE_1609020001CUST	✓	edit

In the grid, the Type column shows the type of step and the Name column shows the name of the component for the orchestration step. The "True" or "False" values in the Action column determine the actions the orchestration performs based on the conditions set in the rules. See [Defining the Actions Between a Rule and Dependent Components](#) for information about defining the actions in an orchestration.

To add steps to an orchestration:

1. To add the initial step to an orchestration, click the **Add Step** button (+ symbol).
2. In the "Enter Type of Step" pop-up field, select one of the following steps to add to the orchestration, and then click the **Ok** button.

- **Cross Reference**
- **Rule**
- **Service Request**
- **White List**

The Orchestrator Studio displays the first step in the grid.

3. Add a component to the step:
 - a. To use an existing component for the new step, click the drop-down menu in the Name column and select a component.
 - b. To create a new component for the step, click the **Edit** button (pencil icon) at the end of the row to access the design page for creating the new component.

After creating the component, when you return to the Orchestration design page, you will need to select the component from the drop-down menu in the orchestration steps grid to add it.

See the appropriate topics in this chapter on how to create each type of orchestration component.

4. In the Transformations area on the right side of the page, configure the transformations for each orchestration step as described in the [Mapping Inputs in the Transformations Area](#) section.
5. To add additional steps to an orchestration:
 - a. Select a step in the grid, and then click either the **Insert Step Before** or **Insert Step After** button to add an additional step before or after the selected step.
 - b. In the "Enter Type of Step" pop-up dialog box, select the step that you want to add.
 - c. Click the **Ok** button.

To remove a step from an orchestration:

Caution: Before you delete a step, make sure the step is not used by any other component in the orchestration.

1. Select the step that you want to remove.
2. Above the grid with the steps, click the **Remove Step** button (the X button).

If you make a mistake when updating an orchestration, click the **Restore All** button in the upper-right corner of the design page to restore the orchestration to its last saved version, which erases any changes made since the last save.

B.9.4.1 Defining the Actions Between a Rule and Dependent Components

The Orchestration design page contains an Action column for defining the actions between a rule step and other orchestration steps in an orchestration. After you create a rule with conditions and add it as a step to an orchestration, you need to define the action the orchestration takes if the condition in the rule is met. For example, if a rule contains a condition that when met should invoke a service request step, then in the row with the service request step, you set the Action column to **True**.

You can also define a **False** action to invoke a different orchestration step when a condition in the initial rule is NOT met. A **False** action can invoke a different Service Request step or another rule step that contains additional conditions for which the incoming data is evaluated against. Thus, you can design an orchestration with as many rules and service requests as your business requirements necessitate.

[Figure B-9](#) shows an example of an orchestration with two rule steps and two service request steps, with the following defined actions:

- ❑ For the first service request step, the action is set to **True** to instruct the orchestration to invoke this service request step when the condition in the first rule step is met.
- ❑ The action for the second (nested) rule step is set to **False** to instruct the orchestration to invoke this rule when the condition in the first rule is NOT met.
- ❑ The action for the second service request step is set to **True** to instruct the orchestration to invoke this service request when the condition in the second rule is met.

Figure B–9 Defining the Orchestration Actions

Type	Action	Name
CrossReference		JDE_XREF_Sample_SensorLocation
CrossReference		JDE_XREF_Sample_AlertNotificationRecipients
Rule	True	JDE_RULE_Sample_CBMAlarm_1
ServiceRequest	False	JDE_SREQ_Sample_AddCBArt_Alarm
Rule	True	JDE_RULE_Sample_CBMAlarm_2
ServiceRequest	True	JDE_SREQ_Sample_AddCBArt_Alarm

To set the Action column to True or False:

1. In the Orchestration design page, in the appropriate Rule or Service Request step row, click in the Action column.
2. Select either **True** or **False** as appropriate.
3. Click the **Save** button.

After completing the orchestration, you can use the Orchestrator Client to test the orchestration in a test environment. You can access the Orchestrator Client from the drop-down menu in the upper-right corner of the Orchestrator Studio. See [Chapter 9, "Testing Orchestrations in the EnterpriseOne Orchestrator Client"](#) for more information.

B.9.5 Mapping Inputs in the Transformations Area

Use the Transformations area on the Orchestration design page to map orchestration inputs to inputs defined in an orchestration step, such as inputs in a rule, cross reference, white list, or service request component. The Transformations area includes an Auto Map button that automatically maps any matching inputs in the orchestration and orchestration step.

If an orchestration and an orchestration component use different input names to identify the same data, you can use the grid in the Transformations area to map the inputs. This facilitates the reuse of components, so you can avoid having to create a new component simply to match the input names (input from a device) in a new orchestration.

To better understand how you use transformations, consider the following scenario which is depicted in [Figure B–10](#):

- ❑ You have an existing service request that creates alerts in the EnterpriseOne Condition-Based Alerts Revisions application (P1311). This service request includes an input called EquipmentNumber that is mapped to the Asset Number field in P1311.
- ❑ You want to reuse this service request as a step in a new orchestration, but the new orchestration will consume data from devices that supply the EquipmentNumber as "SerialNumber."
- ❑ To reuse the existing service request designed to consume data labeled as EquipmentNumber, in the new orchestration, you can create a transformation to pass the SerialNumber input to the EquipmentNumber input in the service request.

- Create the new orchestration with SerialNumber as one of the inputs, and then add the existing service request as a step.
- When you click the Service Request step in the orchestration steps grid, the Service Request column in the Transformations table displays all of the inputs defined in the service request. Notice in [Figure B-10](#) that EquipmentNumber, not SerialNumber, is defined as an input in the Service Request Input column.
- Next, you click the Transformations Auto Map button, which automatically maps the matching inputs. [Figure B-10](#) shows the SiteNumber, Latitude, and Longitude inputs in the Orchestration Input column, which were populated automatically after clicking Auto Map.
- Finally, to map the SerialNumber input in the orchestration to the EquipmentNumber input in the service request, you click the drop-down menu in the Orchestration Input column and select SerialNumber, as shown in [Figure B-10](#).

Figure B-10 Transformations Example

Orchestration Steps			Transformations	
Type	Action	Name	Orchestration Input	Service Request Input
Service Request		JDE_SREQ_Sample_UpdateEquipmentLocation	SerialNumber	EquipmentNumber
Input		Value Type		
SerialNumber		String	SiteNumber	CustomerNumber
SiteNumber		String		SiteNumber
Latitude		String		StartEffectiveDate
Longitude		String		StartEffectiveTime
				Remark
				Latitude
				Latitude
				Longitude
				Longitude

Initially, when you have a small number of orchestrations in your system, it is recommended to keep all input names consistent among the orchestration and all components (orchestration steps) used by orchestration. But as your library of orchestrations and orchestration components increases, instead of creating new components with input names that match the orchestration inputs, you can use transformations to map the orchestration inputs to an existing component.

To map inputs:

1. In the Orchestration design page, select an orchestration step to which you want to map orchestration inputs.
2. Click the Transformations **Auto Map** button to map any matching input names. Matching inputs automatically appear in the Orchestration Input column next to the matching input in the "<orchestration step> Input" column.
3. To map an orchestration input to an orchestration step input that does not have the same name:
 - a. In the Transformations table, click the drop-down menu in the Orchestration Input column next to the orchestration step input.
 - b. Select the orchestration input to map to the orchestration step input.

For example, in [Figure B-10](#), SerialNumber is mapped to EquipmentNumber in the Service Request Input.

4. You can map a literal value to the orchestration step input by entering a value in the Default Value column.

B.9.6 Updating Version 1 Orchestration to Version 2 Orchestration

When executed by the Orchestrator on the AIS Server, orchestrations programmatically call AIS services on the AIS Server to perform specific actions in EnterpriseOne.

Starting with EnterpriseOne Tools 9.2.1.2 is the availability of version 2 AIS services. Version 2 AIS services include all services originally available on the AIS Server (referred to as version 1 services), plus additional services that enable you to create an orchestration that includes the following types of service requests:

- ❑ Data request
- ❑ Message
- ❑ Connector

All orchestrations that you create with Orchestrator Studio 5.0.1 or higher use version 2 AIS services and are referred to as version 2 orchestrations.

All orchestrations created prior to Orchestrator Studio 5.0.1 use version 1 AIS services and are referred to as version 1 orchestrations.

In Orchestrator Studio 5.0.1 or higher, you can update version 1 orchestrations. However, if you add any of the components in the preceding list that rely on version 2 AIS services, you must save the orchestration as version 2 by selecting **Version 2** from the Orchestrator Version drop-down list.

Caution:

If you save a version 1 orchestration as version 2, the following parameters are used for the service request by default:

- ❑ Output Type = grid data
- ❑ Turbo Mode = low

This changes the format of the response, which will affect any device consuming the output returned from the orchestration.

B.9.7 Working with Orchestration Output

Use the Orchestration Output page to view the JSON representation of orchestration output. Viewing the output enables you to validate the fields that contain the data you want returned in the orchestration output for output mapping.

Orchestration output can include values from fields and grid columns that you specify as returns when setting up a form request, data request, or cross reference in an orchestration. It can also include the response from a connector, custom service request, or the result of a rule (true or false).

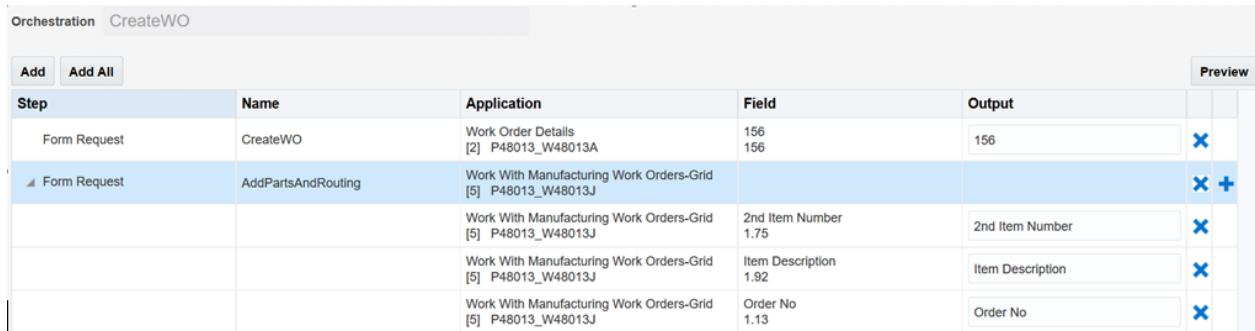
On the Orchestration Output page, you can use Groovy scripting to manipulate the contents of the response to refine the orchestration output as required by the parameters in the consuming device or program.

Note: Orchestration output is available only with version 2 orchestrations created in Orchestrator Studio 5.0.1 or higher. For orchestrations created prior to Orchestrator Studio 5.0.1, you must update them to version 2 orchestrations to view orchestration output.

To work with orchestration output:

1. On the Orchestration design page, click the **Orchestration Output** button.
2. If you want to view the JSON code of all return fields in an orchestration, click **Add All**.

The grid displays all of the components or steps in the orchestration that contain return fields, as shown in the following example:



The screenshot shows a grid interface titled 'Orchestration CreateWO'. At the top, there are buttons for 'Add' and 'Add All'. The 'Add' button is highlighted. The grid has columns: Step, Name, Application, Field, and Output. The 'Step' column shows 'Form Request' and 'AddPartsAndRouting'. The 'Name' column shows 'CreateWO' and 'AddPartsAndRouting'. The 'Application' column shows 'Work Order Details [2] P48013_W48013A' and 'Work With Manufacturing Work Orders-Grid [5] P48013_W48013J'. The 'Field' column lists various fields like '156', '2nd Item Number 1.75', 'Item Description 1.92', and 'Order No 1.13'. The 'Output' column shows variable names like '156', '2nd Item Number', 'Item Description', and 'Order No'. There are also 'X' and '+' icons in the 'Output' column.

3. Instead of Add All, you can individually select the return fields for which you want to preview the JSON code:
 - a. Click **Add** to view a list of all steps (orchestration components) in the orchestration that contain return fields.
You can expand each step to see the return fields. The Output Field column displays the variable name defined for a return field.
 - b. In the Select column, click the check box next to any return field that you want to add to your orchestration output, and then click **OK**.
The field will be returned in JSON format, which you can also preview on the design page.
4. To modify the return fields for which you want to see the JSON representation:
 - You can change the variable name for the return field in the Output column.
 - Click **Delete (X)** at the end of row to remove a return field from the JSON preview.
 - Click **Add (+)** to add any additional grid controls not currently selected for output.
5. Click the **Preview** button.

The following image shows an example of the JSON code for three grid return fields: 2nd Item Number, Item Description, and Order Number.

```
{
  "Grids": [
    {
      "Grid ID": "1",
      "Title": "Work With Manufacturing Work Orders",
      "Row Set": [
        {
          "2nd Item Number": "XXXXX",
          "Item Description": "XXXXX",
          "Order No": "XXXXX"
        }
      ]
    }
  ]
}
```

Add **Add All**

Step	Name	Application	Field	Output
Form Request	AddPatsAndRouting	Work With Manufacturing Work Orders [5] P48013_W48013J		
		Work With Manufacturing Work Orders [5] P48013_W48013J	2nd Item Number 1.75	2nd Item Number
		Work With Manufacturing Work Orders [5] P48013_W48013J	Item Description 1.92	Item Description
		Work With Manufacturing Work Orders [5] P48013_W48013J	Order No 1.13	Order No

Close **Copy to Clipboard**

- In the Preview area, you can click the **Copy to Clipboard** button so you can paste the JSON code into your program.

- (Advanced) In Orchestrator Studio 5.1.0, edit the provided Groovy script template in the Manipulate Output area to manipulate the contents of the response to refine the orchestration output.

See [Groovy Template for Manipulating Output from an Orchestration Response](#) for more information.

- Click the **Test** button to view the output.

The Orchestrator Studio displays a preview of the output manipulated by the Groovy script.

- To save the orchestration outputs, return to the Orchestration design page and click **Save**.

B.10 Reloading Orchestrations and Orchestration Components

Reload Files enables you to reload orchestrations and orchestration components to the state in which they were last saved in the Orchestrator Studio. Use this feature if you do not want to save any modifications that you made.

To reload all files, click the drop-down menu in the upper-right corner of the Orchestrator Studio and then click the **Reload Files** link.

Each individual orchestration component panel also contains a Restore button that you can use to restore an individual component.

B.11 Supported Input Message Formats

The Orchestrator supports three input message formats for orchestrations: a standard JD Edwards EnterpriseOne format, a generic format, and Oracle Cloud IoT format. [Example B–1](#) and [Example B–2](#) show an example of the code for each format.

Example B–1 Standard JD Edwards EnterpriseOne Input Message Format

```
{
  "inputs": [
    {
      "name": "equipmentNumber",
      "value": "41419"
    },
    {
      "name": "description",
      "value": "test"
    },
    {
      "name": "date",
      "value": "2018-01-01T00:00:00Z"
    }
  ]
}
```

```

        "value": "1427774400000"
    },
    {
        "name": "time",
        "value": "12:15:15"
    },
    {
        "name": "temperature",
        "value": "99"
    }
]
}

```

Example B–2 Generic or Oracle Cloud IoT Input Message Format

```
{
    "equipmentNumber": "41419",
    "description": "test",
    "date": "1427774400000",
    "time": "12:15:15",
    "temperature": "99"
}
```

B.11.1 Additional Supported Input Message Formats for the Generic Input Format

Additional formats are supported when using the generic input format, as long as the orchestration input values are defined using the full path to the elements used. You may have a deeper JSON structure like this.

```
{
    "equipmentNumber": "41419",
    "equipementDetail": {
        "type": "thermometer",
        "readingDetail": {
            "temperature": 200,
            "units": "F"
        }
    }
}
```

To reference the temperature within the orchestration, you can use the full path delimited by periods, for example:

`equipmentDetail.readingDetail.temperature`

B.11.2 Differences Between Input Message Formats

The following list describes the differences between the input message formats:

- ▣ The iterateOver attribute for the orchestrationStep element is supported only by the standard JD Edwards EnterpriseOne input message format.
- ▣ When using the "detail" form action type with the standard format, it will automatically iterate over all detailInputs and repeatingInputs in order to add multiple rows to a grid. If the generic format is used, only a single grid row can be added.

As shown in [Example B–3](#), "detailInputs" would correspond to grid data; "repeatingInputs" would correspond to individual rows that contain "inputs" that correspond to columns.

If you have a power form with two grids, you could populate both grids using two "detailInputs" structures with different "name" values that correspond to the different grids. "repeatingInputs" would contain a list of rows and for each row you would define a list of "inputs".

You could also define a CrossRefReference orchestration step with `iterateOver="GridData"` that converts the item value into an EnterpriseOne specific item number. For example, if A123=220 and A124=230, the single orchestration step would convert both.

Example B-3

```
{  
    "inputs": [  
        {  
            "name": "BranchPlant",  
            "value": "30"  
        },  
        {  
            "name": "customer",  
            "value": "4242"  
        }  
    ],  
    "detailInputs": [  
        {  
            "name": "GridData",  
            "repeatingInputs": [  
                {  
                    "inputs": [  
                        {  
                            "name": "item",  
                            "value": "A123"  
                        },  
                        {  
                            "name": "Quantity",  
                            "value": "3"  
                        }  
                    ]  
                },  
                {  
                    "inputs": [  
                        {  
                            "name": "item",  
                            "value": "A124"  
                        }  
                    ]  
                }  
            ]  
        }  
    ]  
}
```

B.12 Setting Up Orchestration Security

Before the EnterpriseOne Orchestrator can process an orchestration, authentication of the JD Edwards EnterpriseOne user ID and password must take place. It is the responsibility of the originator of the service request to tell the orchestration about the user. The user's credentials must be supplied in a basic authorization header or in the JSON body of the request. The user must also have authorized access to the EnterpriseOne application in which the resulting

transaction takes place. The following code is an example of credentials in the JSON body of the request:

```
{
    "username": "JDE",
    "password": "JDE",
    "environment": "JDV900",
    "role": "*ALL"
}
```

The AIS service used with orchestrations is stateless; each call passes credentials to establish a separate EnterpriseOne session. After the transaction is complete, the session closes.

In addition to passing credentials for authentication, you can employ a second level of security for the Orchestrator through whitelisting. Whitelisting enables an initial rudimentary pass/fail check of the incoming device signature against a predefined list of signatures. A white list provides an additional layer of security to the Orchestrator security. If a value passed to the Orchestrator is not a valid value included in the orchestration's white list, the Orchestrator rejects the input. For more information, see [Creating a White List](#) in this guide.

B.12.1 Restricting Access to Exposed Orchestrations

If you expose orchestrations for business partners or customers to invoke, it is recommended to use an http proxy to allow access to only the endpoints required to execute the orchestration. Configure the proxy to restrict all endpoints except /orchestrator, /discover, /tokenrequest, and /tokenrequest/logout. This allows external users set up with the proper UDO security to discover and call orchestrations.

B.12.2 How to Maintain a Single Session for Multiple Calls to an Orchestration

You can maintain a single AIS Server session for multiple calls to an orchestration by first performing a token request to get a session token. Otherwise, each call to an orchestration on the AIS Server establishes a separate session, which can decrease AIS Server performance.

When performing a token request, a session is established and the AIS Server returns an AIS token in the response. The token is then used for all orchestration calls in the same session.

The amount of time the session remains open is established through the session lifetime settings in the AIS server. The third party client is responsible for ensuring a valid token is available or re-requesting a new token once a previous token has timed out. If a valid token is not available, the client will receive an error in the response stating the token is invalid.

B.13 Exporting Orchestration Components from the Orchestrator Studio

Caution: Do not use the Export and Import tools to share orchestration component files with other users. With orchestration components stored as UDOs in EnterpriseOne, the management of orchestration components, including the sharing and modifying of shared orchestration components and promoting of orchestration components to the appropriate AIS Server directory for test or production purposes, is administered through the life cycle management tools in EnterpriseOne.

You can export any of the orchestration component types from the Orchestrator Studio to view the source XML files of the components. This is only recommended for advanced users for troubleshooting purposes or for making manual customizations. The Orchestrator Studio exports the source XML files in a zip file.

To export an *orchestration* component, the Orchestrator Studio gives you the option to export only the selected orchestration component or to export the orchestration component and all components associated with it. For example, if an orchestration has a service request component and a rule component associated with it, if you export all components, the zip file will contain XML files for the orchestration, service request, and rule.

To export orchestration components:

1. On a component design page, select the component to export and then click the **Export File** button in the upper-right corner.

If you are exporting an orchestration, a dialog box appears in which you can click **All** or **Orchestration Only**.

2. Follow the instructions in the browser to save the zip file to a location on your local machine.

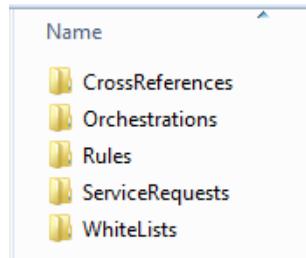
B.14 Importing Orchestration Files in the Orchestrator Studio

Caution: Do not use the Export and Import tools to share orchestration component files with other users. With orchestration components stored as UDOs in EnterpriseOne, the sharing and modifying of shared orchestration components, as well as the promoting of orchestration components to the appropriate AIS Server directory for test or production purposes, is administered through the life cycle management tools in EnterpriseOne.

You might need to import orchestration component XML files that were exported from the Orchestrator Studio for advanced troubleshooting or customization purposes. The Orchestrator Studio Import tool enables you to import individual orchestration XML files or a zip file containing XML files.

Caution: You cannot import orchestration component files that were exported from the EnterpriseOne Object Management Workbench - Web application. The zip file created from an OMW - Web export stores the XML files in a structure that cannot be read by the Orchestrator Studio Import tool.

If importing a zip file that contains orchestration XML files and dependent orchestration component XML files, the zip should contain the following folders with each of the XML files stored in the appropriate folder:



If you import an XML file that has the same name as a current orchestration component (UDO) in the Orchestrator Studio, you have the following options:

- If the UDO is at a status of "Personal" or "Reserved," you can overwrite the UDO with the XML file.
- If the UDO is in a "Shared" status, you cannot overwrite it. But you are given the option to import it as a new UDO with a status of "Personal."

To import files:

1. On the Orchestrator Studio Home page, click the **Tools** link in the upper-right corner.
2. On the Tools page, click the **Import Files** icon.
3. On Import Files, click the **Choose File** button.
4. Locate the orchestration component XML files or zip file that contains the orchestration component files that you want to import.

After selecting the files to import, the Import tool checks the XML files that you selected against the current UDOs in the Orchestrator Studio and displays the options that are available for importing the files, as shown in the example in [Figure B–11](#).

Figure B–11 Orchestrator Studio Import Tool

The screenshot shows the 'Import Tool' interface. At the top, there's a file selection area with a dropdown for 'Select File for Import' containing 'Orchestrations-1461866371912.zip', an 'Update...' button, and a 'Submit' button. Below this, it shows 'Number of Files 9' and 'Number of Files Existing on Server 2'. A 'Select All' checkbox is available. The main area is divided into sections: 'Cross Reference', 'Service Requests', 'Rules', and 'Orchestrations'. Each section lists XML files with their corresponding import options. In the 'Service Requests' section, the first file 'JDE_SREQ_Sample_AddCBArt_Warning.xml' has its 'Overwrite existing file' checkbox checked. The second file 'JDE_SREQ_Sample_AddCBArt_Alarm.xml' has its 'File is ready to submit' checkbox checked. In the 'Rules' section, all four files have their 'File is ready to submit' checkboxes checked. In the 'Orchestrations' section, the single listed file has its 'File is ready to submit' checkbox checked. A cursor is visible over the 'File is ready to submit' checkbox for the second 'Service Requests' file.

Section	File	Action
Cross Reference	JDE_XREF_Sample_SensorLocation.xml	No update allowed at current status <input type="checkbox"/>
	JDE_XREF_Sample_AlertNotificationRecipients.xml	Select to add or else reserve record and re-import to update <input type="checkbox"/>
Service Requests	JDE_SREQ_Sample_AddCBArt_Warning.xml	Overwrite existing file <input checked="" type="checkbox"/>
	JDE_SREQ_Sample_AddCBArt_Alarm.xml	File is ready to submit <input checked="" type="checkbox"/>
Rules	JDE_RULE_Sample_CBMAAlarm_3.xml	File is ready to submit <input checked="" type="checkbox"/>
	JDE_RULE_Sample_CBMAAlarm_1.xml	File is ready to submit <input checked="" type="checkbox"/>
	JDE_RULE_Sample_CBMAAlarm_2.xml	File is ready to submit <input checked="" type="checkbox"/>
	JDE_RULE_Sample_CBMWarning.xml	File is ready to submit <input checked="" type="checkbox"/>
Orchestrations	JDE_ORCH_Sample_AddConditionBasedAlert_Generic.xml	File is ready to submit <input checked="" type="checkbox"/>

5. Review the import option for each file to determine how to proceed with the import. The options are:

- No update allowed at current status.

The XML file name is the same as an existing component, which has a status of "Pending Approval," so it cannot be overwritten.

- Select to add or else reserve record and re-import to update.

A component with the same name exists in the Orchestrator Studio in a "Shared" status. Click the check box if you want to import the file as a new UDO. A new name will be generated for the new component upon import.

If you want to overwrite the current component in the Orchestrator Studio, clear the check box. And then change the status of the current component to "Reserved" and reimport the XML file to overwrite the component.

- File already exists.

The XML file matches a component that is in a "Personal" or "Reserved" status. Click the check box to import it and overwrite the current component.

- File is ready to submit.

The XML file is unique and does not already exist in the Orchestrator Studio. Click the check box to import it.

6. You can also click the **Select All** check box to import all XML files that are available for importing.

7. Click the **Submit** button.

The Orchestrator Studio imports the selected components into the components list on the respective design page.

C

Creating Orchestrations with Orchestrator Studio 3.0.1 (Release 9.2.1)

Note:

This appendix describes how to use Orchestrator Studio 3.0.1 that was available starting with EnterpriseOne Tools 9.2.1.

To use Orchestrator Studio 7.0.x.0, the latest version available with EnterpriseOne Tools 9.2.3, see:

- [Chapter 2, "Implementing the Orchestrator Studio"](#) for installation instructions
- [Chapter 4, "Creating Orchestrations with Orchestrator Studio 7.x.x.x"](#)

This appendix contains the following topics:

- [Section C.1, "Understanding the Orchestrator Studio and Orchestrations"](#)
- [Section C.2, "Accessing the Orchestrator Studio"](#)
- [Section C.3, "Navigating the Orchestrator Studio"](#)
- [Section C.4, "Working with the Orchestrator Studio Design Pages"](#)
- [Section C.5, "Creating Service Requests"](#)
- [Section C.6, "Creating Rules"](#)
- [Section C.7, "Creating Cross References"](#)
- [Section C.8, "Creating White Lists"](#)
- [Section C.9, "Creating Orchestrations"](#)
- [Section C.10, "Reloading Orchestrations and Orchestration Components"](#)
- [Section C.11, "Supported Input Message Formats"](#)
- [Section C.12, "Setting Up Orchestration Security"](#)
- [Section C.13, "Exporting Orchestration Components from the Orchestrator Studio"](#)
- [Section C.14, "Importing Orchestration Files in the Orchestrator Studio"](#)

C.1 Understanding the Orchestrator Studio and Orchestration

The Orchestrator Studio is a web-based application for creating orchestrations and the components that comprise an orchestration. The Orchestrator uses these components to process a single orchestration instance on the AIS Server.

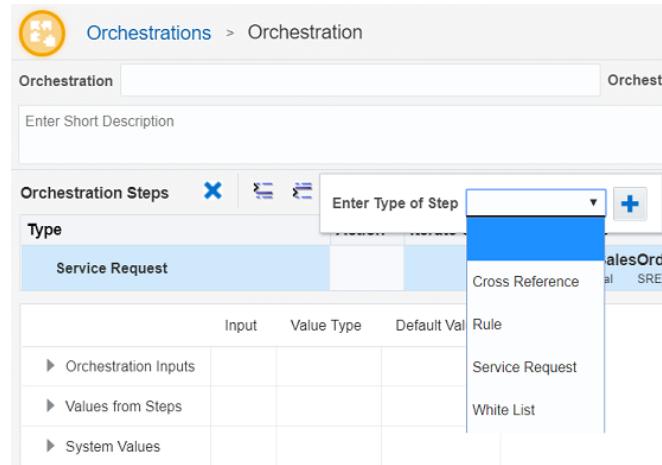
Use the Orchestrator Studio to create the following components:

- ❑ **Orchestrations.** An orchestration is the master component that provides a unique name for an orchestration process. The orchestration is where you define the inputs for the orchestration, the expected incoming data from a device. The orchestration also includes orchestration steps, which are invocations to the other components described in this list. When the Orchestrator invokes an orchestration, it processes the steps defined in the orchestration to enable the transfer of data from third-party systems to EnterpriseOne.
- ❑ **Service Requests.** A service request contains a sequence of actions to perform a particular process in EnterpriseOne. The Orchestrator uses the service request as an invocation of a JD Edwards EnterpriseOne interactive application or a Java application via a REST service call to the AIS Server.
- ❑ **Rules.** A set of conditions that the input to the orchestration is evaluated against to produce a true or false state. With rules, a false outcome or true outcome can invoke further orchestration steps. You can also nest rules, in which an outcome of one rule can invoke a different rule, to produce complex evaluations. You can also use custom Java to define rules.
- ❑ **Cross References.** A set of data relationships that map third-party values to EnterpriseOne values. For example, a device's serial number can be cross-referenced to a JD Edwards EnterpriseOne Equipment Number for use in service requests.
- ❑ **White Lists.** A white list contains an inclusive list of values permitted in the orchestration and terminates the orchestration process if the data is not recognized.

A simple orchestration requires at a minimum an orchestration and a service request to run.

Figure C-1 shows the drop-down list of steps (or components) that you can add to an orchestration. Each step in an orchestration is simply a reference to a cross reference, rule, service request, or white list component.

Figure C-1 Orchestrations Steps in the Orchestrator Studio



C.1.1 Reusable Orchestration Components

Orchestration components are reusable. A single component, such as a rule or cross reference, can be included as a step in more than one orchestration. If a component is used as a step in more than one orchestration, you should evaluate how it is used in other orchestrations before modifying it. To determine if a component is used by other orchestrations, select the component and then click the "i" button to display a pop-up window with a list of orchestrations where the component is used.

When in doubt, use the "Save As" button to create a new component from an existing one. This enables you to give it a new name and modify it as necessary, and eliminates the risk of breaking other orchestrations where the component is used.

C.1.2 Orchestrations as User Defined Objects

All orchestration components that you create in the Orchestrator Studio are stored as user defined objects (UDOs) in EnterpriseOne. The Orchestrator Studio includes UDO features for creating, sharing, and modifying orchestration components as UDOs. See [User Defined Object \(UDO\) Features](#) for a description of the UDO features.

C.2 Accessing the Orchestrator Studio

The Orchestrator Studio is a web application that runs in a web browser. Ask your system administrator for the URL to the Orchestrator Studio.

Important: Before users can access the Orchestrator Studio, an administrator must set up security to authorize access to the Orchestrator Studio design pages and determine the actions Orchestrator Studio users can perform. See [Chapter 10, "Administering the Orchestrator Studio and Orchestrations"](#) for more information.

To access the Orchestrator Studio:

1. In a web browser, enter the URL to the Orchestrator Studio:
`http://<adf_server>:<port>/OrchestratorStudio/faces/index.jsf`
2. On the Orchestrator Studio Sign In screen, enter your EnterpriseOne User credentials, environment, and role.

Note: It is highly recommended that you enter an EnterpriseOne environment used for testing, not a production environment.

3. Click the **Login** button.

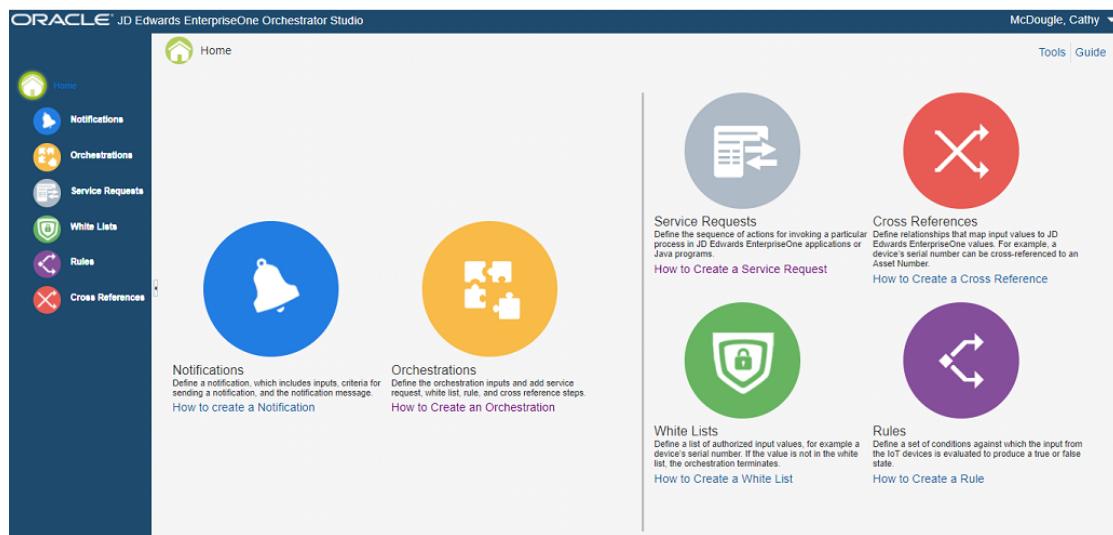
In the Orchestrator Studio, click the drop-down menu in the upper-right corner to view the path to the AIS Server. The drop-down menu also provides a link to log out of the Orchestrator Studio.

C.3 Navigating the Orchestrator Studio

The first page that appears after you log in to the Orchestrator Studio is the Home page. The orchestration component icons on the Home page take you to the design pages for creating and modifying each orchestration component. You can click the Home icon at the top left of the Home page to display a side panel, which provides another way to access the orchestration component design pages. You can also access this side panel within the component design

pages for easy navigation between the different design pages. Figure C–2 shows the side panel enabled on the Home page.

Figure C–2 Orchestrator Studio Home



The Tools link in the upper-right corner of the Home page provides access to the Orchestrator Studio Tools page. This page provides links to the Orchestrator Client for testing orchestrations, the Import tool for importing orchestration files, and the JD Edwards EnterpriseOne web client. For more information, see the following topics:

- ❑ Testing Orchestrations in the EnterpriseOne Orchestrator Client
- ❑ Importing Orchestration Files in the Orchestrator Studio

C.4 Working with the Orchestrator Studio Design Pages

This section describes:

- ❑ Design Page Features
- ❑ User Defined Object (UDO) Features
- ❑ Working with the Graphical Representation of an Orchestration

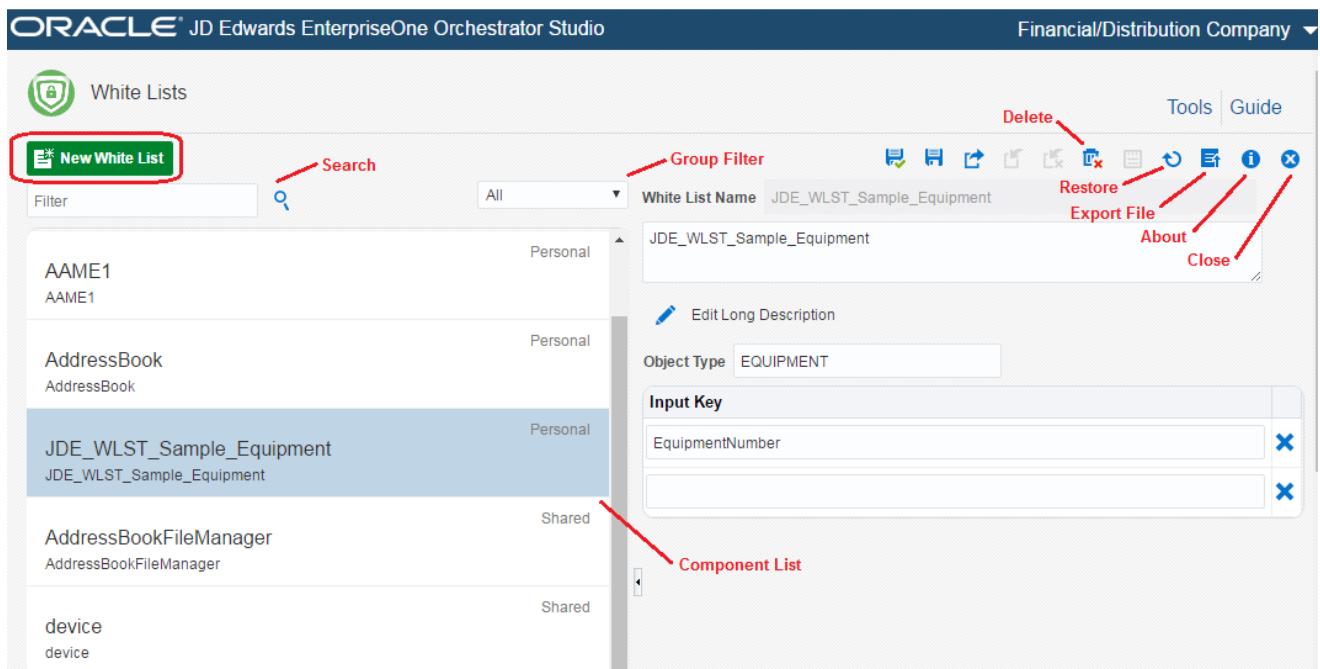
C.4.1 Design Page Features

The Orchestrator Studio design pages contain the following features, which are highlighted in Figure C–3:

- ❑ **Component list.** Displays a list of existing components.
You can use the vertical divider next to the component list to adjust the size of the list. Or you can click the raised tab on the divider to hide or show the component list.
- ❑ **Group Filter drop-down list.** Enables you to display components in the component list by UDO status: Personal, Pending Approval, Rework, Reserved, or All.
- ❑ **Search field.** Search for an existing component in the list.
- ❑ **New <Component> button.** Create a new component.

- **i (About)**. Takes you to the About page which provides the Status, Detail, Description, and Object Name of the selected component. It also shows a list of the orchestrations where the component is used.
- **Restore All or Restore <Component>**. Restore a component to its original state if you made a mistake and do not want to save your changes.
- **Export File**. Export a component file to your local machine, which you should use only to inspect the XML of the component. See [Exporting Orchestration Components from the Orchestrator Studio](#) in this guide for more information.
- **Close**. Exit the design page.

Figure C–3 Orchestrator Studio Design Page Features



C.4.2 User Defined Object (UDO) Features

Orchestration components are saved and managed as UDOs in EnterpriseOne, which enables administrators to use EnterpriseOne administration tools to manage the life cycle of orchestration components. See [Chapter 10, "Administering the Orchestrator Studio and Orchestrations"](#) in this guide for more information.

The Orchestrator Studio includes UDO features that enable you to create orchestration components for your own personal use, publish or "share" orchestration component UDOs with other Orchestrator Studio users, and modify shared UDOs created by others. The actions you can perform, or the UDO features you can use, depends on UDO permissions granted to you by a system administrator. See [Setting Up UDO Security for Orchestrator Studio Users \(Release 9.2.1\)](#) for more information.

See Also:

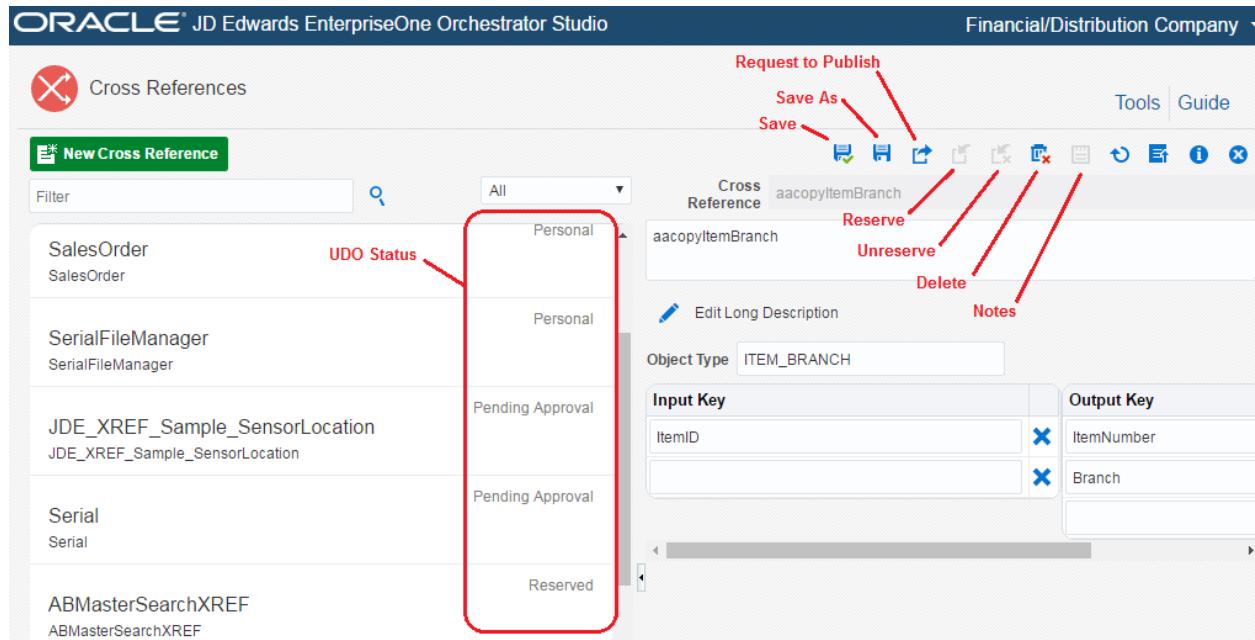
"Understanding UDO Life Cycle and Statuses" in the *JD Edwards EnterpriseOne Tools Using and Approving User Defined Objects Guide*.

Table C–1 describes the UDO buttons in the Orchestrator Studio design pages and the life cycle status enacted by each UDO action.

Table C–1 UDO Buttons in Orchestrator Studio Design Pages

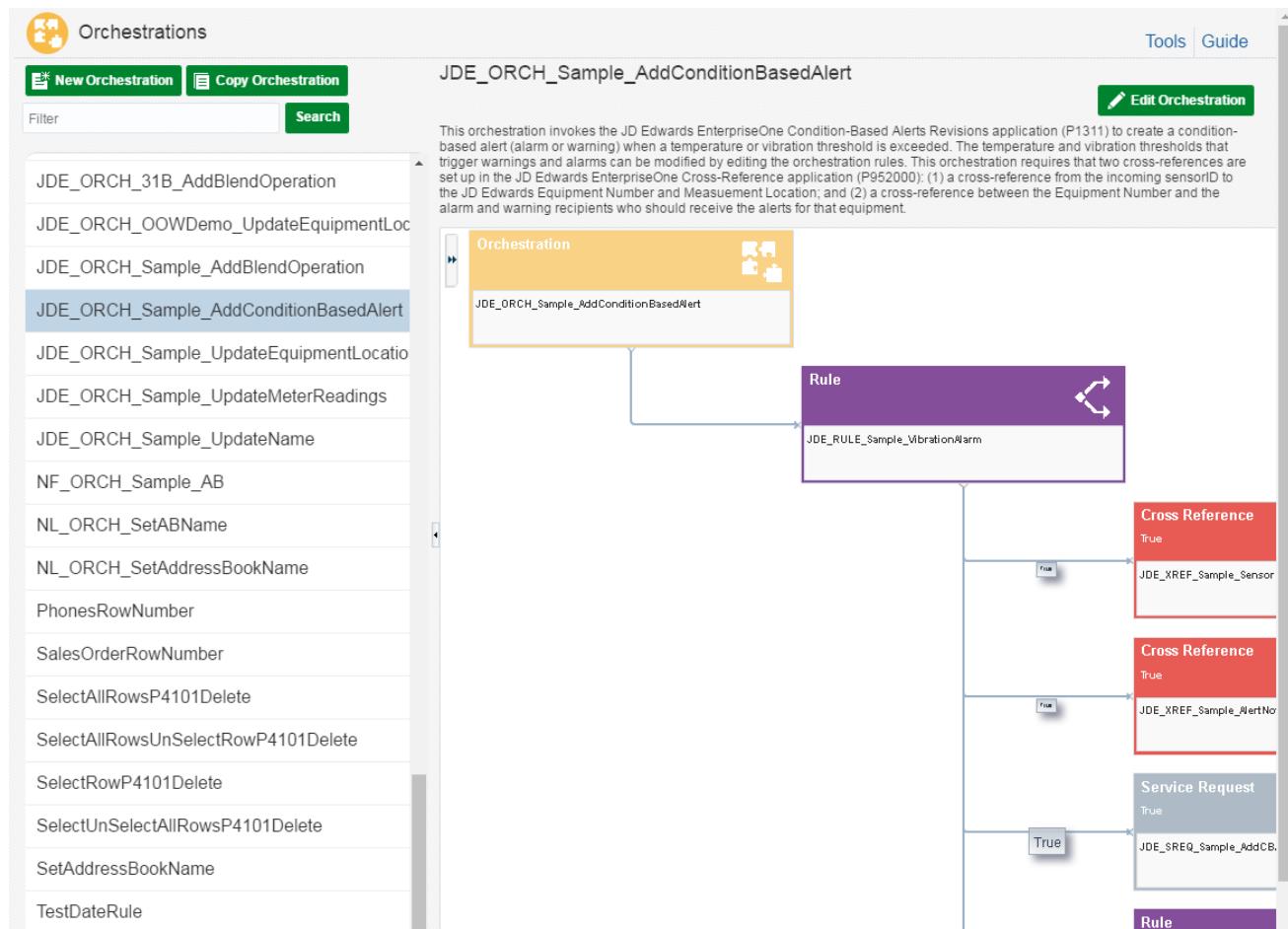
UDO Button	Description
Save and Save As	Saves the orchestration component to a status of "Personal." This means that the component has not been shared. Components with a status of "Personal" are components that you are developing and have not been shared for publishing to the AIS Server.
Request to Publish	Sends the orchestration component for approval for sharing. An administrator or approver must approve it in order for the UDO to be shared. The component status changes to "Pending Approval" in the component list and then changes to "Shared" once it is approved. If rejected, that status changes to "Rework." At that point, you can edit the component and then use the Request to Publish button to send it for approval again.
Reserve	Reserves a shared UDO so you can modify it. When reserved, no other users can make changes to it. The component status changes to "Reserved."
Unreserve	Cancels the reserved component, which returns the status of the component to "Shared."
Delete	Deletes a "Personal" UDO. You cannot use this button to delete a shared UDO. Shared UDOs can only be deleted by an administrator.
Notes	Available when the component is in the "Pending Approval" status, this button enables you to add an additional note to send to the approver of the UDO. The Notes button is active only if there was a note added the first time the UDO was sent for approval using the "Request to Publish" button. This feature enables you to add an addendum to the original note.

Figure C–4 highlights the UDO features.

Figure C–4 Design Page UDO Features

C.4.3 Working with the Graphical Representation of an Orchestration

The initial Orchestrations design page shows a graphical representation of a selected orchestration with all its components, as shown in the example in Figure C–5.

Figure C–5 Initial Orchestrations Design Page

Use the following features in the graphic area to work with a graphical representation of the orchestration:

- **Navigation toolbox**

In the upper-left corner of the graphic area, click the raised tab to display a toolbox for navigating the graphic. Use the directional controls in the toolbox to pan left, right, up, and down, as well as zoom in or zoom out. Or you can click the "Zoom to Fit" button to display the entire graphical representation in the window. Use the layout buttons to change the layout to vertical, horizontal, tree, radial, or circle. Changing the layout can help with viewing more complex orchestrations that contain multiple components.

- **Informational hover help**

You can hover your mouse over a component in the graphical area to view an enlarged image of the component. Hovering over the labels on the lines between a rule component and its child components magnify the "True" or "False" label. A "True" label indicates that the child component will be invoked if the conditions in the rule are met. A "False" label indicates that the child component will be invoked when the condition of the rule is not met.

- **Isolate and Restore buttons**

The left side of each component contains an Isolate button that you can click to show only that component in the graphic area. You can then click the Restore button to display the entire graphical representation of the orchestration.

- ❑ Access to the design page for editing the component

When you click a box representing a component, the Orchestrator Studio takes you to the design page for modifying that particular component.

C.5 Creating Service Requests

This section contains the following topics:

- ❑ Understanding Service Requests
- ❑ Understanding the Service Request Design Page
- ❑ Creating a Service Request
- ❑ Configuring the Actions in the Service Request
- ❑ Defining the Order of Execution in the Service Request
- ❑ Creating a Custom Java Service Request

C.5.1 Understanding Service Requests

At a minimum, an orchestration requires an orchestration component with inputs and a service request component to run. The service request provides the instructions that the Orchestrator uses to invoke and perform a particular task in EnterpriseOne. The service request provides the instructions on how to process the data from the orchestration inputs.

Before you create a service request, you must first identify the EnterpriseOne task or business process that you want the service request to perform. See the [Identifying the Service Request Information for the Orchestration](#) section in this guide for more information.

Use the Service Request design page in the Orchestrator Studio to define a service request, which involves:

- ❑ Specifying the EnterpriseOne buttons, fields, columns, menu items, and so forth that are involved in performing a task in EnterpriseOne.
In the Service Request design page, you can search on an EnterpriseOne application form and view all of the controls and fields for a form.
- ❑ Determining the order of execution in which the controls and fields are used to perform the desired task in EnterpriseOne.

Note: You can also use custom Java to execute a custom process or to route data into another database. See [Chapter 7, "Creating Custom Java for Orchestrations"](#).

C.5.1.1 Understanding the Application Stack Option

The Application Stack option in the Service Request design page enables you to create a service request that establishes a session for a specific application and maintains that session across calls. With application stack processing, the service request can invoke multiple forms in different applications to complete a business process. Without the application stack processing, each form in the service request is opened independently.

If you use the Application Stack option, the service request must include instructions for navigating between forms. Without the navigation, additional forms will not be called.

Only values from the last form in the application stack can be returned; return controls specified for any form except the last form are ignored.

C.5.2 Understanding the Service Request Design Page

The Service Request design page contains two areas: the "Order of Execution" area and the "Available Actions" area. The Available Actions area serves as the workspace for finding and configuring all of the EnterpriseOne controls and fields that make up the sequence of actions in the Order of Execution area. You locate and configure the controls and fields in the Available Actions area, and then move each item to the Order of Execution and place them in the proper sequence.

Available Actions Area

In the Available Actions area, you can search on an EnterpriseOne form to load all of the available controls (menu items, buttons, check boxes, and so forth) and fields for that form. If the service request needs to invoke more than one application to complete the task, you can load the controls and fields of additional application forms as needed. You can select the Application Stack check box to enable application stack processing as described in the [Understanding the Application Stack Option](#) section.

Figure C–6 shows the results of loading an EnterpriseOne application form in the Available Actions grid.

Figure C–6 Available Actions Area in the Service Request Design Page

Available Actions		Application Stack <input checked="" type="checkbox"/>	Run Synchronously <input checked="" type="checkbox"/>	Bypass Form Processing <input type="checkbox"/>	Form M
Application	Form	Version	Version	Form Mode	
P1311	W1311B - Condition-Based Alerts Revisions				
Description	Mapped Value	Default Value	ID	Version	Form Mode
Condition-Based Alerts Revisions			P1311_W1311B		
Buttons and Exits					
Cancel			12		
Create W.O.			114		
Failure Analysis			124		
Investigation			120		
Investigation Msg			115		
Notification			119		

The columns in the Available Actions grid provide interactive features for configuring the controls and fields before you add them to the Order of Execution area. The following list describes how to use each column:

■ Description

This column displays the controls and fields for each form in a collapsible/expandable parent node named after the EnterpriseOne form. Child nodes categorize other items, and include a Buttons and Exits node, a node for displaying the available Query by Example (QBE) fields in a grid (if applicable), and a node for displaying all of the available columns in a grid.

■ Mapped Value

Use this column to define the input variable name. Later, when you add the service request to an orchestration, you map the orchestration input to this name. You can only map inputs to a field if 1) the EnterpriseOne field is an editable field and 2) you defined the inputs in the orchestration as described in [Adding Inputs to an Orchestration](#).

- **Default Value**

Enter a default, hard-coded value if there is no mapped value. You can also add a hard-coded value to use if the mapped value returns a null.

- **"Text substitution" check box column**

This check box enables you to add text substitution for an input. Text substitution enables you to combine input values into a text string for input into an EnterpriseOne field. For example, you might have a string with "Temp was {0} at {1}" that contains a temperature input value and a time input value.

- **ID**

This column displays the ID of the control or field in EnterpriseOne. This is for informational purposes only.

- **Form Mode**

If a form mode was selected, this column indicates if a field or control is only available in the selected form mode.

- **Return**

You can specify any values that you want returned in the orchestration response after EnterpriseOne actions are invoked. If you do not specify any return values, all values on the form will be returned. Return values may be used by customized third-party gateway programs or devices to do further processing based on the values returned from EnterpriseOne applications.

- **Return Name**

This field is automatically enabled when the Return check box is selected. When enabled, you can manually enter a name in this column, which makes this value available for mapping in subsequent orchestration steps when the service request is added to an orchestration.

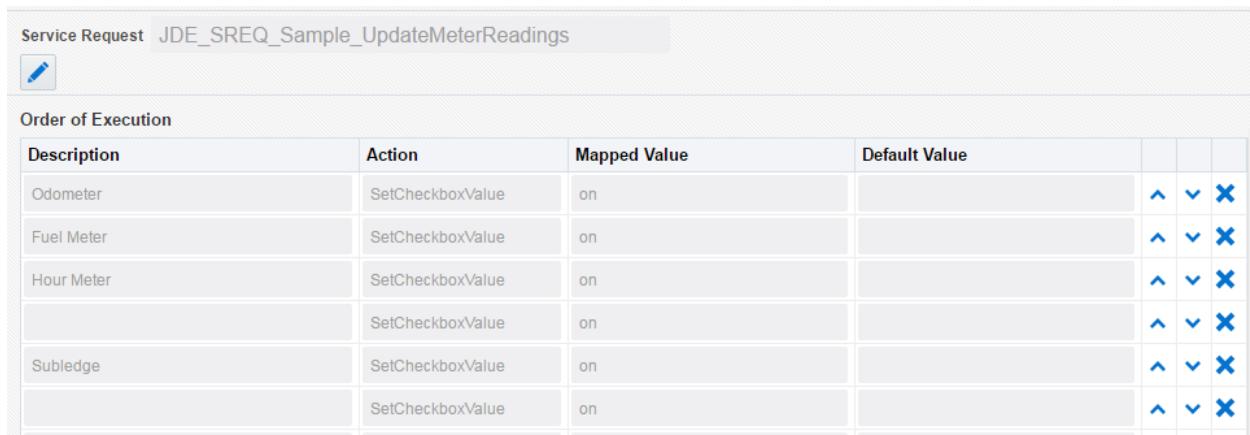
- **Select First Row**

Move this action to the Order of Execution if the service request tasks involves selecting the first row in the grid.

After you configure each applicable control or field in the Available Actions area, you click the **Add Action** button in the last column to add each item as an action in the Order of Execution area at the top of the design page.

Order of Execution Area

When actions are added to the Order of Execution grid, you can reorder them using the up and down arrow buttons to the right of each row. You can also delete any actions using the Delete (X) button. [Figure C-7](#) shows the Order of Execution area.

Figure C–7 Order of Execution in the Service Request Design Page


The screenshot shows the Service Request design page for a service named "JDE_SREQ_Sample_UpdateMeterReadings". At the top, there is a "Service Request" button and a pencil icon. Below that is a section titled "Order of Execution" containing a table.

Order of Execution

Description	Action	Mapped Value	Default Value	Up	Down	Remove
Odometer	SetCheckboxValue	on		^	▼	X
Fuel Meter	SetCheckboxValue	on		^	▼	X
Hour Meter	SetCheckboxValue	on		^	▼	X
	SetCheckboxValue	on		^	▼	X
Subledge	SetCheckboxValue	on		^	▼	X
	SetCheckboxValue	on		^	▼	X

C.5.3 Creating a Service Request

The first time a new service request is saved, it is saved as a "Personal" UDO. Thereafter, you can use the UDO buttons described in the [User Defined Object \(UDO\) Features](#) section to move the service request to the appropriate status.

To create a service request:

1. Access the Service Request design page:
 - On the Orchestrator Home page, click the **Service Requests** icon.
 - OR
 - After adding a Service Request step to an orchestration, click the **Service Request** step row.

The Orchestrator Studio displays the initial Service Requests design page.

2. On this page, click the **New Service Request** button.
3. On the Service Request design page, enter a name for the service request in the Service Request field.
Do **NOT** include any spaces in the service request name.
4. In the space provided, enter a short description with a maximum of 200 characters. This description appears below the service request name in the component list.
5. Click the **Edit Long Description** button to add a long description to provide more detail about the purpose of the component.
6. At this point, click the **Save** button, which saves the service request as a "Personal" UDO.
7. Proceed to configure the actions in the service request as described in the [Configuring the Actions in the Service Request](#) section.

C.5.4 Configuring the Actions in the Service Request

In the Service Request design page, use the Available Actions area to define actions that you want the service request to perform. After you define the actions, you can place the actions in the proper order in the Order of Execution area as described in [Defining the Order of Execution in the Service Request](#).

To configure the actions for the service request:

1. In the Available Actions area, complete the following fields to specify the EnterpriseOne form that contains the controls you want to load:
 - ❑ **Application.** Enter the ID of the EnterpriseOne application.
 - ❑ **Form.** Enter the form ID.
 - ❑ **Version.** Enter the version of the EnterpriseOne application.
 - ❑ **Form Mode.** (Available with Release 9.2.0.3) Click the drop-down menu and select the appropriate form mode: **Add**, **Update**, or **Inquiry**.

The form mode determines the controls that are displayed for the form. At runtime, the controls that appear on an EnterpriseOne form are dependent on the form mode as specified in Form Design Aid (FDA). This ensures that you can see all of the form controls in the Orchestrator Studio that are available in the form at runtime.

 - ❑ **Application Stack.** Click this check box if you want to use application stack processing. See [Understanding the Application Stack Option](#) for more information.
2. Click the **Load Form** button.

The Orchestrator Studio loads the controls and fields for the form in the grid.

If the action involves more than one form, you can search on another form and the Orchestrator Studio will load the controls for the form in a collapsible node at the end of the grid.
3. To map service request inputs to EnterpriseOne fields:
 - a. Select the row with the field in which you want to be mapped.
 - b. In the Mapped Value column, enter a variable name to use for the mapping.

When a service request is added to an orchestration, the inputs in the orchestration should match the inputs defined in the service request. You can also use the "Transformations" feature to map non-matching input names. See [Adding Inputs to an Orchestration](#) and [Configuring Transformations](#) for more information.
4. If the service request task involves selecting the first grid row in the EnterpriseOne form, in the "Select First Row" row, click the **Add Action** button to move it to the "Order of Execution" area.
5. If the service request task requires updating a particular row in an input capable grid, then map the appropriate input value containing the row number to the "Row Number for Update" row, or specify the row number in the Default Value column.
6. In the Default Value column, you can perform the following actions to configure the default values for the controls:
 - a. If the control is a check box or a radio button, you can select it to include it. If there are multiple radio buttons on a form, select only one. If you select more than one, only the last radio button in the Order of Execution list will be used.
 - b. If the control is a combo box (a list of items in a drop-down menu), then the Studio displays a drop-down menu in the Default Value column in which you can select the appropriate item from the list.
 - c. If the field is editable, you can enter a hard-coded value to map for the input. You can also enter a value to be used if the mapped value returns a null.
- Fields that are grayed out in this column are not editable.
7. Select the **Return** check box next to the fields that contain values that you want returned in the orchestration response. If you do not specify any return values, all values on the form will be returned.

Important: If using the Application Stack option:

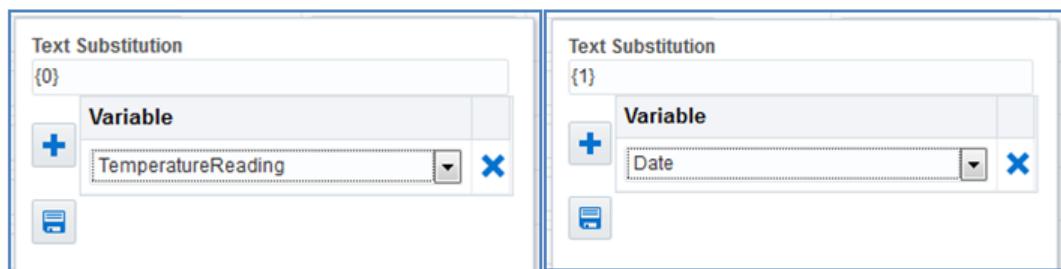
- Only values from the last form in the application stack can be returned; return controls specified for any form except the last form are ignored.
- If you use a form to add or update a value that upon saving exits to another form (such as a Find/Browse form), the return values will come from the last form that appears after the save. In this scenario, you need to add the last form to the application stack and select the return controls from the last form.

8. To use text substitution to combine input values into a text string for input into an EnterpriseOne field, perform these steps:

Note: If you use text substitution for the input, the field cannot contain a mapped value. The substituted text string becomes the default value for the field when you add it to the Order of Execution list.

- a. In the field that you want to substitute the input with text, click the **Add Text Substitution** button (plus symbol).
- b. In the pop-up dialog box, in the Text Substitution field, enter the following variable (with brackets) to use for the first input:
{0}
- c. In the Variable field, enter the input variable name.
- d. Click the **Add** button (plus symbol) to add the variable for the text substitution.
- e. Add the next variable, {1}, and then in the next blank row below the Variable field, enter the input variable name.
If you add a variable by mistake, click the **Remove Variable** button.
- f. Click the **Add** button to add it to the text substitution.
- g. Click the **OK** button to save the variables for the text substitution.

The following image shows an example of variables added for a temperature input value and a date input value:



9. As you finish configuring the applicable rows in the Available Actions area, in the right-most column, click the **Add Action** button to move the configured rows to the "Order of Execution" area.

Next, order the actions as appropriate as described in [Defining the Order of Execution in the Service Request](#).

C.5.5 Defining the Order of Execution in the Service Request

After you move the actions from the Available Actions area to the Order of Execution area, you arrange the actions in the order required to perform the task in EnterpriseOne.

To define the order of execution:

1. Click the action that you want to move up or down in the order.
2. Click the "up arrow" or "down arrow" buttons at the end of the row with the action until the action is in the proper order.
3. Continue until the order of the actions reflects how the task is performed in EnterpriseOne.
4. If you added an action that is not needed, click the **Delete** (X) button at the end of the row to delete the action.
5. If you make a mistake when modifying a service request, click the **Restore Service Request** button to restore the service request to its last saved version.
6. Click the **Save** button to save your changes.
7. Use the UDO buttons described in the [User Defined Object \(UDO\) Features](#) section to move the service request to the appropriate status.

C.5.6 Creating a Custom Java Service Request

You can create a service request that uses custom Java to execute a custom process or to route data into another database. Before you can create a custom Java service request, you must create the custom Java as described in [Chapter 7, "Creating Custom Java for Orchestrations"](#).

1. Access the Service Requests design page from the Orchestrator Studio Home page.
2. On the Service Requests design page, click the **New Custom Java** button.
3. On the Service Request design page, enter a name for the custom Java service request.
Do NOT include spaces in the name.
4. Click the **Edit Description** button to enter a description for the custom Java service request.
5. In the Class field, enter the Java class.
This is the custom Java class that you created to use for the service request. See [Chapter 7, "Creating Custom Java for Orchestrations"](#).
6. Complete the following fields:
 - ❑ **Attribute**. Enter the name of the field in the class.
 - ❑ **Input Value**. Enter the input variable name. If the attribute is a boolean and you pass in "true", then you could use a default value.
 - ❑ **Default Value**. Enter a default, hard-coded value if there is no mapped value. You can also add a hard-coded value to use if the mapped value returns a null.
7. If you make a mistake, you can click the **Restore Custom Java** button which refreshes the custom Java rule to its last saved state.
8. Click the **Save** button.
The Orchestrator Studio saves the custom Java request. You can then access the orchestration in the Orchestration design page and add this custom Java service request as a step in the orchestration.

C.6 Creating Rules

This section contains the following topics:

- [Understanding Rules](#)
- [Creating a Rule](#)
- [Adding a Custom Java Rule](#)

C.6.1 Understanding Rules

A rule contains conditional logic that the Orchestrator uses to evaluate conditions, such as true or false conditions that determine how the orchestration processes the incoming data. You can define a rule with a list of conditions or you can define a more complex rule using a custom Java class. See [Chapter 7, "Creating Custom Java for Orchestrations"](#) for more information about using a custom Java class for rules.

An orchestration rule functions similar to an EnterpriseOne query in that each rule has:

- A "Match Any" or "Match All" setting.
- One or more conditions defined, each being a binary operation on two values.

After creating a rule and adding it to an orchestration, you use the Action column in the Orchestration design page to determine the orchestration step that is invoked when the conditions in the rule are met. See [Defining the Actions Between a Rule and Dependent Components](#) for more information.

C.6.2 Creating a Rule

Create a rule component to define the conditions for an orchestration.

The first time a new rule is saved, it is saved as a "Personal" UDO. Thereafter, you can use the UDO buttons described in the [User Defined Object \(UDO\) Features](#) section to move the rule to the appropriate status.

To create a rule:

1. Access the Rules design page:
 - On the Orchestrator Home page, click the **Rules** icon. And then on the Rules design page, click the **New Rule** button.
 - OR
 - After adding a Rule step to the grid in the Orchestration design page, click the **Edit** button next to the step.

The Orchestrator Studio displays the Rule design page.

2. In the Rule field, enter a name for the rule.

Do NOT include any spaces in the rule name.

3. In the space provided, enter a short description with a maximum of 200 characters. This description appears below the rule name in the component list.
4. Click the **Edit Long Description** button to add a long description to provide more detail about the purpose of the component.
5. Select the Match Value drop-down menu and select one of the following values:
 - **Match All.** Select this option if all conditions in the rule must be met.
 - **Match Any.** Select this option if any conditions in the rule can be met.

6. In the first row in the grid, complete the following columns to add a condition:
 - ❑ **Rule Type.** Click the drop-down menu and select String, Numeric, or Date depending on the format of the input.
 - ❑ **Value 1.** Enter a variable for the input name.
 - ❑ **Operator.** In the drop-down menu, select from the list of operands which include: >, <, >=, <=, =, !=, startsWith, endWith, contains, between, inList.
 - ❑ **Literal.** Click this check box to indicate a literal value.
 - ❑ **Value 2.** If you selected the Literal check box, manually enter a value.
 - ❑ **Literal Value Type.** If you selected the Literal check box, click the drop-down menu and select the format of the literal value: string, numeric, data.
7. Add additional conditions to the rule as needed. If you add a condition by mistake, you can delete it by clicking the **Remove** button at the end of the row with the condition.
8. Click the **Save** button, which saves the rule as "Personal" UDO.
9. Use the UDO buttons described in the [User Defined Object \(UDO\) Features](#) section to move the rule to the appropriate status.

After adding a rule and a service request to an orchestration, in the Orchestration design page, you must define the action for the rule to invoke the service request. See [Defining the Actions Between a Rule and Dependent Components](#) for more information.

C.6.3 Adding a Custom Java Rule

You can create custom Java for a rule and then add it to an orchestration as a rule, which is referred to as a custom Java rule. Before you can add a custom Java rule to an orchestration, you must create the custom Java as described in [Chapter 7, "Creating Custom Java for Orchestrations"](#).

To add a new custom Java rule:

1. Access the Rules design page from the Orchestrator Studio Home page.
2. On the Rules design page, click the **New Custom Java** button.
3. In the Rule field, enter a name for the custom Java rule.
Do NOT include spaces in the custom Java rule name.
4. In the Class field, enter the path to the Java class, which includes the name of the Java class.
This is the custom Java class that you created to use for the rule.
5. Complete the following fields in the grid:
 - ❑ **Attribute.** Enter the name of the field in the class.
 - ❑ **Input Value.** Enter a variable for the input name. If the attribute is a boolean and you pass in "true", then you could enter a hard-coded default value.
 - ❑ **Default Value.** Enter a hard-coded value if there is no mapped value. You can also add a hard-coded value to use if the mapped value returns a null.
6. If you make a mistake while configuring any of the fields, you can click the **Restore Rule** button which refreshes the custom Java rule to its last saved state.
7. Click the **Save** button.

C.7 Creating Cross References

This section contains the following topics:

- [Understanding Cross References](#)
- [Creating a Cross Reference](#)

C.7.1 Understanding Cross References

The Orchestrator uses a cross reference component to map incoming data (third-party data) to an EnterpriseOne value. The EnterpriseOne value identified in the cross reference is considered the output of the cross reference. The output from the cross reference becomes the data passed to another orchestration step.

For each cross reference that you create in the Orchestrator Studio, you have to create a cross reference record in P952000 in EnterpriseOne. When defining cross reference records for orchestrations in P952000, you must use the "AIS" cross reference type. For more information on how to create these cross reference records in P952000, see [Chapter 5, "Setting Up Cross References and White Lists in EnterpriseOne \(P952000\)"](#) in this guide.

Note: If a cross reference lookup fails in an orchestration, the orchestration is terminated.

C.7.2 Creating a Cross Reference

You must define the orchestration inputs before you can create a cross reference. See [Adding Inputs to an Orchestration](#) for more information.

To create a cross reference:

1. Access the Cross Reference design page:

- On the Orchestrator Home page, click the **Cross References** icon. And then on the Cross References design page, click the **New Cross Reference** button.
- OR
- After adding a Cross Reference step to the grid in the Orchestration design page, click the **Edit** button next to the step.

The Orchestrator Studio displays the Cross Reference design page.

2. In the Cross Reference field, enter a name for the cross reference.

Do NOT include spaces in the cross reference name.

3. In the space provided, enter a short description with a maximum of 200 characters. This description appears below the cross reference name in the component list.
4. Click the **Edit Long Description** button to add a long description to provide more detail about the purpose of the component.
5. In the Object Type field, enter a name for the object type.

This is the object type used to categorize the orchestration cross references in EnterpriseOne. See "Adding Orchestration Cross References" in the *JD Edwards EnterpriseOne Tools Interoperability Guide* for more information.

6. Complete the Input Key and Output Key columns to map the inputs to EnterpriseOne fields:

- **Input Key.** The inputs can be one or many values. The inputs must correspond to the value or pipe (|) delimited values in the Third Party Value column in P952000. See "[Chapter 5, "Setting Up Cross References and White Lists in EnterpriseOne \(P952000\)"](#)" for more information.
 - **Output Key.** The outputs can be one or many values. The outputs must correspond to the value or pipe (|) delimited values in the EOne Value column in P952000. See "[Chapter 5, "Setting Up Cross References and White Lists in EnterpriseOne \(P952000\)"](#)" for more information.
7. If you enter any values by mistake, you can click the **X** button next to the field to delete the entry.
 8. Click the **Save** button, which saves the white list as a "Personal" UDO.
The output values in the cross reference are now available for mapping in subsequent orchestration steps.
 9. Use the UDO buttons described in the [User Defined Object \(UDO\) Features](#) section to move the cross reference to the appropriate status.

C.8 Creating White Lists

This section contains the following topics:

- [Understanding White Lists](#)
- [Creating a White List](#)

C.8.1 Understanding White Lists

A white list contains a list of IDs permitted in the Orchestrator. By adding a white list to an orchestration, only inputs with IDs listed in the white list are permitted for processing by the Orchestrator. You add a white list component as a step in the orchestration.

In addition to specifying the permitted IDs in the white list, you must also specify the white list IDs in P952000 in EnterpriseOne. This is the same application that you use to set up and store orchestration cross references. As with cross references, use the "AIS" cross reference type in P952000 for a white list. In P952000, the Third Party App ID should have a value of WHITELIST. The EnterpriseOne value is not used for white lists and will default to NA.

If a white list lookup fails in an orchestration, the orchestration is terminated.

C.8.2 Creating a White List

The first time a new white list is saved, it is saved as a "Personal" UDO. Thereafter, you can use the UDO buttons described in the [User Defined Object \(UDO\) Features](#) section to move the cross reference to the appropriate status.

1. Access the White List design page:
 - On the Orchestrator Home page, click the **White Lists** icon. And then on the White Lists design page, click the **New White List** button.
OR
 - After adding a White List step to the grid in the Orchestration design page, click the **Edit** button next to the step.

The Orchestrator Studio displays the White Lists design page.

2. In the White Lists design page, click the **New White List** button.

3. In the White List, enter a name for the white list.
Do NOT include spaces in the white list name.
4. In the space provided, enter a short description with a maximum of 200 characters. This description appears below the white list name in the component list.
5. Click the **Edit Long Description** button to add a long description to provide more detail about the purpose of the component.
6. In the Object Type field, enter a name for the object type.
The value you enter in the Object Type field must match a cross reference object type in the EnterpriseOne Business Services Cross Reference application (P952000).
The cross reference object type is a named group of records in P952000. For example, you may have thousands of records in P952000. You can use cross reference object types, for example "Equipment" or "Alert_Notification_Recipients" to group cross reference records into manageable categories. You define the cross reference object types as needed. See [Chapter 5, "Setting Up Cross References and White Lists in EnterpriseOne \(P952000\)"](#) for more information.
7. In the Input Key column, enter the input that you want to add as a permitted input.
8. Click the **Save** button.
9. Use the UDO buttons described in the [User Defined Object \(UDO\) Features](#) section to move the white list to the appropriate status.

C.9 Creating Orchestrations

This section contains the following topics:

- [Understanding Orchestrations](#)
- [Creating an Orchestration](#)
- [Adding Inputs to an Orchestration](#)
- [Adding Steps to an Orchestration](#)
- [Configuring Transformations](#)

C.9.1 Understanding Orchestrations

Creating an orchestration in the Orchestrator Studio involves:

- Naming the orchestration and specifying the input format for the orchestration.
The input format can be JDE Standard, Oracle Cloud IoT, or Generic. See [Supported Input Message Formats](#) for more information about which input format to use.
- Adding inputs to the orchestration.
The inputs define the data passed from the device to the orchestration. This may include an ID that identifies the device as well as other data that the orchestration will receive from a device, such as temperature, date, or any other data you want to capture. Each input has a name and a type which can be string, numeric, or various date formats.
- Adding steps to the orchestration.

Each step is a component of the orchestration: a service request, rule, cross reference, or white list. At a minimum, an orchestration requires only a service request step, which provides the actions or the instructions to perform a particular task in EnterpriseOne.

- ☒ Configuring transformations.

You use transformations to map orchestration inputs to inputs defined in each orchestration step, such as inputs in a rule, cross reference, white list, or service request component.

Important: Remember that when you are ready to "request to publish" an orchestration, you need to make sure that you also request to publish all components associated with the orchestration. This enables an administrator to save all required components to the proper directory on the AIS Server for processing by the Orchestrator. If a component is missing, the orchestration process will end in error.

C.9.2 Creating an Orchestration

To create an orchestration:

1. On the Orchestrator Studio Home page, click the **Orchestrations** icon.
2. On the Orchestrations page, click the **New Orchestration** button.
3. On the Orchestration design page, enter a unique name for the orchestration in the **Orchestration** field.

Do NOT use spaces when naming the orchestration.

4. In the space provided, enter a short description with a maximum of 200 characters. This description appears below the orchestration name in the component list.
5. Click the **Edit Long Description** button to provide more detail about the component.

Use this field to describe the purpose of the orchestration and any details that differentiate the orchestration from other orchestrations that you create.

6. Click the **Input Format** drop-down menu and select the appropriate format:

- ☒ **JDE Standard** (default)
- ☒ **Oracle Cloud IoT**
- ☒ **Generic**

See [Supported Input Message Formats](#) for more information about which input format to use.

7. At this point, you can click **Save** before defining inputs or steps for the orchestration.

The Orchestrator Studio saves the orchestration as a "Personal" UDO. Next, add inputs to the orchestration as described in the [Adding Inputs to an Orchestration](#) section.

You can also use the Save As button to create a new orchestration by copying an existing one and renaming it.

Caution: If you create a copy of an orchestration, only the orchestration is copied. The Orchestrator Studio does NOT create a copy of the components that are associated with the orchestration's steps. That is, both the original orchestration and the new orchestration use the same components that comprise the orchestration. Therefore, in the new orchestration, do NOT modify the components in any way that would break other orchestrations that use the same components.

C.9.3 Adding Inputs to an Orchestration

Orchestration inputs identify the data an orchestration consumes from external devices. In the orchestration, you enter names for the inputs. For example you can enter "SensorID" to pass a sensor ID value to an orchestration and you can enter "TemperatureReading" to pass a temperature value to the orchestration.

You also use these orchestration inputs to configure other components used by the orchestration, such as a service request, rule, white list, or cross reference. For example, if the orchestration requires a rule, you use the orchestration inputs to define the conditions for the rule. If an incoming value from the device does not match an EnterpriseOne value, you can create a cross reference that uses the orchestration input to map third-party data to data in EnterpriseOne.

To add the orchestration inputs:

1. In the first empty row in the Orchestration grid, enter the name of the input in the Input column.
2. In the Value Type column, select the input value type. Valid values are:
 - ▀ String
 - ▀ Numeric

If the input is a date, you can use any of the following date formats:

- ▀ dd/MM/yyyy
- ▀ dd/MM/yy
- ▀ yyyy/MM/dd
- ▀ MM/dd/yyyy
- ▀ MM/dd/yy
- ▀ yy/MM/dd

You can also use the following date formats, which create additional inputs derived from the passed value:

- ▀ Milliseconds
- ▀ yyyy-MM-dd'T'HH:mm:ss.SSSZ

3. Click **Save** to save your changes.

C.9.4 Adding Steps to an Orchestration

When you add a service request, rule, cross reference, or white list to an orchestration, you add it as a step in the orchestration. It is recommended that you create the component before adding it as a step in an orchestration.

The Orchestrator Studio provides "Insert Step Before" and "Insert Step After" buttons so you can add a step before or after another step in the orchestration. Therefore, you do not have to determine whether or not you need to add a particular step, such as a white list, to an orchestration before you add other steps.

Figure C-8 shows an orchestration that contains multiple rules, cross references, and service requests, which are displayed in the grid on the Orchestration design page:

Figure C-8 Steps in the AddConditionBased Alert Sample Orchestration in the Orchestrator Studio

Type	Action	Name	
Rule	True	JDE_RULE_Sample_VibrationAlarm Personal RUL_1608250018CUST	✓ <input type="button" value="Edit"/>
Cross Reference	True	JDE_XREF_Sample_SensorLocation Pending Approval XRE_1608250008CUST	✓ <input type="button" value="Edit"/>
Cross Reference	True	JDE_XREF_Sample_AlertNotificationRecipients Shared XRE_1608250008CUST	✓ <input type="button" value="Edit"/>
Service Request	True	JDE_SREQ_Sample_AddCBArt_Alarm Personal SRE_1609270001CUST	✓ <input type="button" value="Edit"/>
Rule	False	JDE_RULE_Sample_VibrationWarning Personal RUL_1608250017CUST	✓ <input type="button" value="Edit"/>
Cross Reference	True	JDE_XREF_Sample_SensorLocation Pending Approval XRE_1608250008CUST	✓ <input type="button" value="Edit"/>
Cross Reference	True	JDE_XREF_Sample_AlertNotificationRecipients Shared XRE_1608250008CUST	✓ <input type="button" value="Edit"/>
Service Request	True	JDE_SREQ_Sample_AddCBArt_Warning Personal SRE_1609020001CUST	✓ <input type="button" value="Edit"/>

In the grid, the Type column shows the type of step and the Name column shows the name of the component for the orchestration step. The "True" or "False" values in the Action column determine the actions the orchestration performs based on the conditions set in the rules. See [Defining the Actions Between a Rule and Dependent Components](#) for information about defining the actions in an orchestration.

To add steps to an orchestration:

1. To add the initial step to an orchestration, click the **Add Step** button (+ symbol).
2. In the "Enter Type of Step" pop-up field, select one of the following steps to add to the orchestration, and then click the **Ok** button.

- **Cross Reference**

- **Rule**

- **Service Request**

- **White List**

The Orchestrator Studio displays the first step in the grid.

3. Add a component to the step:
 - a. To use an existing component for the new step, click the drop-down menu in the Name column and select a component.
 - b. To create a new component for the step, click the **Edit** button (pencil icon) at the end of the row to access the design page for creating the new component.

After creating the component, when you return to the Orchestration design page, you will need to select the component from the drop-down menu in the orchestration steps grid to add it.

See the appropriate topics in this chapter on how to create each type of orchestration component.

4. In the Transformations area on the right side of the page, configure the transformations for each orchestration step as described in the [Configuring Transformations](#) section.
5. To add additional steps to an orchestration:
 - a. Select a step in the grid, and then click either the **Insert Step Before** or **Insert Step After** button to add an additional step before or after the selected step.
 - b. In the "Enter Type of Step" pop-up dialog box, select the step that you want to add.
 - c. Click the **Ok** button.

To remove a step from an orchestration:

Caution: Before you delete a step, make sure the step is not used by any other component in the orchestration.

1. Select the step that you want to remove.
2. Above the grid with the steps, click the **Remove Step** button (the X button).

If you make a mistake when updating an orchestration, click the **Restore All** button in the upper-right corner of the design page to restore the orchestration to its last saved version, which erases any changes made since the last save.

C.9.4.1 Defining the Actions Between a Rule and Dependent Components

The Orchestration design page contains an Action column for defining the actions between a Rule step and other orchestration steps in an orchestration. After you create a rule with conditions and add it as a step to an orchestration, you need to define the action the orchestration takes if the condition in the rule is met. For example, if a rule contains a condition that when met should invoke a Service Request step, then in the Service Request step row, you need to set the Action column to **True**.

You can also define a **False** action to invoke a different orchestration step when a condition in the initial rule is NOT met. A **False** action can invoke a different Service Request step or another Rule step that contains additional conditions for which the incoming data is evaluated against. Thus, you can design an orchestration with as many rules and service requests as your business requirements necessitate.

Figure C-9 shows an example of an orchestration with two Rule steps and two Service Request steps, with the following defined actions:

- For the first Service Request step, the action is set to **True** to instruct the orchestration to invoke this Service Request step when the condition in the first Rule step is met.
- The action for the second (nested) Rule step is set to **False** to instruct the orchestration to invoke this rule when the condition in the first rule is NOT met.
- The action for the second Service Request step is set to **True** to instruct the orchestration to invoke this service request when the condition in the second rule is met.

Figure C–9 Defining the Orchestration Actions

Type	Action	Name
CrossReference		JDE_XREF_Sample_SensorLocation
CrossReference		JDE_XREF_Sample_AlertNotificationRecipients
Rule		JDE_RULE_Sample_CBMAlar_1
ServiceRequest	True	JDE_SREQ_Sample_AddCBArt_Alarm
Rule	False	JDE_RULE_Sample_CBMAlar_2
ServiceRequest	True	JDE_SREQ_Sample_AddCBArt_Alarm

To set the Action column to True or False:

1. In the Orchestration design page, in the appropriate Rule or Service Request step row, click in the Action column.
2. Select either **True** or **False** as appropriate.
3. Click the **Save** button.

After completing the orchestration, you can use the Orchestrator Client to test the orchestration in a test environment. You can access the Orchestrator Client from the drop-down menu in the upper-right corner of the Orchestrator Studio. See [Chapter 9, "Testing Orchestrations in the EnterpriseOne Orchestrator Client"](#) for more information.

C.9.5 Configuring Transformations

Use the Transformations area on the Orchestration design page to map orchestration inputs to inputs defined in an orchestration step, such as inputs in a rule, cross reference, white list, or service request component. The Transformations area includes an Auto Map button that automatically maps any matching inputs in the orchestration and orchestration step.

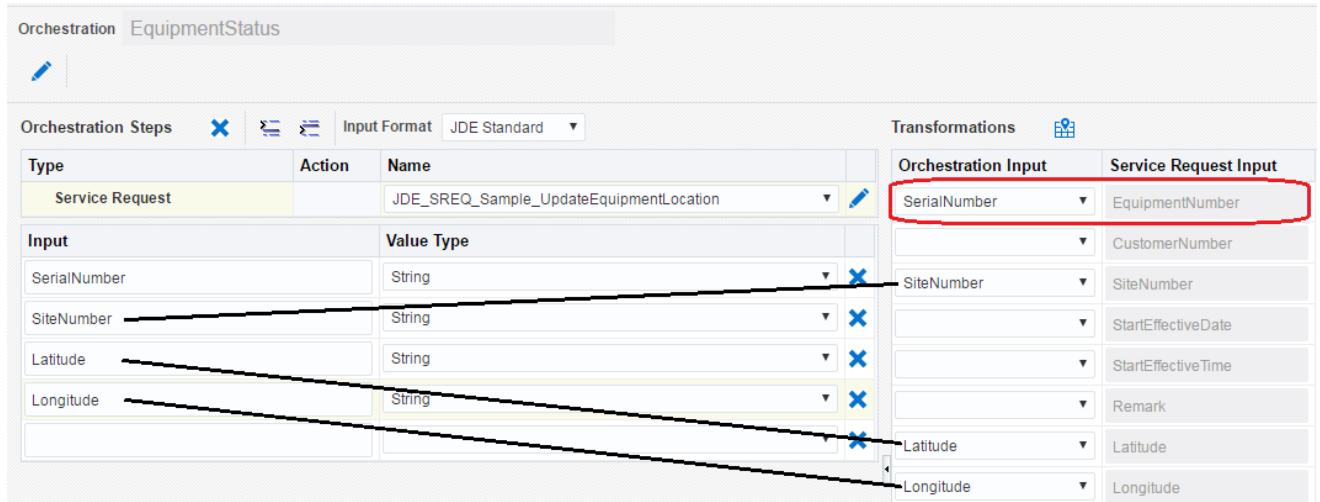
If an orchestration and an orchestration component use different input names to identify the same data, you can use the grid in the Transformations area to map the inputs. This facilitates the reuse of components, so you can avoid having to create a new component simply to match the input names (input from a device) in a new orchestration.

To better understand how you use transformations, consider the following scenario which is depicted in [Figure C–10](#):

- ❑ You have an existing service request that creates alerts in the EnterpriseOne Condition-Based Alerts Revisions application (P1311). This service request includes an input called EquipmentNumber that is mapped to the Asset Number field in P1311.
- ❑ You want to reuse this service request as a step in a new orchestration, but the new orchestration will consume data from devices that supply the EquipmentNumber as "SerialNumber."
- ❑ To reuse the existing service request designed to consume data labeled as EquipmentNumber, in the new orchestration, you can create a transformation to pass the SerialNumber input to the EquipmentNumber input in the service request.
- ❑ Create the new orchestration with SerialNumber as one of the inputs, and then add the existing service request as a step.
- ❑ When you click the Service Request step in the orchestration steps grid, the Service Request column in the Transformations table displays all of the inputs defined in the service request. Notice in [Figure C–10](#) that EquipmentNumber, not SerialNumber, is defined as an input in the Service Request Input column.
- ❑ Next, you click the Transformations Auto Map button, which automatically maps the matching inputs. [Figure C–10](#) shows the SiteNumber, Latitude, and Longitude inputs in the

- Orchestration Input column, which were populated automatically after the Auto Map button was selected.
- Finally, to map the SerialNumber input in the orchestration to the EquipmentNumber input in the service request, you click the drop-down menu in the Orchestration Input column and select SerialNumber, as shown in Figure C–10.

Figure C–10 Transformations Example



Initially, when you have a small number of orchestrations in your system, it is recommended to keep all input names consistent among the orchestration and all components (orchestration steps) used by orchestration. But as your library of orchestrations and orchestration components increases, instead of creating new components with input names that match the orchestration inputs, you can use transformations to map the orchestration inputs to an existing component.

To create transformations:

- In the Orchestration design page, select an orchestration step for which you want to configure transformations.
- Click the Transformations **Auto Map** button to map matching input names. Matching inputs automatically appear in the Orchestration Input column next to the matching input in the "<orchestration step> Input" column.
- To map an orchestration input name to a non-matching input name in an orchestration step:
 - In the Transformations table, select the drop-down menu in the appropriate Orchestration Input row.
 - Select the orchestration input to map to the input name for the orchestration step.

For example, in Figure C–10, SerialNumber is mapped to EquipmentNumber in the Service Request Input.

C.10 Reloading Orchestrations and Orchestration Components

The Reload Files feature enables you to reload orchestrations and orchestration components to the state in which they were last saved in the Orchestrator Studio. Use the Reload Files feature if you do not want to save any modifications that you made.

To reload all files, click the drop-down menu in the upper-right corner of the Orchestrator Studio and then click the **Reload Files** link.

Each individual orchestration component panel also contains a Restore button that you can use to restore an individual component.

C.11 Supported Input Message Formats

The Orchestrator supports three input message formats for orchestrations: a standard JD Edwards EnterpriseOne format, a generic format, and Oracle Cloud IoT format. [Example C-1](#) and [Example C-2](#) show an example of the code for each format.

Example C-1 Standard JD Edwards EnterpriseOne Input Message Format

```
{  
    "inputs": [  
        {  
            "name": "equipmentNumber",  
            "value": "41419"  
        },  
        {  
            "name": "description",  
            "value": "test"  
        },  
        {  
            "name": "date",  
            "value": "1427774400000"  
        },  
        {  
            "name": "time",  
            "value": "12:15:15"  
        },  
        {  
            "name": "temperature",  
            "value": "99"  
        }  
    ]  
}
```

Example C-2 Generic or Oracle Cloud IoT Input Message Format

```
{  
    "equipmentNumber": "41419",  
    "description": "test",  
    "date": "1427774400000",  
    "time": "12:15:15",  
    "temperature": "99"  
}
```

C.11.1 Additional Supported Input Message Formats for the Generic Input Format

Additional formats are supported when using the generic input format, as long as the orchestration input values are defined using the full path to the elements used. You may have a deeper JSON structure like this.

```
{
```

```

"equipmentNumber": "41419",
"equipementDetail": {
    "type": "thermometer",
    "readingDetail": {
        "temperature": 200,
        "units": "F"
    }
}
}

```

To reference the temperature within the orchestration, you can use the full path delimited by periods, for example:

```
equipmentDetail.readingDetail.temperature
```

C.11.2 Differences Between Input Message Formats

The following list describes the differences between the input message formats:

- The iterateOver attribute for the orchestrationStep element is supported only by the standard JD Edwards EnterpriseOne input message format.
- When using the "detail" form action type with the standard format, it will automatically iterate over all detailInputs and repeatingInputs in order to add multiple rows to a grid. If the generic format is used, only a single grid row can be added.

As shown in [Example C-3](#), "detailInputs" would correspond to grid data; "repeatingInputs" would correspond to individual rows that contain "inputs" that correspond to columns.

If you have a power form with two grids, you could populate both grids using two "detailInputs" structures with different "name" values that correspond to the different grids. "repeatingInputs" would contain a list of rows and for each row you would define a list of "inputs".

You could also define a CrossRefReference orchestration step with iterateOver="GridData" that converts the item value into an EnterpriseOne specific item number. For example, if A123=220 and A124=230, the single orchestration step would convert both.

Example C-3

```
{
    "inputs": [
        {
            "name": "BranchPlant",
            "value": "30"
        },
        {
            "name": "customer",
            "value": "4242"
        }
    ],
    "detailInputs": [
        {
            "name": "GridData",
            "repeatingInputs": [
                {
                    "inputs": [
                        {
                            "name": "item",
                            "value": "A123"
                        },
                        {
                            "name": "item",
                            "value": "A124"
                        }
                    ]
                }
            ]
        }
    ]
}
```

```
{  
    "name": "Quantity",  
    "value": "3"  
}  
]  
},  
{  
    "inputs": [  
        {  
            "name": "item",  
            "value": "A124"  
        }  
    ]  
}  
]  
]  
}
```

C.12 Setting Up Orchestration Security

Before the EnterpriseOne Orchestrator can process an orchestration, authentication of the JD Edwards EnterpriseOne user ID and password must take place. It is the responsibility of the originator of the service request to tell the orchestration about the user. The user's credentials must be supplied in a basic authorization header or in the JSON body of the request. The user must also have authorized access to the EnterpriseOne application in which the resulting transaction takes place. The following code is an example of credentials in the JSON body of the request:

```
{  
    "username": "JDE",  
    "password": "JDE",  
    "environment": "JDV900",  
    "role": "*ALL"  
}
```

The AIS service used with orchestrations is stateless; each call passes credentials to establish a separate EnterpriseOne session. After the transaction is complete, the session closes.

In addition to passing credentials for authentication, you can employ a second level of security for the Orchestrator through whitelisting. Whitelisting enables an initial rudimentary pass/fail check of the incoming device signature against a predefined list of signatures. A white list provides an additional layer of security to the Orchestrator security. If a value passed to the Orchestrator is not a valid value included in the orchestration's white list, the Orchestrator rejects the input. For more information, see [Creating White Lists](#) in this guide.

C.12.1 Restricting Access to Exposed Orchestrations

If you expose orchestrations for business partners or customers to invoke, it is recommended to use an http proxy to allow access to only the endpoints required to execute the orchestration. Configure the proxy to restrict all endpoints except /orchestrator, /discover, /tokenrequest, and /tokenrequest/logout. This allows external users set up with the proper UDO security to discover and call orchestrations.

C.13 Exporting Orchestration Components from the Orchestrator Studio

Caution: Do not use the Export and Import tools to share orchestration component files with other users. With orchestration components stored as UDOs in EnterpriseOne, the management of orchestration components, including the sharing and modifying of shared orchestration components and promoting of orchestration components to the appropriate AIS Server directory for test or production purposes, is administered through the life cycle management tools in EnterpriseOne.

You can export any of the orchestration component types from the Orchestrator Studio to view the source XML files of the components. This is only recommended for advanced users for troubleshooting purposes or for making manual customizations. The Orchestrator Studio exports the source XML files in a zip file.

To export an *orchestration* component, the Orchestrator Studio gives you the option to export only the selected orchestration component or to export the orchestration component and all components associated with it. For example, if an orchestration has a service request component and a rule component associated with it, if you export all components, the zip file will contain XML files for the orchestration, service request, and rule.

To export orchestration components:

1. On a component design page, select the component to export and then click the **Export File** button in the upper-right corner.
If you are exporting an orchestration, a dialog box appears in which you can click **All** or **Orchestration Only**.
2. Follow the instructions in the browser to save the zip file to a location on your local machine.

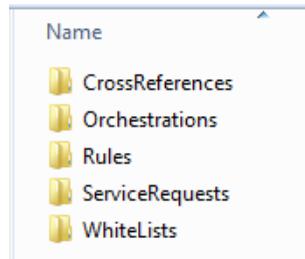
C.14 Importing Orchestration Files in the Orchestrator Studio

Caution: Do not use the Export and Import tools to share orchestration component files with other users. With orchestration components stored as UDOs in EnterpriseOne, the sharing and modifying of shared orchestration components, as well as the promoting of orchestration components to the appropriate AIS Server directory for test or production purposes, is administered through the life cycle management tools in EnterpriseOne.

You might need to import orchestration component XML files that were exported from the Orchestrator Studio for advanced troubleshooting or customization purposes. The Orchestrator Studio Import tool enables you to import individual orchestration XML files or a zip file containing XML files.

Caution: You cannot import orchestration component files that were exported from the EnterpriseOne Object Management Workbench - Web application. The zip file created from an OMW - Web export stores the XML files in a structure that cannot be read by the Orchestrator Studio Import tool.

If importing a zip file that contains orchestration XML files and dependent orchestration component XML files, the zip should contain the following folders with each of the XML files stored in the appropriate folder:



If you import an XML file that has the same name as a current orchestration component (UDO) in the Orchestrator Studio, you have the following options:

- ❑ If the UDO is at a status of "Personal" or "Reserved," you can overwrite the UDO with the XML file.
- ❑ If the UDO is in a "Shared" status, you cannot overwrite it. But you are given the option to import it as a new UDO with a status of "Personal."

To import files:

1. On the Orchestrator Studio Home page, click the **Tools** link in the upper-right corner.
2. On the Tools page, click the **Import Files** icon.
3. On Import Files, click the **Choose File** button.
4. Locate the orchestration component XML files or zip file that contains the orchestration component files that you want to import.

After selecting the files to import, the Import tool checks the XML files that you selected against the current UDOs in the Orchestrator Studio and displays the options that are available for importing the files, as shown in the example in [Figure C-11](#).

Figure C–11 Orchestrator Studio Import Tool

The screenshot shows the 'Select File for Import' screen in the Orchestrator Studio. At the top, it displays 'Orchestrations-1461866371912.zip' and an 'Update...' button. Below that, it shows 'Number of Files 9' and 'Number of Files Existing on Server 2'. On the right, there is a 'Submit' button and a 'Select All' checkbox with a checked status icon.

Cross Reference		
JDE_XREF_Sample_SensorLocation.xml	No update allowed at current status	<input type="checkbox"/>
JDE_XREF_Sample_AlertNotificationRecipients.xml	Select to add or else reserve record and re-import to update	<input type="checkbox"/>
Service Requests		
JDE_SREQ_Sample_AddCBArt_Warning.xml	Overwrite existing file	<input checked="" type="checkbox"/>
JDE_SREQ_Sample_AddCBArt_Alarm.xml	File is ready to submit	<input checked="" type="checkbox"/>
Rules		
JDE_RULE_Sample_CBMArt_3.xml	File is ready to submit	<input checked="" type="checkbox"/>
JDE_RULE_Sample_CBMArt_1.xml	File is ready to submit	<input checked="" type="checkbox"/>
JDE_RULE_Sample_CBMArt_2.xml	File is ready to submit	<input checked="" type="checkbox"/>
JDE_RULE_Sample_CBMWarning.xml	File is ready to submit	<input checked="" type="checkbox"/>
Orchestrations		
JDE_ORCH_Sample_AddConditionBasedAlert_Generic.xml	File is ready to submit	<input checked="" type="checkbox"/>

At the bottom right, there is another 'Submit' button.

5. Review the import option for each file to determine how to proceed with the import. The options are:
 - ❑ No update allowed at current status.
The XML file name is the same as an existing component, which has a status of "Pending Approval," so it cannot be overwritten.
 - ❑ Select to add or else reserve record and re-import to update.
A component with the same name exists in the Orchestrator Studio in a "Shared" status. Click the check box if you want to import the file as a new UDO. A new name will be generated for the new component upon import.
If you want to overwrite the current component in the Orchestrator Studio, clear the check box. And then change the status of the current component to "Reserved" and reimport the XML file to overwrite the component.
 - ❑ File already exists.
The XML file matches a component that is in a "Personal" or "Reserved" status. Click the check box to import it and overwrite the current component.
 - ❑ File is ready to submit.
The XML file is unique and does not already exist in the Orchestrator Studio. Click the check box to import it.
6. You can also click the **Select All** check box to import all XML files that are available for importing.
7. Click the **Submit** button.

The Orchestrator Studio imports the selected components into the components list on the respective design page.

D

Creating Orchestrations with the Orchestrator Studio 2.0 (Release 9.2.0.5)

Important:

This appendix describes how to use Orchestrator Studio 2.0 that was available starting with EnterpriseOne Tools 9.2.0.5.

To use Orchestrator Studio 7.0.x.0, the latest version available with EnterpriseOne Tools 9.2.3, see:

- [Chapter 2, "Implementing the Orchestrator Studio"](#) for installation instructions
- [Chapter 4, "Creating Orchestrations with Orchestrator Studio 7.x.x.x"](#)

This appendix contains the following topics:

- [Section D.1, "Understanding the Orchestrator Studio and Orchestrations"](#)
- [Section D.2, "Accessing the Orchestrator Studio"](#)
- [Section D.3, "Navigating the Orchestrator Studio"](#)
- [Section D.4, "Creating Service Requests"](#)
- [Section D.5, "Creating Rules"](#)
- [Section D.6, "Creating Cross References"](#)
- [Section D.7, "Creating White Lists"](#)
- [Section D.8, "Creating Orchestrations"](#)
- [Section D.9, "Reloading Orchestrations and Orchestration Components"](#)
- [Section D.10, "Setting up Orchestration Security"](#)
- [Section D.11, "Exporting Files from the Orchestrator Studio"](#)
- [Section D.12, "Importing Orchestration Files in the Orchestrator Studio"](#)

D.1 Understanding the Orchestrator Studio and Orchestrations

The Orchestrator Studio is a web-based application for creating orchestrations and the components that comprise an orchestration. All orchestrations and orchestration components that you create in the Orchestrator Studio are saved as separate files in an orchestration directory on the EnterpriseOne Application Interface Services (AIS) Server. The Orchestrator uses these files to process a single orchestration instance on the AIS Server.

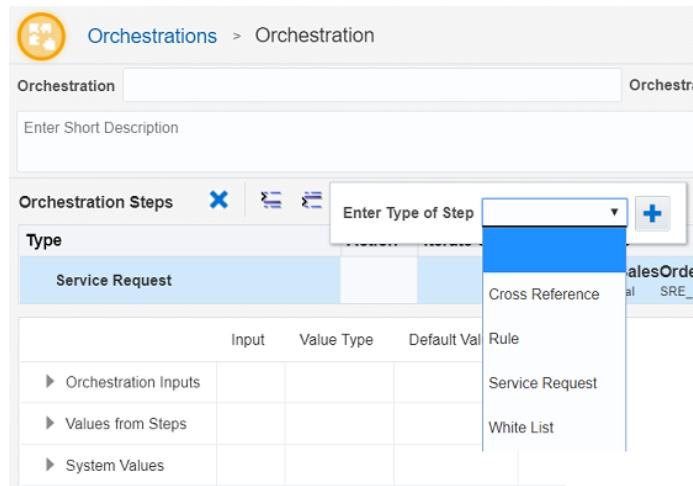
Use the Orchestrator Studio to create the following components:

- ❑ **Orchestrations.** An orchestration is the master component that provides a unique name for an orchestration process. The orchestration is where you define the inputs for the orchestration, the expected incoming data from a device. The orchestration also includes orchestration steps, which are invocations to the other components described in this list. When the Orchestrator invokes an orchestration, it processes the steps defined in the orchestration to enable the transfer of data from third-party systems to EnterpriseOne.
- ❑ **Service Requests.** A service request contains a sequence of actions to perform a particular process in EnterpriseOne. The Orchestrator uses the service request as an invocation of a JD Edwards EnterpriseOne interactive application or a Java application via a REST service call to the AIS Server.
- ❑ **Rules.** A set of conditions that the input to the orchestration is evaluated against to produce a true or false state. With rules, a false outcome or true outcome can invoke further orchestration steps. You can also nest rules, in which an outcome of one rule can invoke a different rule, to produce complex evaluations. You can also use custom Java to define rules.
- ❑ **Cross References.** A set of data relationships that map third-party values to EnterpriseOne values. For example, a device's serial number can be cross-referenced to a JD Edwards EnterpriseOne Equipment Number for use in service requests.
- ❑ **White Lists.** A white list contains an inclusive list of values permitted in the orchestration and terminates the orchestration process if the data is not recognized.

Note: A simple orchestration requires at a minimum an orchestration and a service request to run.

In the Orchestrator Studio, each component that you create can be added as a step in an orchestration. [Figure D–1](#) shows the drop-down list of steps. Each step in an orchestration is simply a reference to a cross reference, rule, service request, or white list component.

Figure D–1 Orchestrations Steps in the Orchestrator Studio



D.1.1 Reusable Orchestration Components

Orchestration components are reusable. A single component, such as a rule or cross reference, can be included as a step in more than one orchestration. Because components are reusable,

before modifying an existing component, the Orchestrator Studio has a "Where Used" button that you can use to view a list of all orchestrations where the component is used.

If a component is used by more than one orchestration, you should evaluate how it is used in other orchestrations before modifying it. You must not modify the component in any way that would break the functionality of other orchestrations.

The Orchestrator Studio also provides a copy feature so you can create a new component by copying an existing component. When you copy an existing component, you give the copy a new, unique name. The original component is preserved, enabling you to modify the new component as necessary, thereby eliminating the risk of breaking other orchestrations where the original component is used.

D.2 Accessing the Orchestrator Studio

The Orchestrator Studio is a web application that you access through a URL in a web browser. Your system administrator should provide you with this URL.

Important: Orchestration components created in the Orchestrator Studio are saved to a directory on the AIS Server. An administrator must define the path to the AIS Server in the Orchestrator Studio before users can use the Orchestrator Studio. See [Define the Path to the AIS Server in the Orchestrator Studio](#).

To access the Orchestrator Studio:

1. In a web browser, enter the URL to the Orchestrator Studio:

`http://<adf_server>:<port>/OrchestratorStudio/faces/index.jsf`

2. On the Orchestrator Studio Sign In screen, enter your EnterpriseOne User credentials, environment, and role.

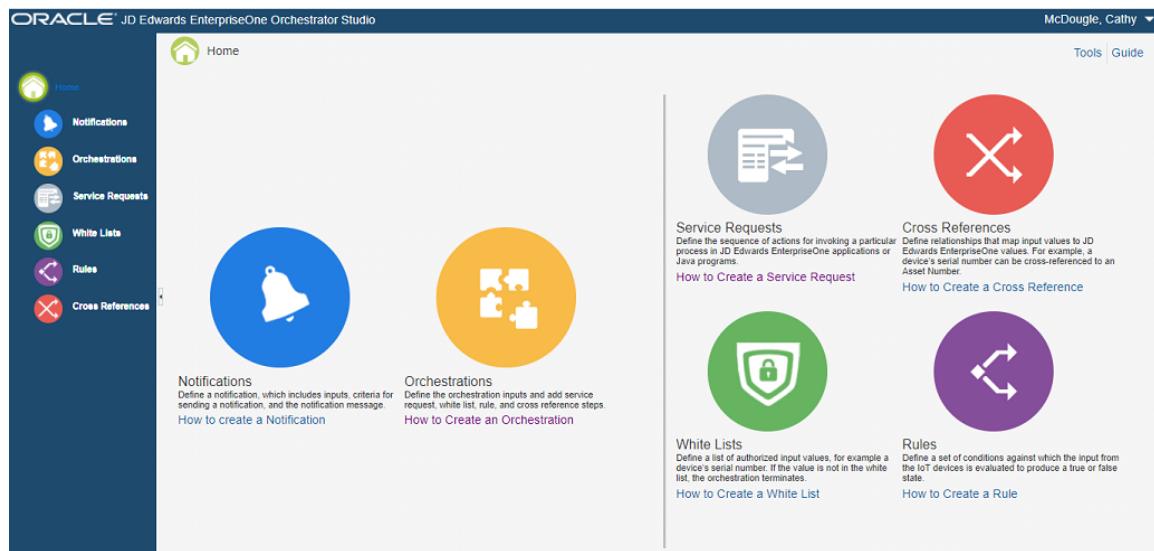
Note: It is highly recommended that you enter an EnterpriseOne environment used for testing, not a production environment.

3. Click the **Login** button.

Before you begin using the Orchestrator Studio, click the drop-down menu in the upper-right corner to view the path to the AIS Server directory where orchestrations created in the Orchestrator Studio are saved. The drop-down menu also provides a link to log out of the Orchestrator Studio.

D.3 Navigating the Orchestrator Studio

The first page that appears after you log in to the Orchestrator Studio is the Home page. The icons for each of the orchestration components on the Home page take you to the design pages for creating and modifying orchestration components. At the top left of the Home page is a Home icon, which you can click to display a side panel that provides another way to access the design pages for the orchestration components. You can also access this side panel from the component design pages for easy navigation between the different component design pages. [Figure D-2](#) shows the Home page with the side panel enabled:

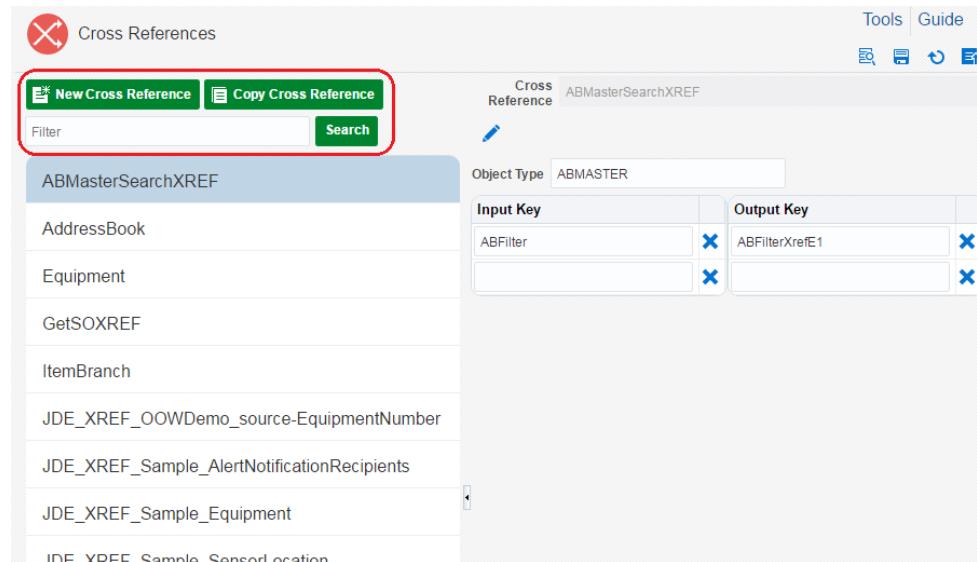
Figure D–2 Orchestrator Studio Home

The Tools link in the upper-right corner of the Home page provides access to the Orchestrator Studio Tools page. This page provides access to the Orchestrator Client for testing orchestrations, the Import tool for importing orchestration files, and a link to the JD Edwards EnterpriseOne web client. For more information about the Orchestrator Client and the Import tool, see the following topics:

- [Chapter 9, "Testing Orchestrations in the EnterpriseOne Orchestrator Client"](#)
- [Section D.12, "Importing Orchestration Files in the Orchestrator Studio"](#)

D.3.1 Navigating Orchestrator Studio Design Pages

Accessed from the Orchestrator Studio Home page, the design page for each component displays a list of existing components. It also provides buttons for creating a new component, copying a component, and searching for a component, as shown in Figure D–3.

Figure D–3 Example of Cross References Design Page

On a component design page, you can click and drag the vertical divider next to the component list to adjust the size of the list. Or you can click the raised tab in the center of the divider to hide or show the component list.

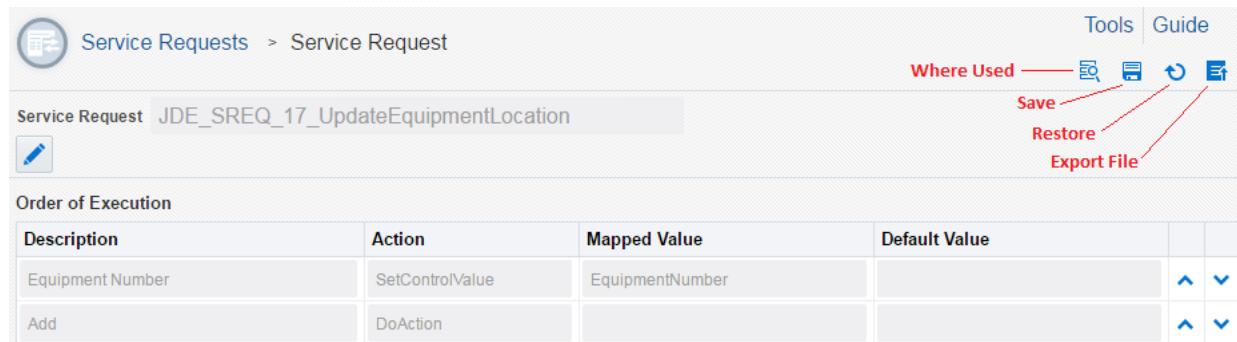
For Cross References, White Lists, and Rules, the right side of the design page contains the controls and fields to create or modify the component.

The initial Orchestrations and Service Requests design pages also contains a list of existing components. However, when creating or modifying an orchestration or service request, the Orchestrator Studio takes you to a different design page to complete the components.

On any design page in which you create or modify a component, the Orchestrator Studio provides the following controls, which are highlighted in [Figure D–4](#):

- **Where Used.** When you select a component from the list of existing components, you can click this button to view other orchestrations where the component is used. This button is not available in the Orchestrations design page because an orchestration cannot be reused.
- **Save.**
- **Restore <component>.** When revising a component, use this button to restore the component to its last saved state.
- **Export File.** Use this button to export the component file to your local machine. See [Exporting Files from the Orchestrator Studio](#) in this appendix for more information.

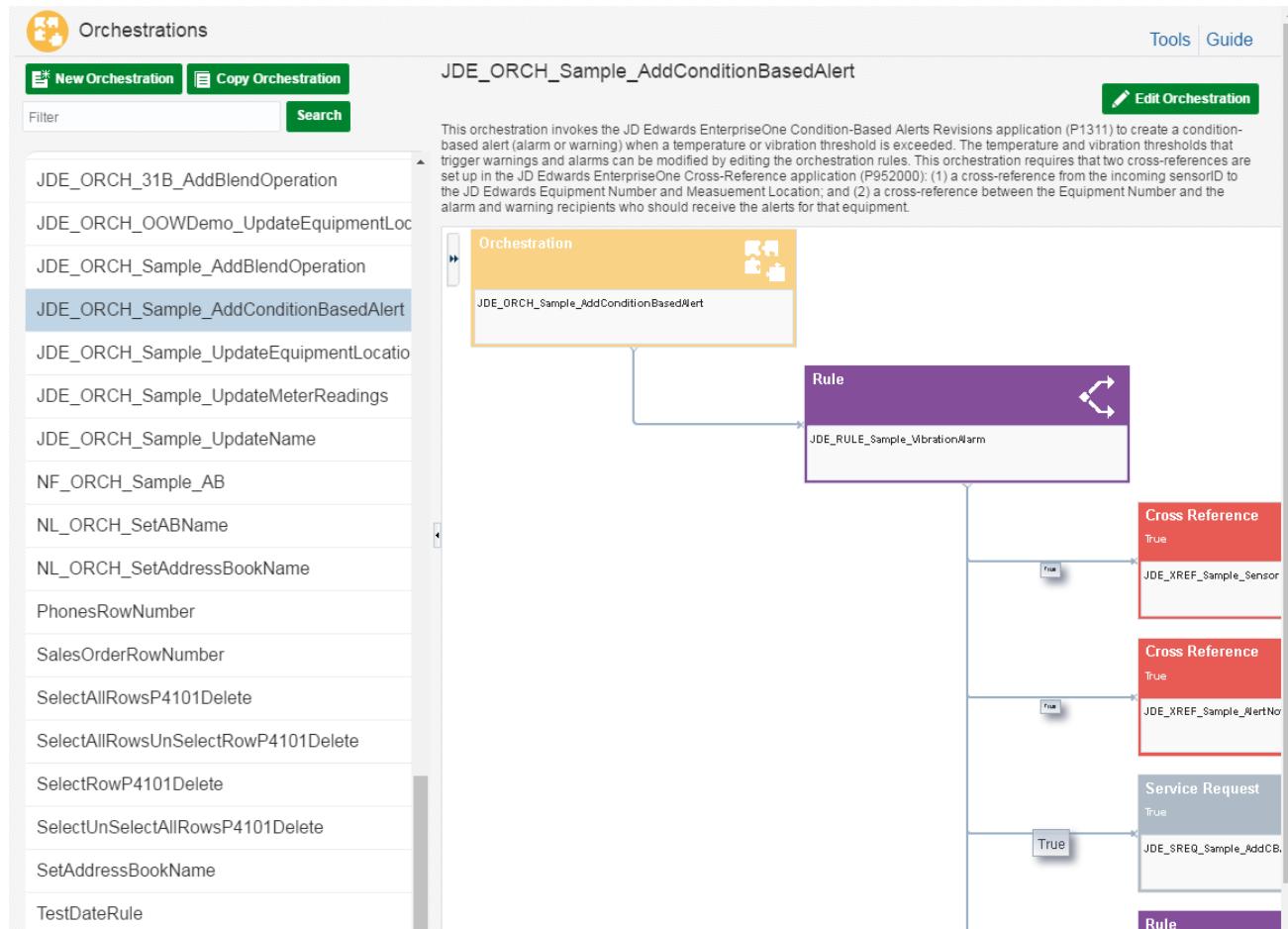
Figure D–4 Example of Service Request Design Page



D.3.2 Working with the Graphical Representation of an Orchestration

The initial Orchestrations design page shows a graphical representation of a selected orchestration and all components associated with the orchestration. [Figure D–5](#) shows the Orchestrations design page with an example of an orchestration with multiple components.

Figure D–5 Initial Orchestrations Design Page



The graphic area includes the following features to help you work with graphical representation of the orchestration:

- ¤ Navigation toolbox

In the upper-left corner of the graphic area, you can click the raised tab to display a toolbox for navigating the graphic. Use the directional controls in the toolbox to pan left, right, up, and down, as well as zoom in or zoom out. Or you can click the "Zoom to Fit" button to display the entire graphical representation in the window. Use the layout buttons to change the layout to vertical, horizontal, tree, radial, or circle. Changing the layout can help with viewing more complex orchestrations that contain multiple components.

- ¤ Informational hover help

You can hover your mouse over a component in the graphical area to view an enlarged image of the component. Hovering over the labels on the lines between a rule component and its child components magnify the "True" or "False" label. A "True" label indicates that the child component will be invoked if the conditions in the rule are met. A "False" label indicates that the child component will be invoked when the condition of the rule is not met.

- ¤ Isolate and Restore buttons

The left side of each component contains an Isolate button that you can click to show only that component in the graphic area. You can then click the Restore button to display the entire graphical representation of the orchestration.

- Access to the design page for editing the component

When you click a box representing a component, the Orchestrator Studio takes you to the design page for modifying that particular component.

D.4 Creating Service Requests

This section contains the following topics:

- [Section D.4.1, "Understanding Service Requests"](#)
- [Section D.4.2, "Understanding the Service Request Design Page"](#)
- [Section D.4.3, "Creating a Service Request"](#)
- [Section D.4.4, "Configuring the Actions in the Service Request"](#)
- [Section D.4.5, "Defining the Order of Execution in a Service Request"](#)
- [Section D.4.6, "Creating a Custom Java Service Request"](#)

D.4.1 Understanding Service Requests

At a minimum, an orchestration requires orchestration inputs and a single service request component to run. The service request provides the instructions that the Orchestrator uses to invoke and perform a particular task in EnterpriseOne. Before you create a service request, you must first identify the EnterpriseOne task or business process that you want the service request to perform. See [Identifying the Service Request Information for the Orchestration](#) for more information.

Use the Service Request design page in the Orchestrator Studio to define a service request, which involves:

- Specifying the EnterpriseOne buttons, fields, columns, menu items, and so forth that are involved in performing a task in EnterpriseOne.
- In the Service Request design page, you can search on an EnterpriseOne application form and view all of the controls and fields for a form.
- Determining the order or execution in which the controls and fields are used to perform the desired task in EnterpriseOne.

Note: You can also use custom Java to execute a custom process or to route data into another database. See [Chapter 7, "Creating Custom Java for Orchestrations"](#) for more information.

D.4.1.1 Understanding the Application Stack Option

The Application Stack option in the Service Request design page enables you to create a service request that establishes a session for a specific application and maintains that session across calls. With application stack processing, the service request can invoke multiple forms in different applications to complete a business process. Without the application stack processing, each form in the service request is opened independently.

If you use the Application Stack option, the service request must include instructions for navigating between forms. Without the navigation, additional forms will not be called.

D.4.2 Understanding the Service Request Design Page

The Service Request design page contains two areas: the "Order of Execution" area and the "Available Actions" area. The Available Actions area serves as the workspace for finding and configuring all of the EnterpriseOne controls and fields that make up the sequence of actions in the Order of Execution area. You locate and configure the controls and fields in the Available Actions area, and then move each item to the Order of Execution and place them in the proper sequence.

Available Actions Area

In the Available Actions area, you can search on an EnterpriseOne form to load all of the available controls (menu items, buttons, check boxes, and so forth) and fields for that form. If the service request needs to invoke more than one application to complete the task, you can load the controls and fields of additional application forms as needed. You can select the Application Stack check box to enable application stack processing as described in the [Understanding the Application Stack Option](#) section.

Figure D–6 shows the results of loading an EnterpriseOne application form in the Available Actions grid.

Figure D–6 Available Actions Area in the Service Request Design Page

Available Actions		Application Stack <input checked="" type="checkbox"/>		Run Synchronously <input checked="" type="checkbox"/>		Bypass Form Processing <input type="checkbox"/>	
Application	P1311	Form	W1311B - Condition-Based Alerts Revisions	Version			Form Mode
Description		Mapped Value	Default Value	ID	Version	Form Mode	Return
Condition-Based Alerts Revisions				P1311_W1311B			
Buttons and Exits							
Cancel				12			
Create W.O.				114			
Failure Analysis				124			
Investigation				120			
Investigation Msg				115			
Notification				119			

The columns in the Available Actions grid provide interactive features for configuring the controls and fields before you add them to the Order of Execution area. The following list describes how to use each column:

- ¤ **Description**

This column displays the controls and fields for each form in a collapsible/expandable parent node named after the EnterpriseOne form. Child nodes categorize other items, and include a Buttons and Exits node, a node for displaying the available Query by Example (QBE) fields in a grid (if applicable), and a node for displaying all of the available columns in a grid.

Note: The separate nodes for QBE fields and Grid columns are available starting with Orchestrator 2.0, which is available starting with EnterpriseOne Tools 9.2.0.5.

Mapped Value

Use this column to define the input variable name. Later, when you add the service request to an orchestration, you map the orchestration input to this name. You can only map inputs to a field if 1) the EnterpriseOne field is an editable field and 2) you defined the inputs in the orchestration as described in [Adding Inputs to an Orchestration](#).

Default Value

Enter a default, hard-coded value if there is no mapped value. You can also add a hard-coded value to use if the mapped value returns a null.

"Text substitution" check box column

This check box enables you to add text substitution for an input. Text substitution enables you to combine input values into a text string for input into an EnterpriseOne field.

ID

This column displays the ID of the control or field in EnterpriseOne. This is for informational purposes only.

Form Mode

If a form mode was selected, this column indicates if a field or control is only available in the selected form mode.

Return

You can specify any values that you want returned in the orchestration response after EnterpriseOne actions are invoked. If you do not specify any return values, all values on the form will be returned. Return values may be used by customized third-party gateway programs or devices to do further processing based on the values returned from EnterpriseOne applications.

Return Name

This field is automatically enabled when the Return check box is selected. When enabled, you can manually enter a name in this column, which makes this value available for mapping in subsequent orchestration steps when the service request is added to an orchestration.

Select First Row

Move this action to the Order of Execution if the service request tasks involves selecting the first row in the grid.

After you configure each applicable control or field in the Available Actions area, you click the **Add Action** button in the last column to add each item as an action in the Order of Execution area at the top of the design page.

Order of Execution Area

When actions are added to the Order of Execution grid, you can reorder them using the up and down arrow buttons to the right of each row. You can also delete any actions using the Delete (X) button. [Figure D-7](#) shows the Order of Execution area.

Figure D-7 Order of Execution in the Service Request Design Page

Order of Execution					
Description	Action	Mapped Value	Default Value	Up	Down
Odometer	SetCheckboxValue	on		^	▼
Fuel Meter	SetCheckboxValue	on		^	▼
Hour Meter	SetCheckboxValue	on		^	▼
	SetCheckboxValue	on		^	▼
Subledge	SetCheckboxValue	on		^	▼
	SetCheckboxValue	on		^	▼

D.4.3 Creating a Service Request

To create a service request:

1. Access the Service Request design page:
 - ¤ On the Orchestrator Home page, click the **Service Requests** icon.
 - OR
 - ¤ After adding a Service Request step to an orchestration, click the **Service Request** step row.
 The Orchestrator Studio displays the initial Service Requests design page.
2. On this page, click the **New Service Request** button.
3. On the Service Request design page, in the Service Request field, enter a name for the service request.

Do NOT include any spaces in the service request name.

As an alternative, you can create a new service request by copying an existing service request. To do so:

 - a. On the Service Requests design page, select a service request that you want to copy from the list, and then click the **Copy Service Request** button.
 - b. In the pop-up box, in the New Service Request field, enter a name for the service request.
 - c. Click **Continue**.
4. Click the **Save** button to save the new service request component, and then proceed to configure the actions in the service request as described in the next section.

D.4.4 Configuring the Actions in the Service Request

In the Service Request design page, use the Available Actions area to define actions that you want the service request to perform. After you define the actions, you can place the actions in the proper order in the Order of Execution area as described in [Defining the Order of Execution in a Service Request](#).

To configure the actions for the service request:

1. In the Available Actions area, complete the following fields to specify the EnterpriseOne form that contains the controls you want to load:

- **Application.** Enter the ID of the EnterpriseOne application.
- **Form.** Enter the form ID.
- **Version.** Enter the version of the EnterpriseOne application.
- **Form Mode.** (Available with Release 9.2.0.3) Click the drop-down menu and select the appropriate form mode: **Add**, **Update**, or **Inquiry**.

The form mode determines the controls that are displayed for the form. At runtime, the controls that appear on an EnterpriseOne form are dependent on the form mode as specified in Form Design Aid (FDA). This ensures that you can see all of the form controls in the Orchestrator Studio that are available in the form at runtime.

- **Application Stack.** Click this check box if you want to use application stack processing. See [Understanding the Application Stack Option](#) for more information.
- 2.** Click the **Load Form** button.

The Orchestrator Studio loads the controls and fields for the form in the grid.

If the action involves more than one form, you can search on another form and the Orchestrator Studio will load the controls for the form in a collapsible node at the end of the grid.

- 3.** To map service request inputs to EnterpriseOne fields:
 - a. Select the row with the field in which you want to be mapped.
 - b. In the Mapped Value column, enter a variable name to use for the mapping.

When a service request is added to an orchestration, the inputs in the orchestration should match the inputs defined in the service request. You can also use the "Transformations" feature to map non-matching input names. See [Adding Inputs to an Orchestration](#) and [Configuring Transformations](#) for more information.
- 4.** If the service request task involves selecting the first grid row in the EnterpriseOne form, in the "Select First Row" row, click the **Add Action** button to move it to the "Order of Execution" area.
- 5.** If the service request task requires updating a particular row in an input capable grid, then map the appropriate input value containing the row number to the "Row Number for Update" row, or specify the row number in the Default Value column.
- 6.** In the Default Value column, you can perform the following actions to configure the default values for the controls:
 - a. If the control is a check box or a radio button, you can select it to include it. If there are multiple radio buttons on a form, select only one. If you select more than one, only the last radio button in the Order of Execution list will be used.
 - b. If the control is a combo box (a list of items in a drop-down menu), then the Studio displays a drop-down menu in the Default Value column in which you can select the appropriate item from the list.
 - c. If the field is editable, you can enter a hard-coded value to map for the input. You can also enter a value to be used if the mapped value returns a null.

Fields that are grayed out in this column are not editable.

- 7.** Select the **Return** check box next to the fields that contain values that you want returned in the orchestration response. If you do not specify any return values, all values on the form will be returned.

Important:

If using the Application Stack option:

- Only values from the last form in the application stack can be returned; return controls specified for any form except the last form are ignored.
- If you use a form to add or update a value that upon saving exits to another form (such as a Find/Browse form), the return values will come from the last form that appears after the save. In this scenario, you need to add the last form to the application stack and select the return controls from the last form.

8. To use text substitution, which enables you to combine input values into a text string for input into an EnterpriseOne field, perform these steps:

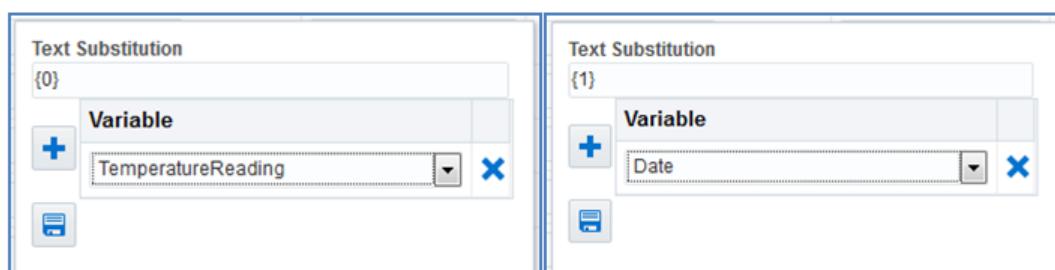
Tip: For example, you might have a string with "Temp was {0} at {1}" that contains a temperature input value and a time input value.

- a. In the field that you want to substitute the input with text, click the **Add Text Substitution** button (plus symbol).
- b. In the pop-up dialog box, in the Text Substitution field, enter the following variable (with brackets) to use for the first input:
{0}
- c. In the Variable field, enter the input variable name.
- d. Click the **Add** button (plus symbol) to add the variable for the text substitution.
- e. Add the next variable, {1}, and then in the next blank row below the Variable field, enter the input variable name.

If you add a variable by mistake, click the **Remove Variable** button.

- f. Click the **Add** button to add it to the text substitution.
- g. Click the **OK** button to save the variables for the text substitution.

The following image shows an example of variables added for a temperature input value and a date input value:



9. As you finish configuring the applicable rows in the Available Actions area, in the right-most column, click the **Add Action** button to move the configured rows to the "Order of Execution" area.

Next, order the actions as appropriate as described in [Defining the Order of Execution in a Service Request](#).

D.4.5 Defining the Order of Execution in a Service Request

After you move the actions from the Available Actions area to the Order of Execution area, you arrange the actions in the order required to perform the task in EnterpriseOne.

To define the order of execution:

1. Click the action that you want to move up or down in the order.
2. Click the "up arrow" or "down arrow" buttons at the end of the row with the action until the action is in the proper order.
3. Continue until the order of the actions reflects how the task is performed in EnterpriseOne.
4. If you added an action that is not needed, click the **Delete** (X) button at the end of the row to delete the action.
5. Click the **Save** button next to the Orchestration drop-down menu to save the orchestration with the service request.
6. If you make a mistake when modifying a service request, click the **Restore Service Request** button to restore the service request to its last saved version.

D.4.6 Creating a Custom Java Service Request

You can create a service request that uses custom Java to execute a custom process or to route data into another database. Before you can create a custom Java service request, you must create the custom Java as described in [Chapter 7, "Creating Custom Java for Orchestrations"](#) in this guide.

1. Access the Service Requests design page from the Orchestrator Studio Home page.
2. On the Service Requests design page, click the **New Custom Java** button.
3. On the Service Request design page, enter a name for the custom Java service request.

Do NOT include spaces in the name.

4. Click the **Edit Description** button to enter a description for the custom Java service request.
5. In the Class field, enter the Java class.

This is the custom Java class that you created to use for the service request.

6. Complete the following fields:
 - **Attribute.** Enter the name of the field in the class.
 - **Input Value.** Enter the input variable name. If the attribute is a boolean and you pass in "true", then you could use a default value.
 - **Default Value.** Enter a default, hard-coded value if there is no mapped value. You can also add a hard-coded value to use if the mapped value returns a null.
7. If you make a mistake, you can click the **Restore Custom Java** button which refreshes the custom Java rule to its last saved state.
8. Click the **Save** button.

The Orchestrator Studio saves the custom Java request. You can then access the orchestration in the Orchestration design page and add this custom Java service request as a step in the orchestration.

D.5 Creating Rules

This section contains the following topics:

- ❑ [Section D.5.1, "Understanding Rules"](#)
- ❑ [Section D.5.2, "Creating a Rule"](#)
- ❑ [Section D.5.3, "Adding a Custom Java Rule"](#)

D.5.1 Understanding Rules

A rule contains conditional logic that the Orchestrator uses to evaluate conditions, such as true or false conditions that determine how the orchestration processes the incoming data. You can define a rule with a list of conditions or you can define a more complex rule using a custom Java class. For more information about using a custom Java class for rules, see [Chapter 7, "Creating Custom Java for Orchestrations"](#) in this guide.

An orchestration rule functions similar to an EnterpriseOne query in that each rule has:

- ❑ A "Match Any" or "Match All" setting.
- ❑ One or more conditions defined, each being a binary operation on two values.

After creating a rule and adding it to an orchestration, you use the Action column in the Orchestration design page to determine the orchestration step that is invoked when the conditions in the rule are met. See [Defining the Actions Between a Rule and Dependent Components](#) for more information.

D.5.2 Creating a Rule

Create a rule component to define the conditions for an orchestration.

1. Access the Rules design page:

- ❑ On the Orchestrator Home page, click the **Rules** icon. And then on the Rules design page, click the **New Rule** button.
- ❑ OR
- ❑ After adding a Rule step to the grid in the Orchestration design page, click the **Edit** button next to the step.

The Orchestrator Studio displays the Rule design page.

2. In the Rule field, enter a name for the rule.

Do not include any spaces in the rule name.

As an alternative, you can create a new rule by copying an existing rule. To do so:

- a.** Access the Rules design page from the Orchestrator Studio Home page.
 - b.** On the Rules design page, select an existing rule from the list of rules and then click the **Copy Rule** button.
 - c.** In the pop-up window, enter a name for the rule in the New Rule field.
 - d.** Click **Continue**.
- 3.** Click the **Edit Description** (pencil icon) button to enter a description for the rule, and then click **OK**.
- 4.** Select the Match Value drop-down menu and select one of the following values:
- ❑ **Match All.** Select this option if all conditions in the rule must be met.

- ❑ **Match Any.** Select this option if any conditions in the rule can be met.
5. In the first row in the grid, complete the following columns to add a condition:
 - ❑ **Rule Type.** Click the drop-down menu and select String, Numeric, or Date depending on the format of the input.
 - ❑ **Value 1.** Enter a variable for the input name.
 - ❑ **Operator.** In the drop-down menu, select from the list of operands which include: >, <, >=, <=, =, !=, startsWith, endWith, contains, between, inList.
 - ❑ **Literal.** Click this check box to indicate a literal value.
 - ❑ **Value 2.** If you selected the Literal check box, manually enter a value.
 - ❑ **Literal Value Type.** If you selected the Literal check box, click the drop-down menu and select the format of the literal value: string, numeric, data.
 6. Add additional conditions to the rule as needed. If you add a condition by mistake, you can delete it by clicking the **Remove** button at the end of the row with the condition.
 7. Click the **Save** button.

After adding a rule and a service request to an orchestration, you must define the actions for the rule and service request in the Orchestration design page. See [Defining the Actions Between a Rule and Dependent Components](#) for more information.

D.5.3 Adding a Custom Java Rule

You can create custom Java for a rule and then add it to an orchestration as a rule, which is referred to as a custom Java rule. Before you can add a custom Java rule to an orchestration, you must create the custom Java as described in [Chapter 7, "Creating Custom Java for Orchestrations"](#) in this guide.

To add a new custom Java rule:

1. Access the Rules design page from the Orchestrator Studio Home page.
2. On the Rules design page, click the **New Custom Java** button.
3. In the Rule field, enter a name for the custom Java rule.
Do NOT include spaces in the custom Java rule name.
4. In the Class field, enter the path to the Java class, which includes the name of the Java class.

This is the custom Java class that you created to use for the rule.

5. Complete the following fields in the grid:
 - ❑ **Attribute.** Enter the name of the field in the class.
 - ❑ **Input Value.** Enter a variable for the input name. If the attribute is a boolean and you pass in "true", then you could enter a hard-coded default value.
 - ❑ **Default Value.** Enter a hard-coded value if there is no mapped value. You can also add a hard-coded value to use if the mapped value returns a null.
6. If you make a mistake while configuring any of the fields, you can click the **Restore Rule** button which refreshes the custom Java rule to its last saved state.
7. Click the **Save** button.

D.6 Creating Cross References

This section contains the following topics:

- ¤ [Section D.6.1, "Understanding Cross References"](#)
- ¤ [Section D.6.2, "Creating a Cross Reference"](#)

D.6.1 Understanding Cross References

The Orchestrator uses a cross reference component to map incoming data (third-party data) to an EnterpriseOne value. The EnterpriseOne value identified in the cross reference is considered the output of the cross reference. The output from the cross reference becomes the data passed to another orchestration step.

For each cross reference that you create in the Orchestrator Studio, you have to create a cross reference record in P952000 in EnterpriseOne. When defining cross reference records for orchestrations in P952000, you must use the "AIS" cross reference type. For more information on how to create these cross reference records in P952000, see [Chapter 5, "Setting Up Cross References and White Lists in EnterpriseOne \(P952000\)"](#) in this guide.

Note: If a cross reference lookup fails in an orchestration, the orchestration is terminated.

D.6.2 Creating a Cross Reference

You must define the orchestration inputs before you can create a cross reference. See [Adding Inputs to an Orchestration](#) for more information.

To create a cross reference:

1. Access the Cross Reference design page:

- ¤ On the Orchestrator Home page, click the **Cross References** icon. And then on the Cross References design page, click the **New Cross Reference** button.
OR
- ¤ After adding a Cross Reference step to the grid in the Orchestration design page, click the **Edit** button next to the step.

The Orchestrator Studio displays the Cross Reference design page.

2. In the Cross Reference field, enter a name for the cross reference.

Do NOT include spaces in the cross reference name.

3. In the Object Type field, enter a name for the object type.

This is the object type used to categorize the orchestration cross references in EnterpriseOne. See "Adding Orchestration Cross References" in the *JD Edwards EnterpriseOne Tools Interoperability Guide* for more information.

4. Complete the Input Key and Output Key columns to map the inputs to EnterpriseOne fields:

- ¤ **Input Key.** The inputs can be one or many values. The inputs must correspond to the value or pipe (|) delimited values in the Third Party Value column in P952000.
- ¤ **Output Key.** The outputs can be one or many values. The outputs must correspond to the value or pipe (|) delimited values in the EOne Value column in P952000.

5. If you enter any values by mistake, you can click the **X** button next to the field to delete the entry.
6. Click the **Save** button.

The Orchestrator Studio saves the new cross reference component.

The output values in the cross reference are now available for mapping in subsequent orchestration steps.

You can also create a new cross reference by copying an existing cross reference. To do so:

1. Access the Cross References design page from the Orchestrator Studio Home page.
2. Select an existing cross reference from the list of cross references, and then click the **Copy Cross Reference** button.
3. In pop-up window, in the New Cross Reference field, enter a name for the new cross reference.
Do not include spaces when naming orchestration components.
4. Click **Continue** and then modify the cross reference as necessary.

D.7 Creating White Lists

This section contains the following topics:

- [Section D.7.1, "Understanding White Lists"](#)
- [Section D.7.2, "Creating a White List"](#)

D.7.1 Understanding White Lists

A white list contains a list of IDs permitted in the Orchestrator. By adding a white list to an orchestration, only inputs with IDs listed in the white list are permitted for processing by the Orchestrator. You add a white list component as a step in the orchestration.

In addition to specifying the permitted IDs in the white list, you must also specify the white list IDs in P952000 in EnterpriseOne. This is the same application that you use to set up and store orchestration cross references. As with cross references, use the "AIS" cross reference type in P952000 for a white list. In P952000, the Third Party App ID should have a value of WHITELIST. The EnterpriseOne value is not used for white lists and will default to NA.

If a white list lookup fails in an orchestration, the orchestration is terminated.

D.7.2 Creating a White List

1. Access the White List design page:
 - On the Orchestrator Home page, click the **White Lists** icon. And then on the White Lists design page, click the **New White List** button.
 - OR
 - After adding a White List step to the grid in the Orchestration design page, click the **Edit** button next to the step.

The Orchestrator Studio displays the White Lists design page.

2. In the White Lists design page, click the **New White List** button.
3. In the White List, enter a name for the white list.

Do NOT include spaces in the white list name.

4. In the Object Type field, enter a name for the object type.

The value you enter in the Object Type field must match a cross reference object type in the EnterpriseOne Business Services Cross Reference application (P952000).

The cross reference object type is a named group of records in P952000. For example, you may have thousands of records in P952000. You can use cross reference object types, for example "Equipment" or "Alert_Notification_Recipients" to group cross reference records into manageable categories. You define the cross reference object types as needed. See [Chapter 5, "Setting Up Cross References and White Lists in EnterpriseOne \(P952000\)"](#) for more information.

5. In the Input Key column, enter the input that you want to add as a permitted input.
6. Click the **Save** button.

You can also create a new white list by copying an existing white list. To do so:

1. After accessing the White Lists design page from the Orchestrator Studio Home page, select an existing white list from the list of available white lists.
2. Click the **Copy White List** button.
3. In the pop-up window, enter a name for white list in the **New White List** field.
Do NOT include spaces in the white list name.
4. Click the **Continue** button and then modify the white list as necessary.

D.8 Creating Orchestration

This section contains the following topics:

- [Section D.8.1, "Understanding Orchestration"](#)
- [Section D.8.2, "Creating an Orchestration"](#)
- [Section D.8.3, "Adding Inputs to an Orchestration"](#)
- [Section D.8.4, "Adding Steps to an Orchestration"](#)
- [Section D.8.5, "Configuring Transformations"](#)

D.8.1 Understanding Orchestration

Creating an orchestration in the Orchestrator Studio involves:

- Naming the orchestration and specifying the input format for the orchestration.
The input format can be JDE Standard, Oracle Cloud IoT, or Generic. See [Supported Input Message Formats](#) for more information about which input format to use.
- Adding inputs to the orchestration.

The inputs define the data passed from the device to the orchestration. This may include an ID that identifies the device as well as other data that the orchestration will receive from a device, such as temperature, date, or any other data you want to capture. Each input has a name and a type which can be string, numeric, or various date formats.

- Adding steps to the orchestration.

Each step is a component of the orchestration: a service request, rule, cross reference, or white list. At a minimum, an orchestration requires only a service request step, which provides the actions or the instructions to perform a particular task in EnterpriseOne.

D.8.2 Creating an Orchestration

To create an orchestration:

1. On the Orchestrator Studio Home page, click the **Orchestrations** icon.
2. On the Orchestrations page, click the **New Orchestration** button.
3. On the Orchestration design page, enter a unique name for the orchestration in the **Orchestration** field.
Do NOT use spaces when naming the orchestration.
4. Click the **Edit Description** (pencil icon) button to complete the Description field, and then click **OK**.
You can use this field to describe the purpose of the orchestration and any details that differentiate the orchestration from other orchestrations that you create.

5. Click the **Input Format** drop-down menu and select the appropriate format:
 - JDE Standard** (default)
 - Oracle Cloud IoT**
 - Generic**

See [Supported Input Message Formats](#) for more information about which input format to use.

6. At this point, you can click **Save All** before defining inputs or steps for the orchestration.

The Orchestrator Studio saves the orchestration. Next, add inputs to the orchestration as described in the [Adding Inputs to an Orchestration](#) section.

You can also create a new orchestration by copying an existing one and renaming it.

To create a new orchestration by copying another orchestration:

1. On the Orchestrations page, select an orchestration to copy from the list of orchestrations and then click the **Copy Orchestration** button.
2. In the New Orchestration field in the pop-up dialog box, enter a name for the orchestration.
3. Click **Continue**.

Caution: If you create a copy of an orchestration, only the orchestration is copied. The Orchestrator Studio does NOT create a copy of the components that are associated with the orchestration's steps. That is, both the original orchestration and the new orchestration use the same components that comprise the orchestration. Therefore, in the new orchestration, do NOT modify the components in any way that would break other orchestrations that use the same components.

D.8.3 Adding Inputs to an Orchestration

Orchestration inputs identify the data an orchestration consumes from external devices. In the orchestration, you enter names for the inputs. For example you can enter "SensorID" to pass a sensor ID value to an orchestration and you can enter "TemperatureReading" to pass a temperature value to the orchestration.

You also use these orchestration inputs to configure other components used by the orchestration, such as a service request, rule, white list, or cross reference. For example, if the orchestration requires a rule, you use the orchestration inputs to define the conditions for the

rule. If an incoming value from the device does not match an EnterpriseOne value, you can create a cross reference that uses the orchestration input to map third-party data to data in EnterpriseOne.

To add the orchestration inputs:

1. In the first empty row in the Orchestration grid, enter the name of the input in the Input column.
2. In the Value Type column, select the input value type. Valid values are:
 - String
 - Numeric

If the input is a date, you can use any of the following date formats:

- dd/MM/YYYY
- dd/MM/YY
- YYYY/MM/dd
- MM/dd/YYYY
- MM/dd/yy
- yy/MM/dd

You can also use the following date formats, which create additional inputs derived from the passed value:

- Milliseconds
- YYYY-MM-dd'T'HH:mm:ss.SSSZ

3. Click **Save All** to save your changes.

D.8.4 Adding Steps to an Orchestration

When you add a service request, rule, cross reference, or white list to an orchestration, you add it as a step in the orchestration. It is recommended that you create the component before adding it as a step in an orchestration.

The Orchestrator Studio provides "Insert Step Before" and "Insert Step After" buttons so you can add a step before or after another step in the orchestration. Therefore, you do not have to determine whether or not you need to add a particular step, such as a white list, to an orchestration before you add other steps.

Figure D–8 shows an orchestration that contains multiple rules, cross references, and service requests, which are displayed in the grid on the Orchestration design page:

Figure D–8 Steps in the AddConditionBased Alert Sample Orchestration in the Orchestrator Studio

Type	Action	Name	
Rule		JDE_RULE_Sample_VibrationAlarm Personal RUL_1608250018CUST	✓
Cross Reference	True	JDE_XREF_Sample_SensorLocation Pending Approval XRE_1608250006CUST	✓
Cross Reference	True	JDE_XREF_Sample_AlertNotificationRecipients Shared XRE_1608250008CUST	✓
Service Request	True	JDE_SREQ_Sample_AddCBArt_Alarm Personal SRE_1609270001CUST	✓
Rule	False	JDE_RULE_Sample_VibrationWarning Personal RUL_1608250017CUST	✓
Cross Reference	True	JDE_XREF_Sample_SensorLocation Pending Approval XRE_1608250008CUST	✓
Cross Reference	True	JDE_XREF_Sample_AlertNotificationRecipients Shared XRE_1608250008CUST	✓
Service Request	True	JDE_SREQ_Sample_AddCBArt_Warning Personal SRE_1609020001CUST	✓

In the grid, the Type column shows the type of step and the Name column shows the name of the component for the orchestration step. The "True" or "False" values in the Action column determine the actions the orchestration performs based on the conditions set in the rules. See [Defining the Actions Between a Rule and Dependent Components](#) for information about defining the actions in an orchestration.

To add steps to an orchestration:

1. To add the initial step to an orchestration, click the **Add Step** button (+ symbol).
2. In the "Enter Type of Step" pop-up field, select one of the following steps to add to the orchestration, and then click the **Ok** button.

- **Cross Reference**

- **Rule**

- **Service Request**

- **White List**

The Orchestrator Studio displays the first step in the grid.

3. Add a component to the step:
 - a. To use an existing component for the new step, click the drop-down menu in the Name column and select a component.
 - b. To create a new component for the step, click the **Edit** button (pencil icon) at the end of the row to access the design page for creating the new component. After creating

the component, when you return to the Orchestration design page, you will need to select the component from the drop-down menu in the orchestration steps grid to add it.

See the appropriate topics in this chapter on how to create the orchestration components.

4. In the Transformations area on the right side of the page, configure the transformations for each orchestration step as described in the [Configuring Transformations](#) section.
5. To add additional steps to an orchestration:
 - a. Select a step in the grid, and then click either the **Insert Step Before** or **Insert Step After** button to add an additional step before or after the selected step.
 - b. In the "Enter Type of Step" pop-up dialog box, select the step that you want to add.
 - c. Click the **Ok** button.

To remove a step from an orchestration:

Caution: Before you delete a step, make sure the step is not used by any other component in the orchestration.

1. Select the step that you want to remove.
2. Above the grid with the steps, click the **Remove Step** button (the X button).

If you make a mistake when updating an orchestration, click the **Restore All** button in the upper-right corner of the design page to restore the orchestration to its last saved version, which erases any changes made since the last save.

D.8.4.1 Defining the Actions Between a Rule and Dependent Components

The Orchestration design page contains an Action column for defining the actions between a Rule step and other orchestration steps in an orchestration. After you create a rule with conditions and add it as a step to an orchestration, you need to define the action the orchestration takes if the condition in the rule is met. For example, if a rule contains a condition that when met should invoke a Service Request step, then in the Service Request step row, you need to set the Action column to **True**.

You can also define a **False** action to invoke a different orchestration step when a condition in the initial rule is NOT met. A **False** action can invoke a different Service Request step or another Rule step that contains additional conditions for which the incoming data is evaluated against. Thus, you can design an orchestration with as many rules and service requests as your business requirements necessitate.

Figure D–9 shows an example of an orchestration with two Rule steps and two Service Request steps, with the following defined actions:

- ❑ For the first Service Request step, the action is set to **True** to instruct the orchestration to invoke this Service Request step when the condition in the first Rule step is met.
- ❑ The action for the second (nested) Rule step is set to **False** to instruct the orchestration to invoke this rule when the condition in the first rule is NOT met.
- ❑ The action for the second Service Request step is set to **True** to instruct the orchestration to invoke this service request when the condition in the second rule is met.

Figure D–9 Defining the Orchestration Actions

Type	Action	Name
CrossReference		JDE_XREF_Sample_SensorLocation
CrossReference		JDE_XREF_Sample_AlertNotificationRecipients
Rule		JDE_RULE_Sample_CBMAlar_1
ServiceRequest	True	JDE_SREQ_Sample_AddCBArt_Alarm
Rule	False	JDE_RULE_Sample_CBMAlar_2
ServiceRequest	True	JDE_SREQ_Sample_AddCBArt_Alarm

To set the Action column to True or False:

1. In the Orchestration design page, in the appropriate Rule or Service Request step row, click in the Action column.
2. Select either **True** or **False** as appropriate.
3. Click the **Save** button.

After completing the orchestration, you should use the Orchestrator Client to test the orchestration in a test environment. You can access the Orchestrator Client from the drop-down menu in the upper-right corner of the Orchestrator Studio. See [Chapter 9, "Testing Orchestrations in the EnterpriseOne Orchestrator Client"](#) for more information.

D.8.5 Configuring Transformations

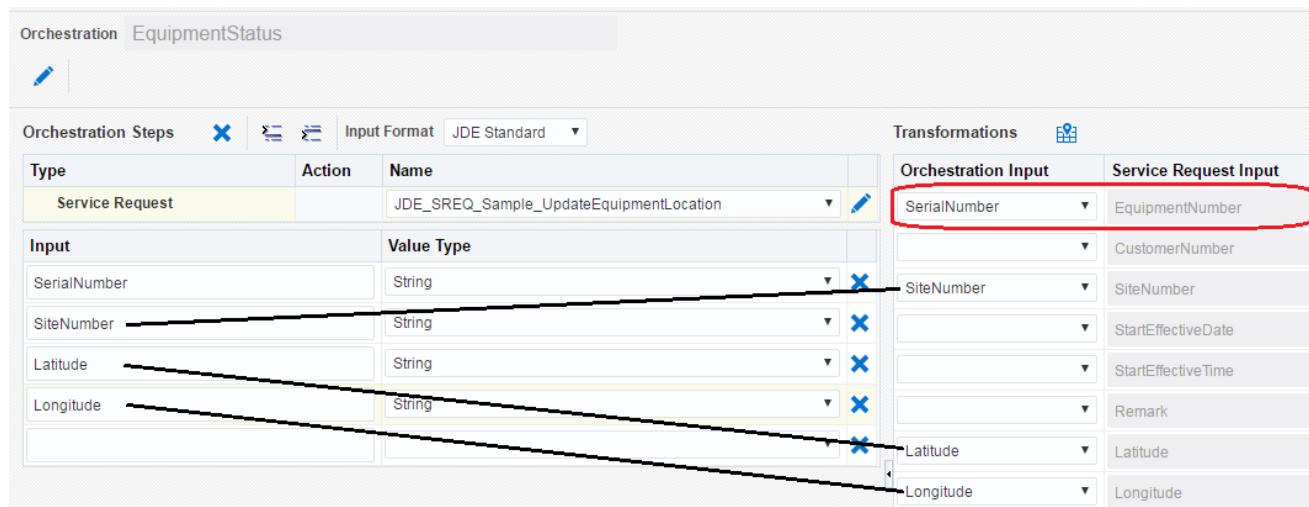
Use the Transformations area on the Orchestration design page to map orchestration inputs to inputs defined in an orchestration step, such as inputs in a rule, cross reference, white list, or service request component. The Transformations area contains an Auto Map button that you can use to automatically associate orchestration inputs to matching inputs defined in an orchestration step.

If an orchestration and an orchestration component use different input names to identify the same data, you can use the grid in the Transformations area to map the inputs. This facilitates the reuse of components, so you can avoid having to create additional orchestration components when an orchestration input name (input from a device) does not match the input name defined in an orchestration step.

To better understand how you use transformations, consider the following scenario which is depicted in [Figure D–10](#):

- You have an existing service request that creates alerts in the EnterpriseOne Condition-Based Alerts Revisions application (P1311). This service request includes an input called EquipmentNumber that is mapped to the Asset Number field in P1311.
- You want to reuse this service request as a step in a new orchestration, but the new orchestration will consume data from devices that supply the EquipmentNumber as "SerialNumber."
- To reuse the existing service request designed to consume data labeled as EquipmentNumber, in the new orchestration, you can create a transformation to pass the SerialNumber input to the EquipmentNumber input in the service request.
- Create the new orchestration with SerialNumber as one of the inputs, and then add the existing service request as a step.
- When you click the Service Request step in the orchestration steps grid, the Service Request column in the Transformations table displays all of the inputs defined in the service request. Notice in [Figure D–10](#) that EquipmentNumber, not SerialNumber, is defined as an input in the Service Request Input column.

- Next, you click the Transformations Auto Map button, which automatically maps the matching inputs. [Figure D–10](#) shows the SiteNumber, Latitude, and Longitude inputs in the Orchestration Input column, which were populated automatically after the Auto Map button was selected.
- Finally, to map the SerialNumber input in the orchestration to the EquipmentNumber input in the service request, you click the drop-down menu in the Orchestration Input column and select SerialNumber, as shown in [Figure D–10](#).

Figure D–10 Transformations Example

Initially, when you have a small number of orchestrations in your system, it is recommended to keep all input names consistent among the orchestration and all components (orchestration steps) used by orchestration. But as your library of orchestrations and orchestration components increases, instead of creating new components with input names that match the orchestration inputs, you can use transformations to map the orchestration inputs to an existing component.

To create transformations:

1. In the Orchestration design page, select an orchestration step for which you want to configure transformations.
2. Click the Transformations **Auto Map** button to map matching input names. Matching inputs automatically appear in the Orchestration Input column next to the matching input in the "<orchestration step> Input" column.
3. To map an orchestration input name to a non-matching input name in an orchestration step:
 - a. In the Transformations table, select the drop-down menu in the appropriate Orchestration Input row.
 - b. Select the orchestration input to map to the input name for the orchestration step.

For example, in [Figure D–10](#), SerialNumber is mapped to EquipmentNumber in the Service Request Input.

D.9 Reloading Orchestrations and Orchestration Components

The Reload Files feature enables you to reload orchestrations and orchestration components to the state in which they were last saved in the Orchestrator Studio. Use the Reload Files feature if you do not want to save any modifications that you made.

To reload all files, click the drop-down menu in the upper-right corner of the Orchestrator Studio and then click the **Reload Files** link.

Each individual orchestration component panel also contains a Restore button that you can use to restore an individual component.

D.10 Setting up Orchestration Security

Before the EnterpriseOne Orchestrator can process an orchestration, authentication of the JD Edwards EnterpriseOne user ID and password must take place. It is the responsibility of the originator of the service request to tell the orchestration about the user. The user's credentials must be supplied in a basic authorization header or in the JSON body of the request. The user must also have authorized access to the EnterpriseOne application in which the resulting transaction takes place. The following code is an example of credentials in the JSON body of the request:

```
{
    "username": "JDE",
    "password": "JDE",
    "environment": "JDV900",
    "role": "*ALL"
}
```

The AIS service used with orchestrations is stateless; each call passes credentials to establish a separate EnterpriseOne session. After the transaction is complete, the session closes.

In addition to passing credentials for authentication, you can employ a second level of security for the Orchestrator through whitelisting. Whitelisting enables an initial rudimentary pass/fail check of the incoming device signature against a predefined list of signatures. A white list provides an additional layer of security to the Orchestrator security. If a value passed to the Orchestrator is not a valid value included in the orchestration's white list, the Orchestrator rejects the input. For more information, see [Creating White Lists](#).

D.11 Exporting Files from the Orchestrator Studio

Each component design page in the Orchestrator Studio includes an Export File button for exporting an orchestration component. The Orchestrator Studio exports orchestration components as XML files in a zip file.

When you export an *orchestration*, the Orchestrator Studio gives you the option to either export only the orchestration component or the orchestration component and all components associated with the orchestration.

To export an orchestration:

1. On the design page for the component, click the **Export File** button in the upper-right corner.
If you are exporting an orchestration, a pop-up dialog box appears in which you can click **All** or **Orchestration Only**.
2. Follow the instructions in the browser to save the file to a location on your local machine.

D.12 Importing Orchestration Files in the Orchestrator Studio

All orchestrations and orchestration components are saved as XML files on the AIS Server. The Orchestrator Studio provides an Import Files tool for importing orchestration files into your

current environment, whether it is a production environment or test environment. You can use this feature along with the Export tool to import and export files between a test and production environment.

To import files:

1. On the Orchestrator Studio Home page, click the **Tools** link in the upper-right corner.
2. On the Tools page, click the **Import Files** icon.
3. On Import Files, click the **Choose File** button.
4. Browse and locate the orchestration files in your local file system.

After selecting one or more files to import, the Orchestrator Studio displays the files on the Import Files page.

5. Click the check box next to each file to confirm the file for submission, or you can click the **Select All** check box to select them all.
6. Click the **Submit** button.

Creating Orchestrations with the Orchestrator Studio (Release 9.2.0.2)

Important: This appendix describes how to use Orchestrator Studio 1.0, the first version that was available for download along with EnterpriseOne Tools 9.2.0.2.

To use Orchestrator Studio 7.0.x.0, the latest version available with EnterpriseOne Tools 9.2.3, see:

- [Chapter 2, "Implementing the Orchestrator Studio"](#) for installation instructions
- [Chapter 4, "Creating Orchestrations with Orchestrator Studio 7.x.x.x"](#)

This appendix contains the following topics:

- [Section E.1, "Understanding the Orchestrator Studio and Orchestrations"](#)
- [Section E.2, "Accessing the Orchestrator Studio"](#)
- [Section E.3, "Creating Orchestrations"](#)
- [Section E.4, "Creating Service Requests"](#)
- [Section E.5, "Creating Rules"](#)
- [Section E.6, "Creating Cross References"](#)
- [Section E.7, "Creating White Lists"](#)
- [Section E.8, "Reloading Orchestrations and Orchestration Components"](#)

E.1 Understanding the Orchestrator Studio and Orchestrations

The Orchestrator Studio is a web-based application for creating orchestrations and the components that comprise an orchestration. All orchestrations and orchestration components that you create in the Orchestrator Studio are saved as separate files in an orchestration directory on the AIS Server. The Orchestrator uses the following components to process a single orchestration instance on the AIS Server:

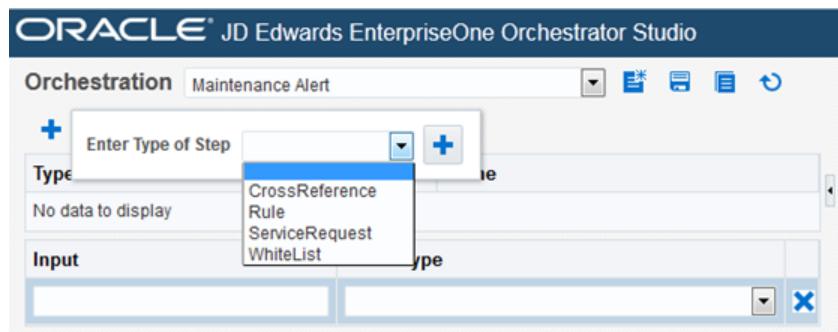
- **Orchestration.** The master process that provides a unique name for the orchestration process and invokes the other components in this list to enable the transfer of data from disparate devices to EnterpriseOne.
- **Service Request.** Contains a sequence of actions to perform a particular process in EnterpriseOne. The Orchestrator uses the service request as an invocation of a JD Edwards

EnterpriseOne interactive application or a Java application via a REST service call to the EnterpriseOne Application Interface Services (AIS) Server.

- ❑ **Rule.** A set of conditions against which the input from devices is evaluated to produce a true or false state. Rules can be nested to produce complex evaluations. With rules, a false outcome or true outcome can invoke further orchestration steps. You can also use custom Java to define rules.
- ❑ **Cross Reference.** A set of data relationships that map third-party values to EnterpriseOne values. For example, a device's serial number can be cross-referenced to a JD Edwards EnterpriseOne Equipment Number for use in service requests.
- ❑ **White List.** An inclusive list of values permitted in the orchestration and terminates the orchestration process if the data is not recognized.

In the Orchestrator Studio, you create and name the orchestration and then add each orchestration component as a "step" in the orchestration. [Figure E-1](#) shows the drop-down list of steps. Each step in an orchestration is simply a reference to a service request, rule, cross reference, or white list component.

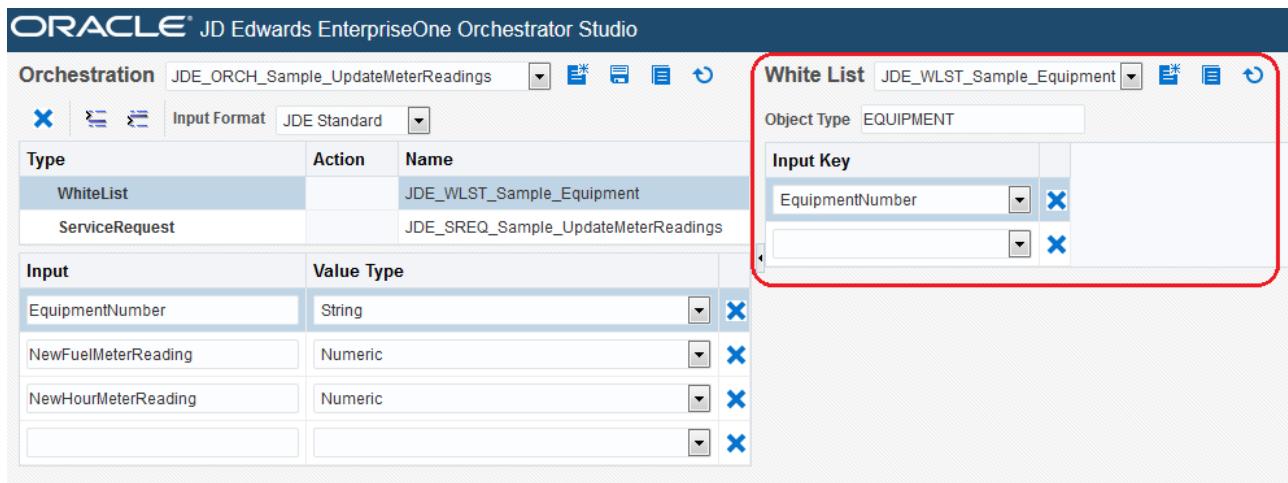
Figure E-1 *Orchestration Steps in the Orchestrator Studio*



When you add a step to an orchestration, the Orchestrator Studio displays a design panel in which you can create the component for that step. For example, when you add a Rule step to an orchestration, the Orchestrator Studio displays a design panel in which you can create the rule; when you add a ServiceRequest step to the orchestration, the Orchestrator Studio displays a design panel in which you can create the service request, and so forth.

Each area in the Orchestrator Studio contains a raised tab with directional arrows for collapsing or expanding the component area. This enables you to utilize the space in the Orchestrator Studio when editing any particular component.

[Figure E-2](#) shows the design panel for creating a white list.

Figure E–2 Creating a White List in the Orchestrator Studio

E.1.1 Reusing or Copying Orchestration Components

Orchestration components are reusable. A component such as a white list or cross reference created for one orchestration can be used in other orchestrations. When you add a step to an orchestration, the Orchestrator Studio gives you the option to create a new component for the step or select from a list of existing components.

If you reuse a component, you must make sure to not modify the component in any way that would break the functionality of other orchestrations that use the same component.

Important: Because you can use the same components in multiple orchestrations, Oracle highly recommends that you keep a record of all orchestration components and where they are used.

The Orchestrator Studio also provides a copy feature that enables you to create a new component from a copy of an existing component. The existing component can serve as a template for creating the new component. Because you are using a copy, you do not have to worry about breaking existing orchestrations where the original component is used.

E.2 Accessing the Orchestrator Studio

You access the Orchestrator Studio by entering a URL in a web browser. Ask your system administrator for the URL.

Important: Because the Orchestrator Studio saves orchestrations to a directory on the AIS Server, an administrator must define the path to the AIS Server before you access the Orchestrator Studio. For identifying the URL to the Orchestrator Studio and setting the path to the AIS Server, refer to the [Setting Up the Orchestrator Studio](#) section in this guide.

To access the Orchestrator Studio:

1. In a Web browser, enter the URL to access the Orchestrator Studio:

`http://<adf_server>:<port>/OrchestratorStudio/faces/index.jsf`

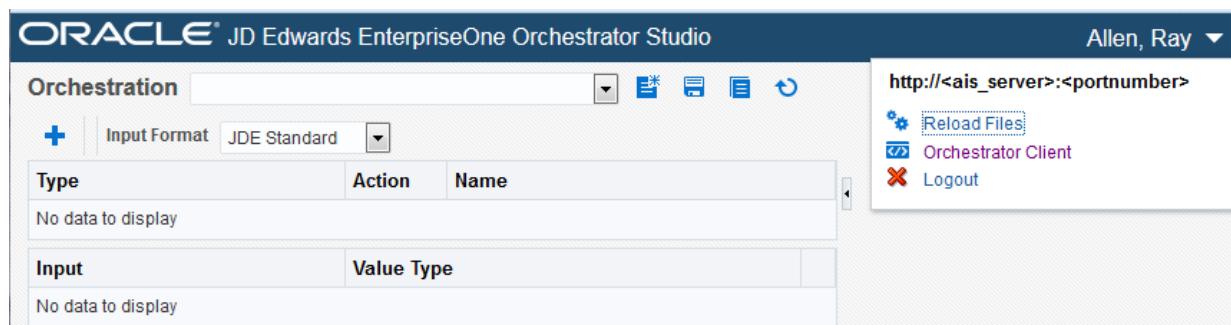
2. On the Orchestrator Studio Sign In screen, enter your EnterpriseOne User credentials, environment, and role.

It is highly recommended that you enter an EnterpriseOne environment used for testing, not a production environment.

3. Click the **Login** button.

Before you begin using the Orchestrator Studio, click the drop-down menu in the upper-right corner to see the path to the AIS Server directory where orchestrations created in the Studio are saved. [Figure E-3](#) shows the drop-down menu, which in addition to the path to the AIS Server directory, contains a Reload Files button, a Logout button, and a link to the Orchestrator Client, a web-based application for testing the orchestrations that you create in the Orchestrator Studio. See [Chapter 9, "Testing Orchestrations in the EnterpriseOne Orchestrator Client"](#) for more information.

Figure E-3 Orchestrator Studio Drop-down Menu



E.3 Creating Orchestrations

This section contains the following topics:

- ❑ [Section E.3.1, "Understanding Orchestrations"](#)
- ❑ [Section E.3.2, "Creating an Orchestration"](#)
- ❑ [Section E.3.3, "Adding Inputs to an Orchestration"](#)
- ❑ [Section E.3.4, "Adding Steps to an Orchestration"](#)
- ❑ [Section E.3.5, "Completing an Orchestration"](#)

E.3.1 Understanding Orchestrations

Creating an orchestration in the Orchestrator Studio involves:

- ❑ Naming the orchestration and specifying the input format for the orchestration.

The name of an orchestration cannot include spaces. The input format can be JDE Standard, Oracle Cloud IoT, or Generic. See [Supported Input Message Formats](#) for more information about which input format to use.

- ❑ Adding inputs to the orchestration.

The inputs define the data passed from the device to the orchestration. This may include an ID which identifies the device as well as other data that the orchestration will receive from a device, such as temperature, date, or any other data you want to capture. Each input has a name and a type which can be string, numeric, or various date formats.

- ❑ Adding steps to the orchestration.

Each step is a component of the orchestration: a service request, rule, cross reference, or white list.

E.3.2 Creating an Orchestration

To create an orchestration:

1. Click the **New Orchestration** button next to the Orchestration field.
2. In the Orchestration pop-up field, enter a unique name for the orchestration.
Do NOT use spaces when naming the orchestration.
3. Click the **Ok** (plus symbol) button.
4. Click the **Input Format** drop-down menu and select the appropriate format:
 - JDE Standard**. (default)
 - Oracle Cloud IoT**.
 - Generic**.

See [Supported Input Message Formats](#) for more information about which input format to use.

5. At this point, you can click **Save All** before defining inputs or steps for the orchestration.
The Orchestrator Studio saves the orchestration. Next, add inputs to the orchestration as described in the [Adding Inputs to an Orchestration](#) section.

You can also create a new orchestration by copying an existing one and renaming it.

To create a new orchestration by copying another orchestration:

1. Click the **Copy** button.
2. In the pop-up, select an orchestration from the "Orchestration to Copy" drop-down menu.
3. In the New Orchestration field, enter a name for the new orchestration.
4. Click **Save**.

Caution: If you make a copy of an orchestration, only the orchestration is copied. The Orchestrator Studio does NOT make a copy of the components that are associated with the orchestration's steps. That is, both the original orchestration and the new orchestration use the same components that comprise the orchestration. Therefore, in the new orchestration, do NOT modify the components in any way that would break other orchestrations that use the same components.

E.3.3 Adding Inputs to an Orchestration

Orchestration inputs identify the data the orchestration consumes from external devices. In the orchestration, you enter the names of the inputs. For example you can enter "SensorID" to pass a sensor ID value to an orchestration and you can enter "TemperatureReading" to pass a temperature value to the orchestration.

Note: If a device input does not match an EnterpriseOne value, such as a column or field name, you can use a cross reference to map the input to the appropriate EnterpriseOne value. See [Creating Cross References](#) for more information.

If the orchestration requires a rule, you use the inputs defined in the orchestration to configure the rule. See [Creating Rules](#) for more information.

You enter inputs directly in the Input and Value Type columns in the grid in the orchestration design panel.

To add the orchestration inputs:

1. In the first empty row, enter the name of the input in the Input column.

Note: When you click in this column, the "Drag to Mapped Value" hover message appears. This message applies to when you use the inputs to configure a service request, rule, or cross reference as described later in this chapter.

2. In the Value Type column, select the input value type. Valid values are:

- ❑ String
- ❑ Numeric

Or you can use any of the following date formats:

- ❑ dd/MM/yyyy
- ❑ dd/MM/yy
- ❑ yyyy/MM/dd
- ❑ MM/dd/yyyy
- ❑ MM/dd/yy
- ❑ yy/MM/dd

You can also use the following date formats, which create additional inputs derived from the passed value:

- ❑ Milliseconds
- ❑ yyyy-MM-dd'T'HH:mm:ss.SSSZ

3. Click **Save All** to save your changes.

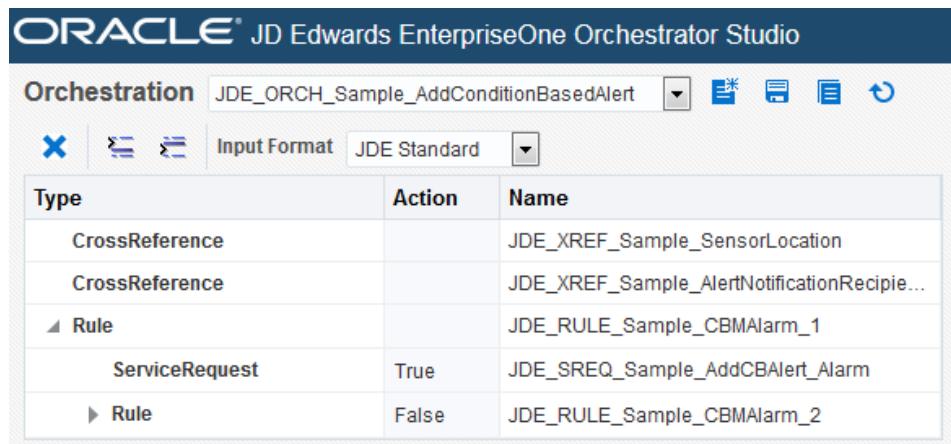
E.3.4 Adding Steps to an Orchestration

At a minimum, an orchestration requires only a service request step to run. You can add a rule step to provide true or false conditions that control the orchestration flow, such as whether the incoming data meets the condition to invoke the service request. You can add a cross reference step if an orchestration requires a cross reference to map data from the device to EnterpriseOne data. You can also add a white list step that contains a list of data permitted in the orchestration to provide an initial pass/fail check for the orchestration.

The Orchestrator Studio provides "Insert Step Before" and "Insert Step After" buttons so you can add a step before or after another step in the orchestration. Thus, you do not have to determine whether or not you need to add a particular step, such as a white list, to an orchestration before you add other steps.

Figure E-4 shows the steps in the AddConditionBasedAlert sample orchestration in the Orchestrator Studio which contains two cross references, two rules, and a service request:

Figure E–4 Steps in the AddConditionBased Alert Sample Orchestration in the Orchestrator Studio



The preceding figure shows how the Orchestrator Studio displays the steps. The Type and Name columns in the Orchestration grid area show the type of step and the name that you give each step or component. The "True" or "False" values in the Action column are used after configuring the rule and service request components as described in the [Creating an Orchestration](#) section.

To add steps to an orchestration:

1. To add the initial step to an orchestration, click the **Add Step** button (+ icon).
2. In the "Enter Type of Step" pop-up field, select one of the following steps to add to the orchestration, and then click the **Ok** button.
 - **CrossReference**
 - **Rule**
 - **ServiceRequest**
 - **Whitelist**

The Orchestrator Studio displays the first step in the grid. After you add a step, if you click the step in the grid, the Orchestrator Studio displays the design panel in which you can create the component for that step. See the other sections in this chapter for instructions on how to configure each component.

3. To add additional steps to an orchestration:
 - a. Select a step in the grid, and then click either the **Insert Step Before** or **Insert Step After** button to add an additional step before or after the selected step.
 - b. In the "Enter Type of Step" pop-up, select the step that you want to add.
 - c. Click the **Ok** button.

To remove a step from an orchestration:

Caution: Before you delete a step, make sure the step is not used by any other component in the orchestration.

1. Select the step that you want to remove.
2. Above the grid with the steps, click the Remove Step button (the X button).

If you make a mistake when updating an orchestration, you can restore the orchestration to its last saved version which erases any changes made since the last save.

To restore an orchestration, click the **Restore All** button, which is the last button to the right of the other Orchestration buttons.

E.3.5 Completing an Orchestration

The Orchestration design panel contains an "Action" column for defining the relationship between a rule and service request in an orchestration. After you add a rule with conditions for invoking a service request, you need to define the action the orchestration takes if the condition in the rule is met. For example, if a rule contains a condition that when met should invoke a service request, then you need to set the "Action" to True in the Orchestration design panel to direct the orchestration to invoke the service request. You can also define a False action when a condition in the initial rule is NOT met, to direct the orchestration to another rule in the orchestration. If the condition in the second rule is met, then you select a True action to direct the orchestration to invoke another service request, and so forth.

[Figure E-5](#) shows an example of an orchestration with two rules and two service requests, with the following defined actions:

- For the first ServiceRequest, the action is set to **True** to instruct the orchestration to invoke this service request when the condition in the first Rule is met.
- The action for the second (nested) Rule is set to **False** to instruct the orchestration to invoke this rule when the condition in the first Rule is NOT met.
- The action for the second ServiceRequest step is set to **True** to instruct the orchestration to invoke this service request when the condition in the second Rule is met.

Figure E-5 Defining Orchestration Actions

Type	Action	Name
CrossReference		JDE_XREF_Sample_SensorLocation
CrossReference		JDE_XREF_Sample_AlertNotificationRecipients
Rule		JDE_RULE_Sample_CBMAlarm_1
ServiceRequest	True	JDE_SREQ_Sample_AddCBArt_Alarm
Rule	False	JDE_RULE_Sample_CBMAlarm_2
ServiceRequest	True	JDE_SREQ_Sample_AddCBArt_Alarm

To set the Action column to True or False:

1. In the Orchestration design panel, in the appropriate Rule or ServiceRequest step row, click in the Action column.
2. Select either **True** or **False** as appropriate.
3. Click the **Save** button.

After completing the orchestration, Oracle recommends that you use the Orchestrator Client to test the orchestration in a test environment. You can access the Orchestrator Client from the drop-down menu in the upper-right corner of the Orchestrator Studio. See Chapter 9, "Testing Orchestrations in the EnterpriseOne Orchestrator Client".

E.4 Creating Service Requests

This section contains the following topics:

- Section E.4.1, "Understanding Service Requests"
- Section E.4.2, "Understanding the Service Request Design Panel"
- Section E.4.3, "Creating a Service Request"
- Section E.4.4, "Configuring the Actions in the Service Request"
- Section E.4.5, "Defining the Order of Execution in the Service Request"
- Section E.4.6, "Adding an Existing Service Request to an Orchestration"
- Section E.4.7, "Adding a Custom Java Service Request"

E.4.1 Understanding Service Requests

At a minimum, an orchestration requires orchestration inputs and a single service request component to run. The service request provides the instructions that the Orchestrator uses to invoke and perform a particular task in EnterpriseOne. Before you create a service request, you must first identify the EnterpriseOne task or business process that you want the service request to perform. See [Identifying the Service Request Information for the Orchestration](#) section in this guide for more information.

Use the Service Request design panel in the Orchestrator Studio to define a service request, which involves:

- Specifying the EnterpriseOne buttons, fields, columns, menu items, and so forth that are involved in performing a task in EnterpriseOne.
- In the Service Request design panel, you can search on an EnterpriseOne application form and view all of the controls and fields for a form.
- Mapping the orchestration inputs defined in the orchestration to the appropriate fields in EnterpriseOne.
 - Determining the order or execution in which the controls and fields are used to perform the desired task in EnterpriseOne.

Note: You can also use custom Java to execute a custom process or to route data into another database. See [Chapter 7, "Creating Custom Java for Orchestrations"](#).

E.4.1.1 Understanding the Application Stack Option

The Application Stack option in the Service Request design panel enables you to create a service request that establishes a session for a specific application and maintains that session across calls. With application stack processing, the service request can invoke multiple forms in different applications to complete a business process. Without the application stack processing, each form in the service request is opened independently.

If you use the Application Stack option, the service request must include instructions for navigating between forms. Without the navigation, additional forms will not be called.

E.4.2 Understanding the Service Request Design Panel

The Service Request design panel contains two areas: the "Order of Execution" area and the "Available Actions" area. The "Order of Execution" area is where you define the sequence of actions to perform the desired task in EnterpriseOne. The "Available Actions" area serves as the workspace for finding and configuring all of the EnterpriseOne controls and fields that make up the sequence of actions in the "Order of Execution" area. You locate and configure the controls

and fields in the "Available Actions" area, and then move each item to the "Order of Execution" and place them in the proper sequence.

Available Actions Area

In the Available Actions area, you can search on an EnterpriseOne form to load all of the available controls (menu items, buttons, check boxes, and so forth) and fields for that form. If the service request needs to invoke more than one application to complete the task, you can load the controls and fields of additional application forms as needed. You can select the Application Stack check box to enable application stack processing as described in the [Understanding the Application Stack Option](#) section.

Figure E–6 shows the results of loading an EnterpriseOne application form in the Available Actions grid.

Figure E–6 Available Actions Area in the Service Request Design Panel

Available Actions		Application Stack <input checked="" type="checkbox"/>		Run Synchronously <input checked="" type="checkbox"/>		Bypass Form Processing <input type="checkbox"/>	
Application	P1311	Form	W1311B - Condition-Based Alerts Revisions	Version			Form Mode
Description		Mapped Value	Default Value	ID	Version	Form Mode	Return
Condition-Based Alerts Revisions				P1311_W1311B			
Buttons and Exits							
Cancel				12			
Create W.O.				114			
Failure Analysis				124			
Investigation				120			
Investigation Msg				115			
Notification				119			

The columns in the Available Actions grid provide interactive features for configuring the controls and fields before you add them to the "Order of Execution" area. The following list describes how to use each column:

- **Description.**

This column displays the controls and fields for each form in a collapsible/expandable parent node named after the EnterpriseOne form. Child nodes categorize other items, and include a Buttons and Exits node, a node for displaying the available Query by Example (QBE) fields in a grid (if applicable), and a node for displaying all of the available columns in a grid.

Note: The separate nodes for QBE fields and Grid columns are available with the IoT_Orchestrator_Components_2.0.1 download. See [Downloading and Installing Orchestrator Studio](#) in this guide.

- **Mapped Value.**

Use this column to map orchestration inputs to an EnterpriseOne field. You can only map inputs to a field if 1) the EnterpriseOne field is an editable field and 2) you defined the inputs in the orchestration as described in [Adding Inputs to an Orchestration](#).

Default Value.

Enter a default, hard-coded value if there is no mapped value. You can also add a hard-coded value to use if the mapped value returns a null.

"Text substitution" check box column.

This check box enables you to add text substitution for an input. Text substitution enables you to combine input values into a text string for input into an EnterpriseOne field.

ID.

This column displays the ID of the control or field in EnterpriseOne. This is for informational purposes only.

Return.

You can specify any values that you want returned in the orchestration response after EnterpriseOne actions are invoked. If you do not specify any return values, all values on the form will be returned. Return values may be used by customized third-party gateway programs or devices to do further processing based on the values returned from EnterpriseOne applications.

Additional actions available in the grid are "Select First Row" and "Row Number for Update." You can use "Select First Row" if the service request tasks involves selecting the first row in the grid. If the service request task requires updating an existing row in an input capable grid, you can use the "Row Number for Update" row to map an orchestration input to a particular row in the grid. "Row Number for Update" is available with the IoT_Orchestrator_Components_2.0.1 download. See [Downloading and Installing Orchestrator Studio](#) in this guide.

After you configure each applicable control or field in the Available Actions area, you click the **Add Action** button in the last column to add each item as an action in the Order of Execution area at the top of the design panel.

Order of Execution Area

When actions are added to the Order of Execution grid, you can reorder them using the up and down arrow buttons to the right of each row. You can also delete any actions using the Delete (X) button. Figure E-7 shows the "Order of Execution" area.

Figure E-7 Order of Execution in the Service Request Design Panel

Order of Execution				
Description	Action	Mapped Value	Default Value	
Odometer	SetCheckboxValue	on		▲ ▼ X
Fuel Meter	SetCheckboxValue	on		▲ ▼ X
Hour Meter	SetCheckboxValue	on		▲ ▼ X
	SetCheckboxValue	on		▲ ▼ X
Subledge	SetCheckboxValue	on		▲ ▼ X
	SetCheckboxValue	on		▲ ▼ X

E.4.3 Creating a Service Request

To create a service request for an orchestration:

1. After adding a ServiceRequest step to an orchestration, click the **ServiceRequest** step row. The Orchestrator Studio displays the Service Request design panel.
2. In the Service Request design panel, click the **New Service Request** button.
3. In the Service Request pop-up field, enter a name for the service request and then click the **Ok** button (plus symbol).

Do NOT include any spaces in the service request name.

As an alternative, you can create a new service request by copying an existing service request. To do so:

- a. Click the **Copy Service Request** button.
 - b. In the pop-up window, select a service request from the "Service Request to Copy" drop-down menu.
 - c. In the New Service Request field, enter a name for the service request.
4. Click the **Save All** button to save the orchestration and the new service request component.

E.4.4 Configuring the Actions in the Service Request

In the Service Request design panel, use the Available Actions area to define actions that you want the service request to perform. After you define the actions, you can place the actions in the proper order in the Order of Execution area as described in [Defining the Order of Execution in the Service Request](#).

To define the actions for the service request:

1. In the Available Actions area, complete the following fields to specify the EnterpriseOne form that contains the controls you want to load:
 - **Application.** Enter the ID of the EnterpriseOne application.
 - **Form.** Enter the form ID.
 - **Version.** Enter the version of the EnterpriseOne application.
 - **Form Mode.** (Available with Release 9.2.0.3) Click the drop-down menu and select the appropriate form mode: **Add**, **Update**, or **Inquiry**.

The form mode determines the controls that are displayed for the form. At runtime, the controls that appear on an EnterpriseOne form are dependent on the form mode as specified in Form Design Aid (FDA). This ensures that you can see all of the form controls in the Orchestrator Studio that are available in the form at runtime.

- **Application Stack.** Click this check box if you want to use application stack processing. See [Understanding the Application Stack Option](#) for more information.
2. Click the **Load Form** button.

The Orchestrator Studio loads the controls and fields for the form in the grid.

If the action involves more than one form, you can search on another form and the Orchestrator Studio will load the controls for the form in a collapsible node at the end of the grid.

3. To map the inputs to fields:
 - a. Select the row with the field to be mapped.

- b.** Click the drop-down menu in the Mapped Value column and select the input to map to the field. You can also click and drag inputs defined in the Orchestration design panel to the field in the Mapped Value column.

If you have not already done so, you must first add orchestration inputs, or if necessary add cross references for the inputs, before you can map them. See [Adding Inputs to an Orchestration](#) for more information.

- 4.** If the service request task involves selecting the first grid row in the EnterpriseOne form, in the "Select First Row" row, click the **Add Action** button to move it to the "Order of Execution" area.
- 5.** If the service request task requires updating a particular row in an input capable grid, then map the appropriate input value containing the row number to the "Row Number for Update" row, or specify the row number in the Default Value column.
- 6.** In the Default Value column, you can perform the following actions to configure the default values for the controls:
 - a.** If the control is a check box or a radio button, you can select it to include it. If there are multiple radio buttons on a form, select only one. If you select more than one, only the last radio button in the Order of Execution list will be used.
 - b.** If the control is a combo box (a list of items in a drop-down menu), then the Studio displays a drop-down menu in the Default Value column in which you can select the appropriate item from the list.
 - c.** If the field is editable, you can enter a hard-coded value to map for the input. You can also enter a value to be used if the mapped value returns a null.

Fields that are grayed out in this column are not editable.

- 7.** Select the **Return** check box next to the fields that contain values that you want returned in the orchestration response. If you do not specify any return values, all values on the form will be returned.

Important: If using the Application Stack option:

- Only values from the last form in the application stack can be returned; return controls specified for any form except the last form are ignored.
 - If you use a form to add or update a value that upon saving, exits to another form (such as a Find/Browse form), the return values will come from the last form that appears after the save. In this scenario, you need to add the last form to the application stack and select the return controls from the last form.
- 8.** To use text substitution, in the field that you want to substitute the input with text, click the **Add Text Substitution** button (plus symbol) and perform the following steps in the pop-up window:

Text substitution enables you to combine input values into a text string for input into an EnterpriseOne field. For example, you might have a string with "Temp was {0} at {1}" that contains a temperature input value and a time input value.

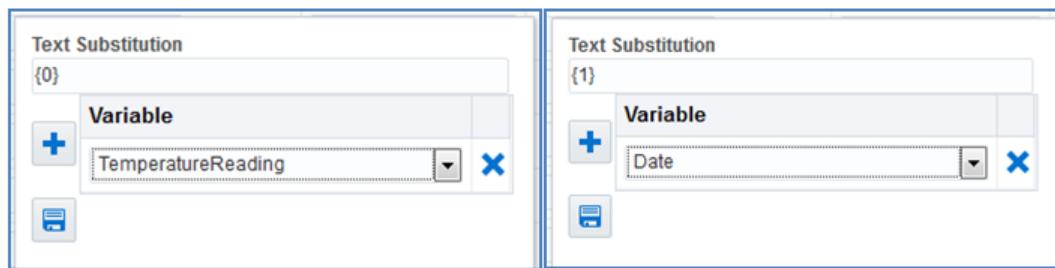
- a.** In the Text Substitution field, enter the following variable to use for the first input:
`{0}`
- b.** Below the variable, select the drop-down menu to select an input for the variable.
- c.** Click the **Add** button (plus symbol) to add the variable for the text substitution.

- d. Add the next variable, {1}, and then select the input for this variable, and click the **Add** button to add it to the text substitution.

If you add a variable by mistake, click the **Remove Variable** button.

- e. Click the **OK** button to save the variables for the text substitution.

The following image shows an example of variables added for a temperature input value and a date input value:



9. As you finish configuring the applicable rows in the Available Actions area, in the right-most column, click the **Add Action** button to move the configured rows to the "Order of Execution" area.

Next, order the actions as appropriate as described in [Defining the Order of Execution in the Service Request](#).

E.4.5 Defining the Order of Execution in the Service Request

After you move the actions from the "Available Actions" area to the "Order of Execution" area, you arrange the actions in the order required to perform the task in EnterpriseOne.

To define the order of execution:

1. Click the action that you want to move up or down in the order.
2. Click the "up arrow" or "down arrow" buttons at the end of the row with the action until the action is in the proper order.
3. Continue until the order of the actions reflects how the task is performed in EnterpriseOne.
4. If you added an action that is not needed, click the **Delete** (X) button at the end of the row to delete the action.
5. Click the **Save All** button next to the Orchestration drop-down menu to save the orchestration with the service request.
6. If you make a mistake when modifying a service request, click the **Restore Service Request** button to restore the service request to its last saved version.

If you add a rule component with conditions for invoking the service request, after creating the service request and rule, you must define the actions for the rule and service request in the orchestration design panel. See [Completing an Orchestration](#) in this guide for more information.

E.4.6 Adding an Existing Service Request to an Orchestration

If an existing service request provides the tasks that you want to use in a new orchestration, you can reuse the service request in the new orchestration.

However, make sure that you do not modify the service request in any way that would break other orchestrations where the service request is used.

To add an existing service request to an orchestration:

1. After adding the ServiceRequest step to the orchestration, click the **ServiceRequest** row.
2. In the Service Request drop-down menu, select an existing service request.
3. Click the **Save All** button.

The Orchestrator Studio saves the service request component and saves the orchestration with the service request as the orchestration step.

E.4.7 Adding a Custom Java Service Request

You can create custom Java for a service request, which is referred to as a custom Java service request. Before you can add a custom Java service request to an orchestration, you must create the custom Java as described in [Chapter 7, "Creating Custom Java for Orchestrations"](#).

To add a custom Java service request, you can add a new custom Java service request or copy an existing custom Java service request.

To add a new custom Java service request:

1. After adding the ServiceRequest step to the orchestration, click the **ServiceRequest** step row.
2. In the Service Request area, click the **New Custom Java** button.
3. In the Custom Java pop-up field, enter a name for the custom Java service request.
Do NOT include spaces in the name.
4. Click the **Ok** button.
5. In the Class field, enter the Java class.

This is the custom Java class that you created to use for the service request. See [Chapter 7, "Creating Custom Java for Orchestrations"](#).

6. Complete the following fields:
 - **Attribute.** Enter the name of the field in the class.
 - **Input Value.** Click the drop-down menu and select the input value. If the attribute is a boolean and you pass in "true", then you could use a default value.
 - **Default Value.** Enter a default, hard-coded value if there is no mapped value. You can also add a hard-coded value to use if the mapped value returns a null.
7. Click the **Save** button.

The Orchestrator Studio saves the custom Java request and saves the orchestration with the newly added ServiceRequest step.

8. If you modify any fields and make a mistake, you can click the **Restore Custom Java** button which refreshes the custom Java rule to its last saved state.

To add a new custom Java service request by copying an existing custom Java service request:

1. Click the **Copy Java** button.
2. In the pop-up fields, click the **Custom Java to Copy** drop-down menu and select a custom Java service request to copy.
3. In the New field, enter a name for the new custom Java service request.
4. Modify the custom Java service request values as necessary.
5. Click the **Save** button.

E.5 Creating Rules

This section contains the following topics:

- [Section E.5.1, "Understanding Rules"](#)
- [Section E.5.2, "Creating a Rule"](#)
- [Section E.5.3, "Adding an Existing Rule to an Orchestration"](#)
- [Section E.5.4, "Adding a Custom Java Rule"](#)

E.5.1 Understanding Rules

A rule contains conditional logic that the Orchestrator uses to evaluate conditions, such as true or false conditions that determine how the orchestration processes the incoming data. You can define a rule with a list of conditions or you can define a more complex rule using a custom Java class. For more information about using a custom Java class for rules, see [Chapter 7, "Creating Custom Java for Orchestrations"](#).

An orchestration rule functions similar to an EnterpriseOne query in that each rule has:

- A "Match Any" or "Match All" setting.
- One or more conditions defined, each being a binary operation on two values.

After creating a rule, you must use the Action column in the Orchestration design panel to determine the course of action for any orchestration steps that are dependent on the rule. See [Completing an Orchestration](#) for more information.

E.5.1.1 Understanding Nested Rules (and Nested Service Requests)

If the input to an orchestration does not meet the "true" condition defined in the orchestration Rule, then it is "false." Typically at this point, the orchestration stops and no further action is taken. However, you can add a "false" condition to the original rule that you can use to invoke another rule, referred to as a nested rule. The nested rule can have an additional condition that when met, invokes a different service request.

Figure E–8 shows an orchestration with a nested rule and nested ServiceRequest. The action in the first ServiceRequest is set to "True" because it is invoked when the true condition of the parent rule is met. The action in the first nested Rule is "False" because it is invoked when the false condition of the parent rule is met. The nested Rule then contains a true condition that invokes another ServiceRequest (to perform a task in an EnterpriseOne form), therefore the action in the nested ServiceRequest is "True." You can repeat the nesting of rules and service requests as required by the design of your orchestration.

Figure E–8 Nested Rules and Service Requests in an Orchestration

Type	Action	Name
CrossReference		JDE_XREF_Sample_SensorLocation
CrossReference		JDE_XREF_Sample_AlertNotificationRecipients
Rule		JDE_RULE_Sample_CBMAccess_1
ServiceRequest	True	JDE_SREQ_Sample_AddCBArt_Alarm
Rule	False	JDE_RULE_Sample_CBMAccess_2
ServiceRequest	True	JDE_SREQ_Sample_AddCBArt_Alarm

E.5.2 Creating a Rule

Create a rule component to define the conditions for the orchestration.

To create a rule:

1. After adding a Rule step to an orchestration, click the **Rule** step row.

The Orchestrator Studio displays the Rule design panel.

2. In the Rule design panel, click the **New Rule** button.

3. In the Rule pop-up field, enter a name for the rule and click the **Ok** button.

Do not include any spaces in a rule name.

As an alternative, you can create a new rule by copying an existing rule. To do so:

- a. Click the **Copy Rule** button.
- b. In the pop-up window, select a rule from the **Rule to Copy** drop-down menu.
- c. In the New Rule field, enter a name for the rule.
4. Select the Match Value drop-down menu and select one of the following values:
 - **Match All.** Select this option if all conditions in the rule must be met.
 - **Match Any.** Select this option if any conditions in the rule can be met.
5. Add a row with a condition by completing the following columns:
 - **Rule Type.** In the drop-down menu, select either string, numeric, or date depending on the format of the input.
 - **Value 1.** The drop-down menu contains a list of the inputs defined for the orchestration. You can select the value here, or you can click and drag the input from the orchestration design panel and drag it to this field.
 - **Operator.** In the drop-down menu, select from the list of operands which include: >, <, >=, <=, =, !=, startsWith, endWith, contains, between, inList.
 - **Literal.** Click this check box to indicate a literal value.
 - **Value 2.** If you selected the Literal check box, manually enter a value. If not, select the value from the list of inputs.
 - **Literal Value Type.** If you selected the Literal check box, click the drop-down menu and select the format of the literal value: string, numeric, data.
6. Add additional rules as necessary.
7. Click the **Save All** button to save the orchestration and the new rule component.

After adding a rule and a service request to an orchestration, you must define the actions for the rule and service request in the orchestration design panel. See [Completing an Orchestration](#) for more information.

E.5.3 Adding an Existing Rule to an Orchestration

You can reuse orchestration components, including rule components. If an existing rule contains the conditions that you want to use for another orchestration, you can reuse the existing rule.

Make sure that you do not modify the rule in any way that would break other orchestrations where the rule is used.

To add an existing rule to an orchestration:

1. After adding the Rule step to the orchestration, click the **Rule** step row.
2. Click the **Rule** drop-down menu and select an existing rule.

3. Click the **Save All** button.

The Orchestrator Studio saves the rule and saves the orchestration with the rule as the orchestration step.

E.5.4 Adding a Custom Java Rule

You can create custom Java for a rule and then add it to the orchestration as a rule, which is referred to as a custom Java rule. Before you can add a custom Java rule to an orchestration, you must create the custom Java as described in [Chapter 7, "Creating Custom Java for Orchestrations"](#).

You can add a new custom Java rule or copy an existing custom Java rule.

To add a new custom Java rule:

1. After adding the Rule step to the orchestration, click the **Rule** step row.
2. In the Rule design panel, click the **New Custom Java** button.
3. In the Custom Java pop-up field, enter a name for the custom Java rule.
Do NOT include spaces in the custom Java name.
4. Click the **Ok** button.
5. In the Class field, enter the path to the Java class, which includes the name of the Java path, for example:

This is the custom Java class that you created to use for the rule. See [Chapter 7, "Creating Custom Java for Orchestrations"](#).

6. Complete the following fields:
 - **Attribute.** Enter the name of the field in the class.
 - **Input Value.** Click the drop-down menu and select the input value. If the attribute is a boolean and you pass in "true", then you could use a default value.
 - **Default Value.** Enter a default, hard-coded value if there is no mapped value. You can also add a hard-coded value to use if the mapped value returns a null.
7. If you make a mistake while configuring any of the fields, you can click the **Restore Custom Java** button which refreshes the custom Java rule to its last saved state.
8. Click the **Save All** button.

The Orchestrator Studio saves the custom Java rule and saves the orchestration with the newly added step.

To add a new custom Java rule by copying an existing custom Java rule:

1. Click the **Copy Java** button.
2. In the pop-up fields, click the **Custom Java to Copy** drop-down menu and select a custom Java rule to copy.
3. Enter a name for the new custom Java rule.
4. Modify the custom Java rule values as necessary.
5. Click the **Save All** button.

E.6 Creating Cross References

This section contains the following topics:

- Section E.6.1, "Understanding Cross References"
- Section E.6.2, "Creating a Cross Reference"
- Section E.6.3, "Adding an Existing Cross Reference to an Orchestration"

E.6.1 Understanding Cross References

The Orchestrator uses a cross reference component of an orchestration to map third-party data to an EnterpriseOne value. The EnterpriseOne value identified in the cross reference is considered the output of the cross reference. The output from the cross reference becomes the input to the service request when you configure the service request.

Each cross reference that you define in the Orchestrator Studio must also be defined in P952000 in EnterpriseOne. In P952000, you must use the "AIS" cross reference type when defining cross references for orchestrations. See [Chapter 5, "Setting Up Cross References and White Lists in EnterpriseOne \(P952000\)"](#) for instructions on how to set up cross references in P952000.

If a cross reference lookup fails in an orchestration, the orchestration is terminated.

E.6.2 Creating a Cross Reference

You must define the orchestration inputs before you can create a cross reference. See [Adding Inputs to an Orchestration](#) for more information.

To create a cross reference:

1. After adding a CrossReference step to an orchestration, click the **CrossReference** step row. The Orchestrator Studio displays the Cross Reference design panel.
2. In the Cross Reference design panel, click the **New Cross Reference** button.
3. In the Cross Reference pop-up field, enter a name for the cross reference. Do NOT include spaces in the cross reference name.
4. In the Object Type field, enter a name for the object type.

This is the object type used to categorize the orchestration cross references in EnterpriseOne. See "Adding Orchestration Cross References" in the *JD Edwards EnterpriseOne Tools Interoperability Guide* for more information.

The Orchestrator Studio displays a grid with an Input Key column and Output Key column, which you use to map the input name to the name of the EnterpriseOne field (output).

5. Click the **Input Key** drop-down menu and select an input.

The inputs can be one or many values. The inputs must correspond to the value or pipe () delimited values in the Third Party Value column in P952000. See ["Chapter 5, "Setting Up Cross References and White Lists in EnterpriseOne \(P952000\)"](#) for more information.

If the drop-down menu does not contain any values, this means that you have not defined the inputs for the orchestration, which you must complete before creating the cross reference. See [Adding Inputs to an Orchestration](#) for more information.

6. Click in the **Output** column to define the names of the output values.

The outputs can be one or many values. The outputs must correspond to the value or pipe () delimited values in the EOne Value column in P952000. See ["Chapter 5, "Setting Up Cross References and White Lists in EnterpriseOne \(P952000\)"](#) for more information.

7. If you enter any values in error, you can click the X button next to the field to remove it.

8. Click the **Save All** button.

The Orchestrator Studio saves the new cross reference component and saves the orchestration with the cross reference step.

The output values in the cross reference are now available for mapping in subsequent orchestration steps.

You can also create a new cross reference by copying an existing cross reference. To do so:

1. After adding the CrossReference step to the orchestration, click the **CrossReference** step in the grid.
2. Click the **Copy Cross Reference** button.
3. In the pop-up window, click the **Cross Reference to Copy** drop-down menu and select an existing cross reference.
4. In the New Cross Reference field, enter a name for the new cross reference.
Do not include spaces when naming orchestration components.
5. Click the **Save** button.

E.6.3 Adding an Existing Cross Reference to an Orchestration

You can reuse orchestration components, including cross references. If an existing cross reference used by another orchestration contains mappings that you want to use for a new orchestration, you can add the existing cross reference to the new orchestration.

Make sure that you do not modify the cross reference in any way that would break other orchestrations that use the same cross reference.

To add an existing cross reference to an orchestration:

1. After adding the CrossReference step to the orchestration, click the CrossReference row.
2. In the Cross Reference drop-down menu, select an existing cross reference.
3. Click the **Save All** button.

The Orchestrator Studio saves the cross reference and saves the orchestration with the cross reference as the orchestration step.

E.7 Creating White Lists

This section contains the following topics:

- ¤ Section E.7.1, "Understanding White Lists"
- ¤ Section E.7.2, "Creating a White List"
- ¤ Section E.7.3, "Adding an Existing White List to an Orchestration"

E.7.1 Understanding White Lists

A white list contains a list of IDs permitted in the Orchestrator. By adding a white list to an orchestration, only inputs with IDs listed in the white list are permitted for processing by the Orchestrator. You add a white list component as a step in the orchestration.

In addition to specifying the permitted IDs in the white list, you must also specify the white list IDs in P952000 in EnterpriseOne. This is the same application that you use to set up and store orchestration cross references. As with cross references, use the "AIS" cross reference type in P952000 for a white list. In P952000, the Third Party App ID should have a value of WHITELIST. The EnterpriseOne value is not used for white lists and will default to NA.

If a white list lookup fails in an orchestration, the orchestration is terminated.

E.7.2 Creating a White List

Before you can create a white list, you must define the inputs for the orchestration as described in [Adding Inputs to an Orchestration](#). When you create a white list, you select the input IDs that you want to add from the list of available inputs in the orchestration.

To create a white list:

1. After adding the **WhiteList** step to the orchestration, click the **WhiteList** step in the grid.
2. In the White List design panel, click the **New White List** button.
3. In the White List pop-up field, enter a name for the white list.
Do NOT include spaces in the white list name.
4. In the Object Type field, enter a name for the object type.

The value you enter in the Object Type field must match a cross-reference object type in the EnterpriseOne Business Services Cross Reference application (P952000).

The cross-reference object type is a named group of records in P952000. For example, you may have thousands of records in P952000; those thousands of records can be grouped by cross-reference object type names such as "Equipment" or "Alert_Notification_Recipients." See [Chapter 5, "Setting Up Cross References and White Lists in EnterpriseOne \(P952000\)"](#) for more information.

5. In the Input Key column, click the drop-down menu and select input that you want to add as a permitted input.
6. Click the **Save All** button.

The Orchestrator Studio saves the new white list component and saves the orchestration with the newly added white list step.

You can also create a new white list by copying an existing white list. To do so:

1. After adding the **WhiteList** step to the orchestration, click the **WhiteList** step in the grid.
2. Click the **Copy White List** button.
3. In the pop-up window, click the **White List to Copy** drop-down menu and select a white list.
4. In the New White List field, enter a name for the new white list.
Do NOT include spaces in the white list name.
5. Click the **Save** button.

E.7.3 Adding an Existing White List to an Orchestration

As mentioned previously, you can reuse orchestration components. If an existing white list contains the IDs that you want to use for another orchestration, you can reuse the existing white list.

To add an existing white list to an orchestration:

1. After adding the **WhiteList** step to the orchestration, click the **WhiteList** step in the grid.
2. In the White List drop-down menu, select an existing white list.
3. Click the **Save All** button.

The Orchestrator Studio saves the white list and saves the orchestration with the white list as the orchestration step.

E.8 Reloading Orchestrations and Orchestration Components

In the Orchestrator Studio, the Reload All Files feature enables you to reload orchestrations and orchestration components to the state in which they were last saved. Use the Reload All Files feature if you do not want to save any modifications that you made.

To reload all files, click the drop-down menu in the upper-right corner of the Orchestrator Studio and then click the **Reload All Files** link.

Each individual orchestration component panel also contains a Restore button that you can use to restore an individual component.

F

Sample Orchestras

This appendix describes the prerequisite for using sample orchestrations and describes how to use the following sample orchestrations which are intended for testing purposes only:

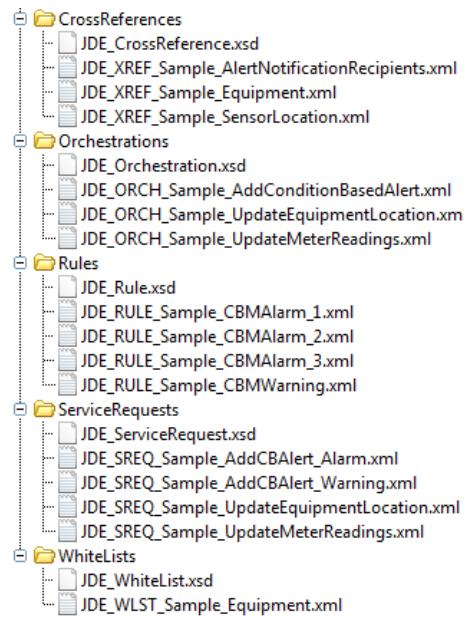
- Section F.1, "Prerequisite"
- Section F.2, "Running the Sample Orchestrations"
- Section F.3, "Add Conditioned Based Maintenance Alert Sample Orchestration"
- Section F.4, "Update Equipment Location Sample Orchestration"
- Section F.5, "Update Meter Reading Sample Orchestration"

F.1 Prerequisite

If you have not already done so, download the Sample Orchestrations package. See [Section 2.1, "Downloading and Installing Orchestrator Studio"](#) for download instructions.

F.2 Running the Sample Orchestrations

Use the EnterpriseOne Orchestrator Client to run each of the sample orchestrations. Before you can run the sample orchestrations, copy the XML files for each sample to the appropriate folder in the Orchestration directory, as shown in [Figure F-1](#).

Figure F–1 Orchestration Directory Folders and Files

After the XML files are in the directory, use the Orchestration Client to enter inputs and test the sample orchestrations. The following sections in this appendix describe the inputs, or name-value pairs, in the sample orchestration XML files. You enter these inputs in the Orchestration Client to test the sample orchestrations. See [Chapter 9, "Testing Orchestrations in the EnterpriseOne Orchestrator Client"](#) for instructions on how to test the sample orchestrations.

F.3 Add Conditioned Based Maintenance Alert Sample Orchestration

The "Add Conditioned Based Maintenance Alert" sample orchestration conditionally creates conditioned based alerts based on vibration and temperature readings from a device. The orchestration is defined in the JDE_ORCH_Sample_AddConditionBasedAlert.xml located in the Orchestrations folder. This sample orchestration includes:

- ❑ A cross reference that converts the incoming SensorID into EquipmentNumber and MeasurementLocation defined in the JDE_XREF_Sample_SensorLocation.xml file in the CrossReferences folder.
- ❑ An orchestration step to find the WarningRecipient and AlarmRecipient from the EquipmentNumber, which is also a cross reference defined in JDE_XREF_Sample_AlertNotificationRecipients.xml.
- ❑ A series of rules to determine if an alert is needed and if so, whether it is an alarm or warning. The rules used for this orchestration are defined in the following XML files in the Rules folder:
 - ❑ JDE_RULE_Sample_CBMAccess_1.xml
 - ❑ JDE_RULE_Sample_CBMAccess_2.xml
 - ❑ JDE_RULE_Sample_CBMAccess_3.xml
 - ❑ JDE_RULE_Sample_CBMWarning.xml
- ❑ If necessary, the alert is created using either the JDE_SREQ_Sample_AddCBALert_Alarm.xml or JDE_SREQ_Sample_AddCBALert_Warning.xml defined in the

ServiceRequests folder. These service requests invoke the edit form (W1311B) of P1311 application to create the conditioned based alert.

F.3.1 Sample Input

In the Orchestrator Client, use the following inputs to test the Add Conditioned Based Maintenance Alert orchestration.

```

"inputs": [
    {
        "name": "SensorID",
        "value": "1-345285J"
    },
    {
        "name": "Date",
        "value": "1433311200000"
    },
    {
        "name": "Time",
        "value": "12:15:15"
    },
    {
        "name": "VibrationReading",
        "value": "100"
    },
    {
        "name": "TemperatureReading",
        "value": "350"
    }
]
}

```

F.4 Update Equipment Location Sample Orchestration

The "Update Equipment Location" sample orchestration creates a new equipment location to store the current latitude and longitude of the asset. The orchestration is defined in the JDE_ORCH_Sample_UpdateEquipmentLocation.xml file located in the Orchestrations folder. This sample orchestration includes:

- A cross reference orchestration step that converts the incoming DeviceID into EquipmentNumber. This cross reference is defined in the JDE_XREF_Sample_Equipment.xml file in the CrossReferences folder.
- A final orchestration step that includes the service request JDE_SREQ_Sample_UpdateEquipmentLocation.xml. This service request runs a series of applications using the application stack service to create the equipment location details. The application stack service flow performs the following tasks:
 1. Accesses P1704 application - Work with Equipment Locations.
 2. Invokes the Add button to create a new equipment location header record.
 3. After the record is created, the record is queried back in the Work With Equipment Locations form.
 4. It then accesses the Details form and saves the latitude, longitude, and elevation in this form.

F.4.1 Sample Input

```
{  
    "inputs": [  
        {  
            "name": "CustomerNumber",  
            "value": "4244"  
        },  
        {  
            "name": "SiteNumber",  
            "value": "4244"  
        },  
        {  
            "name": "DeviceID",  
            "value": "1-345213A"  
        },  
        {  
            "name": "Latitude",  
            "value": "39.649844"  
        },  
        {  
            "name": "Longitude",  
            "value": "-104.856342"  
        },  
        {  
            "name": "Elevation",  
            "value": "5642"  
        }  
    ]  
}
```

F.5 Update Meter Reading Sample Orchestration

The "Update Meter Reading" sample orchestration updates meter readings of a piece of equipment. The orchestration is defined in the JDE_ORCH_Sample_UpdateMeterReadings.xml file located in the Orchestrations folder. This sample orchestration includes:

- An initial white list orchestration step that validates that the incoming EquipmentNumber is allowed to run the orchestration.
- A final orchestration step that includes the service request JDE_SREQ_Sample_UpdateMeterReadings.xml. This service request runs the Speed Meter Readings application (P12120U) to update the fuel meter reading and the hour meter reading of the passed in equipment number.

F.5.1 Sample Input

```
{  
    "inputs": [  
        {  
            "name": "EquipmentNumber",  
            "value": "34665"  
        },  
        {  
            "name": "NewFuelMeterReading",  
            "value": "13.2"  
        }  
    ]  
}
```

```
        "name": "NewHourMeterReading",
        "value": "101.7"
    }
]
}
```

Troubleshooting

This appendix contains the following topics:

- [Section G.1, "Enable Debugging on the AIS Server"](#)
- [Section G.2, "Troubleshooting Orchestration Runtime Issues"](#)

G.1 Enable Debugging on the AIS Server

Turn on debugging on the AIS Server to generate an AIS Server log file. The log file includes information for all orchestrations unless you activate user level logging. User level logging generates logs based on the EnterpriseOne user ID. See "Available Log Files" in the *JD Edwards EnterpriseOne Tools Server Manager Guide* for more information about generating and managing log files.

The details in the log file can help you troubleshoot and resolve orchestration issues and EnterpriseOne connection issues.

G.2 Troubleshooting Orchestration Runtime Issues

Interruption of EnterpriseOne Web Client When Running the JD Edwards EnterpriseOne Orchestrator Client

If a user is signed in to the EnterpriseOne Web client and then opens the Orchestrator Client in a new tab in the same browser, the user can no longer perform any actions in the EnterpriseOne Web client. The EnterpriseOne Web client session is interrupted and prompts the user to sign in again.

This issue is caused by an Orchestrator Client JSESSIONID cookie conflict with WebSphere and occurs when both the EnterpriseOne AIS Server and EnterpriseOne HTML Server are deployed on WebSphere on the same host. Both servers use the JSESSIONID cookies for session management, which causes a conflict because both JSESSIONID cookies are configured with a generic path.

To fix this issue, set up the cookie path of the AIS Server:

1. In the WebSphere administration console, navigate to the server: **Application servers, *ais_server_name*.**
2. Under Container Settings, select **Session Management**.
3. Click **Enable Cookies** link.
4. Under Cookie path, make sure the "**Set cookie path**" option is selected and set the cookie path to:

/jderest

5. Restart the server.