

JD Edwards EnterpriseOne Tools

Orchestrator Guide

9.2

Copyright © 2011, 2024, Oracle and/or its affiliates.

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 embedded, installed or activated on delivered hardware, and modifications of such programs) and Oracle computer documentation or other Oracle data delivered to or accessed by U.S. Government end users are "commercial computer software" or "commercial computer software documentation" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, reproduction, duplication, release, display, disclosure, modification, preparation of derivative works, and/or adaptation of i) Oracle programs (including any operating system, integrated software, any programs embedded, installed or activated on delivered hardware, and modifications of such programs), ii) Oracle computer documentation and/or iii) other Oracle data, is subject to the rights and limitations specified in the license contained in the applicable contract. The terms governing the U.S. Government's use of Oracle cloud services are defined by the applicable contract for such services. 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 Inside 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, Epyc, and the AMD 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	i
<hr/>	
1 Understanding the JD Edwards EnterpriseOne Orchestrator	1
Welcome	1
JD Edwards EnterpriseOne Orchestrator Overview	1
How It Works	4
EnterpriseOne Architecture for Orchestrator	5
2 Getting Started	7
Certifications (Formerly Known as Minimum Technical Requirements)	7
Prerequisites	7
Before You Begin	8
Accessing the Orchestrator Studio	8
WebSphere WebSocket Support Configuration (Release 9.2.4.3)	9
Set Up a Temporary Directory on the AIS Server for File Transfers	12
Enabling Orchestrator Studio Single Sign-on (Release 9.2.8.2)	12
3 Designing an Orchestration	15
Understanding the Orchestration Design Process	15
Identifying the Problem and Solution	16
Identifying the Data for the Orchestration	17
Identifying the Rules for the Orchestration	17
Identifying the Cross Reference and White List Information for the Orchestration	17
Identifying the Service Request Information for the Orchestration	18
4 Configuring Orchestration Components using Orchestrator Studio 9.2.4	19
Understanding the Orchestrator Studio and Orchestration	19
Navigating the Orchestrator Studio	21
Creating Service Requests	31
Working with Orchestrator Categories (Release 9.2.4.2)	34

Configuring a Form Request in the Orchestrator Studio	35
Configuring a Data Request	44
Configuring a Report Service Request	51
Configuring a Watchlist Service Request	55
Configuring a Message Request	55
Configuring a Connector Service Request	68
Configuring a Custom Service Request	86
Creating Connection Soft Coding Records for Connector Service Requests	91
Creating Rules	101
Creating Cross References	103
Creating White Lists	105
Creating Attachments (Release 9.2.6)	106
Creating Logic Extensions (Release 9.2.6)	111
5 Creating Orchestrations with Orchestrator Studio 9.2.4	143
Creating Orchestrations	143
Creating Schedules for Orchestrations	182
Working with Scheduler	184
Updating to Version 3 Orchestrations	193
Restoring Orchestrations and Orchestration Components	194
Supported Input Message Formats	194
Orchestration Security Considerations	196
Exporting Orchestration Components from the Orchestrator Studio	197
Importing Orchestration Files in the Orchestrator Studio	198
Expanding and Collapsing the Nested Orchestration Steps (Release 9.2.4.4)	200
Printing an Orchestration (Release 9.2.4.4)	202
Monitoring Orchestrations (Release 9.2.7.4)	203
6 Setting Up Cross References and White Lists in EnterpriseOne (P952000)	205
	205
7 Orchestrator Health and Exception Monitoring	207
Understanding the EnterpriseOne Orchestrator Monitor	207
Prerequisites	207
Accessing the Orchestrator Monitor	207
Monitoring UDOs Based on Different EnterpriseOne User, Environment, or Role	208

Refreshing the Data Displayed in the Orchestrator Monitor	208
Resetting the Data Displayed in the Orchestrator Monitor	209
Monitoring Orchestrator Health	209
Monitoring Orchestrator Exceptions	211
Monitoring Orchestrator Run Details (Release 9.2.7.4)	217
Managing Orchestrator Health and Exception Records in EnterpriseOne	221
8 Creating Orchestrations for System Administration with Orchestrator Studio	225
Understanding the Orchestrations for System Administration	225
9 Creating Custom Java for Orchestrations	227
Understanding Custom Java for Orchestrations	227
Creating Custom Java	227
Deploying Custom Java	228
10 Using Scripting Languages for Custom Service Requests, Rules, and Manipulating Output	233
Using Scripting Languages	233
Understanding Groovy for Orchestration Components	233
Groovy Template for a Custom Service Request	233
Groovy Template for a Custom Rule	234
Groovy Template for Manipulating Output from a REST Connector Response	235
Groovy Template for Manipulating Output from an Orchestration Response	236
Defining the Input and Output Variables	237
Additional Attributes and Methods Available in the Groovy Script Templates	238
Installing Optional Scripting Languages on the AIS Server (Release 9.2.5.4)	242
Installing Optional Scripting Languages on the AIS Server (Release 9.2.6)	250
11 Testing Orchestrations	251
Understanding Run Orchestrations	251
Testing an Orchestration	252
Debugging an Orchestration (Release 9.2.4.3)	255
12 Administering the Orchestrator Studio and Orchestrations	259
Understanding Orchestration Life Cycle Management	259

Managing Orchestrator Studio Security	259
Setting Up UDO Security for Orchestrator Studio Users	261
Clearing Orchestration Cache on the AIS Server	262
Setting Up Orchestrator Health and Exception Monitoring	264

13 Understanding the AIS Server Discovery Service **267**

	267
Example: Orchestration Details in the JSON Response from the Discovery Service	267

14 Open Standards for Orchestrations **271**

Understanding Open Standards for Orchestrations	271
Discovering Orchestrations Using Open Standards	271

Preface

Welcome to the JD Edwards EnterpriseOne documentation.

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>

Conventions

The following text conventions are used in this document:

Convention	Meaning
Bold	Boldface type indicates graphical user interface elements associated with an action or terms defined in text or the glossary.
<i>Italics</i>	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
Monospace	Monospace type indicates commands within a paragraph, URLs, code examples, text that appears on a screen, or text that you enter.
> Oracle by Example	Indicates a link to an Oracle by Example (OBE). OBEs provide hands-on, step-by-step instructions, including screen captures that guide you through a process using your own environment. Access to OBEs requires a valid Oracle account.

1 Understanding the JD Edwards EnterpriseOne Orchestrator

Welcome

Welcome to the *JD Edwards EnterpriseOne Tools Orchestrator Guide*. The main chapters in this guide contain instructions for Orchestrator Studio 9.2.4, the latest version of the Orchestrator Studio supported with a minimum release of EnterpriseOne Tools 9.2.4.

This guide has been updated for JD Edwards EnterpriseOne Tools releases 9.2.4.2, 9.2.4.3, 9.2.4.4, 9.2.5, 9.2.5.2, 9.2.5.4, 9.2.6, 9.2.6.2, 9.2.6.3, 9.2.6.4, 9.2.7, 9.2.7.2, 9.2.7.3, 9.2.7.4, 9.2.8, 9.2.8.2, and 9.2.8.3.

For instructions for prior releases of the Orchestrator Studio, see the *JD Edwards EnterpriseOne Orchestrator Guide for Studio Version 8 and Prior*.

The Orchestrator functionality supports various integrations including integrations to Cloud services, third-party applications, custom programs, IoT configuration and many more. You can use the Orchestrator functionality for all your REST-based integrations to EnterpriseOne. You can also use it for integrating JD Edwards EnterpriseOne with IoT devices.

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 Getting Started chapter in this guide.

JD Edwards EnterpriseOne Orchestrator Overview

The JD Edwards EnterpriseOne Orchestrator is a key component of your JD Edwards digital platform. It can transform the EnterpriseOne system from a transaction-based system of records into a system that provides a dynamic reflection of your real-time business operations. Orchestrations provide access to your EnterpriseOne data and applications as services—technically, industry-standard REST services—which you can easily create using the Orchestrator Studio. Internet-connected devices, third-party applications, Cloud services, mobile devices, and even EnterpriseOne itself, can invoke these services and take advantage of the business data and EnterpriseOne application functionality in your system. It all happens within the framework of user-defined objects, not custom-developed objects that need to be built into packages, deployed, and retrofitted during upgrades. All orchestrations are subject to EnterpriseOne application, data, and user-defined object security.

The basic framework of orchestrations as REST services enables a wide variety of usage patterns that provide secure access to your EnterpriseOne business data and applications. For example:

- Internet of Things. Devices such as equipment, sensors, or meters can invoke orchestrations that, in turn, run EnterpriseOne applications. For example, an Internet-connected meter could send readings directly to the EnterpriseOne Meter Reading application.
- Integrations. Third-party systems and Cloud services can invoke orchestrations as REST-based services, allowing for easy and lightweight integrations. For example, a sales order could be created in a third-party

CRM system, which could then invoke an orchestration to create a matching sales order in EnterpriseOne. Conversely, orchestrations can also make outbound calls to other REST services. For example, an orchestration could use the information in a sales order to call out to a third-party transportation system to get a shipping quote.

- **Mobile Applications, Chatbots, and Alternative User Interfaces.** Orchestrations can be used as services to send and receive data between user interfaces and EnterpriseOne. You can use your preferred development framework to build the user interface itself and let orchestrations do the work of exchanging data with EnterpriseOne.
- **Process Automation and Simplification.** Some business processes in EnterpriseOne comprise many steps. In some cases, those steps are connected and sequentially ordered to form a continuous flow across applications; in other cases, the steps are not connected. In such cases, the end user must know which applications to use and in which order. Using form extensions you can also invoke orchestrations directly from EnterpriseOne application forms, thus extending the functionality of the applications without custom modifications.
- **Notifications.** As an extension of the Orchestrator framework, notifications can assess your EnterpriseOne data, and also any REST-enabled external service, to detect conditions or events that need attention. Messages are then sent out to subscribers who need to know that an event has occurred that requires attention. The messages can contain action links to help the recipients act quickly with an appropriate response.
- **Scheduled Orchestrations.** The Orchestrator includes a built-in Scheduler that allows you to automate when the orchestrations run. For example, orchestrations can continually check your EnterpriseOne system for events such as credit limit violations, late shipments, large sales orders, or any other condition that impacts your business.
- **System Administration.** Because the Orchestrator can make outbound calls to REST services, and the JD Edwards Server Manager is enabled with REST services, the two can be combined to automate a wide variety of monitoring and administrative activities.

The Orchestrator Studio is a web-based application that business analysts and technical developers alike will use to create, test, and deploy orchestrations and notifications. The Orchestrator Studio provides an intuitive and easy-to-use graphical interface that enables you to develop and deploy orchestrations without the need for programming or system administration skills, or development. You can even build services by turning on the Process Recorder in EnterpriseOne and then using the applications as you normally would. The Process Recorder captures your steps enabling you to turn those steps into an orchestration using the Orchestrator Studio, all without programming or technical skills.

With the 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 when the inventory is low.
- Alert users when an equipment is down.
- Send an alert when a customer has reached the credit limit.
- Automatically create sales orders.
- Onboard employee records into the system.
- Check on the health of an instance of a server.

Using the Orchestrator you can now transform how you use your EnterpriseOne system to:

- Collect, filter, analyze, and act on real-time data as it is being transmitted by various sources such as a machine, a mobile app, a Cloud service, and any third-party applications and devices.
- Eliminate 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.
- Integrate with external systems and Cloud services, enabling your EnterpriseOne system to send and receive data for integrated business processes.

- Free your EnterpriseOne users from tedious tasks by transforming manual EnterpriseOne business processes into automated operations that meet your specific business needs.
- Empower your EnterpriseOne system to run orchestrations automatically on schedule.

Invoking Orchestrations and Notifications from an EnterpriseOne Interactive Application Using Form Extensibility

You can use the form extensibility framework to invoke orchestrations from an interactive application by associating orchestrations with the events on an EnterpriseOne form. You can also associate notifications with the events on a form, invoke orchestrations in the synchronous mode, and map orchestration outputs to the form controls if any output has been defined for the orchestration. For more information, see *"Extending EnterpriseOne Forms by Associating Orchestrations with Events (Release 9.2.3.3)" in the JD Edwards EnterpriseOne Tools Using and Approving User Defined Objects Guide* .

Invoking Orchestrations and Notifications from an EnterpriseOne Interactive or Batch Application

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:

- An event on a form, such as a button or a field
- A report (UBE)
- A table trigger
- Anywhere you would invoke a business function

With the B98ORCH business function, you can extend EnterpriseOne applications beyond transactional boundaries by linking disparate ERP tasks into a single automated business process. Alternatively, you can enable an EnterpriseOne application to automatically pass data to a REST-enabled third-party application through an orchestration or 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* .

Invoking Orchestrations and Notifications from a Composed EnterpriseOne Page

You can run orchestrations and notifications from a Composed EnterpriseOne Page by associating the orchestration or notification with a tile in the designer pane of the Composed EnterpriseOne Page. For more information, see *"Adding and Configuring a Component in Designer Pane"* and *"Invoking Orchestrations and Notifications from a Composed EnterpriseOne Page (Release 9.2.6)"* .

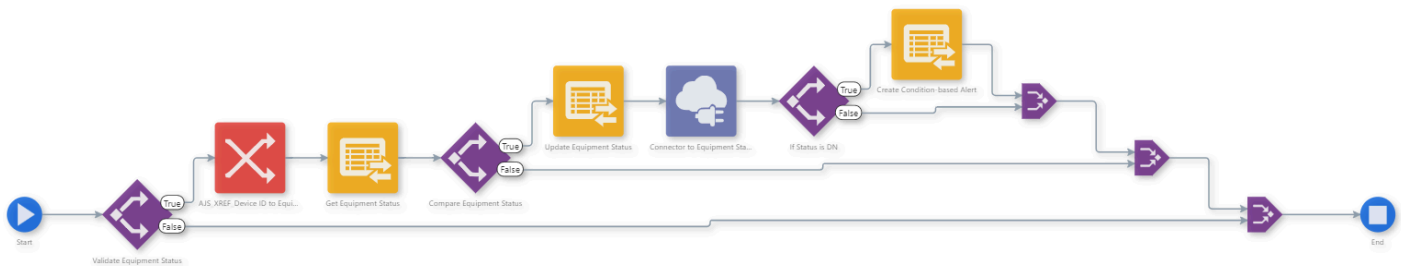
How It Works

When you create and save an orchestration in the Orchestrator Studio, it automatically becomes a published REST service. The name of an orchestration is used to define an endpoint on the AIS Server. The endpoint URL is

`http://<AIS_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 following illustration shows how the EnterpriseOne Orchestrator processes orchestrations. This illustration depicts an orchestration that includes rules, cross-references, form requests, and a connector to a notification.



An external client, which might be a third-party application, a Cloud service, an IoT device, a mobile application, the Scheduler, or anything that is capable of sending a REST request, will invoke the orchestration by its endpoint address and provide the necessary inputs in the body of the REST request.

You can use the following Orchestrator components as building blocks in any order and in any combination to create orchestrations that are as simple or complex as your service requires.

- **Orchestration.** An orchestration is the high-level object that defines the name of the orchestration (the REST endpoint), the inputs, the outputs (REST response), and the order and structure of the other components listed below.
- **Service request.** A service request is a general term for an orchestration component that provides some actual service or "work." As such, service requests are the essential components of an orchestration. The types of service requests are listed below.
 - **Form request:** A form request invokes one or more EnterpriseOne application forms. You use this type of request to automate EnterpriseOne applications. You can create a form request in the Orchestrator Studio, or you can use the Process Recorder to record your actions in EnterpriseOne and then edit the

resulting form request in the Orchestrator Studio. Form requests can use EnterpriseOne applications to read data and also to write data back to EnterpriseOne tables just as a human user would.

- **Data request:** A data request reads data from any EnterpriseOne table or business view. You can provide inputs into a data request so that it performs certain steps, for example to filter data, and return back data sets (arrays) or aggregations such as sums or averages. Data requests are read-only components.
 - **Report:** A report request will run any EnterpriseOne report (UBE). You can specify inputs, for example, the report version, data selection, and data sequencing. The report will execute as if a human user had submitted the report in EnterpriseOne. The report request can return the server and job number of the report for use in subsequent orchestration steps.
 - **Watchlist:** A Watchlist request will return all the information that is available about a Watchlist, including the number of rows (that is, the number that is shown in the badge of a Watchlist), whether that number has triggered a warning or critical threshold, and other Watchlist parameters.
 - **Connector:** Orchestrations can invoke other orchestrations or notifications. A connector request is used as the step within a main orchestration to invoke another orchestrations or notifications. Connector requests are also used to invoke external REST services, external databases, and File Transfer Protocol (FTP) sites for sending and receiving documents.
 - **Custom request:** A custom request enables you to use the Groovy scripting language or Java to create an orchestration step that performs any service that you can conceive and develop.
 - **Message:** A message request allows you to send an e-mail message to one or more recipients. You can create the subject and body of the message, including text substitution variables that are populated from previous orchestration steps. The message can include links to EnterpriseOne applications, links to external URLs, links that launch other orchestrations or notifications, and attachments.
- **Rule.** A rule contains a set of conditions against which the orchestration input or values from orchestration steps are 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 build custom rules using Groovy or Java that enable you to define complex rules..
 - **Cross reference.** A cross-reference is a set of data relationships defined by the designer of the orchestration that enriches the minimal input from devices. For example, the serial number of a device can be cross-referenced to an EnterpriseOne equipment number for use in service requests.
 - **White list.** A white list is an initial rudimentary pass or fail check of the incoming message's device signature of the incoming message against a predefined list of signatures. A white list provides an additional layer of security to the Orchestrator security..
 - **Schedule.** A schedule is a definition of the frequency at which an orchestration or notification will automatically run. A schedule is its own user defined object, enabling you to reuse common schedules, such as "once per day" or "once per hour." The Orchestrator Studio enables you to define simple schedules easily. You can also use the CRON syntax to define more complex schedules, such as "every Monday at 8:00 am."

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* .

2 Getting Started

Certifications (Formerly Known as Minimum Technical Requirements)

Customers must install the supported platforms for the release, which can be found in the Certifications tab on My Oracle Support: <https://support.oracle.com>. For the EnterpriseOne Orchestrator Studio 9.2.4, refer to the certifications for the JD Edwards EnterpriseOne AIS Server product.

For more information about JD Edwards EnterpriseOne minimum technical requirements, see this document on My Oracle Support: JD Edwards EnterpriseOne Minimum Technical Requirements Reference (Doc ID 745831.1). The document is available here:

<https://support.oracle.com/rs?type=doc&id=745831.1>

Note: Orchestrator Studio 9.2.4 is not supported on Internet Explorer (IE).

Prerequisites

Complete the following prerequisites:

- You must be running a minimum of EnterpriseOne Tools Release 9.2.4.

Always apply the latest EnterpriseOne Tools software update to use the latest available features.

- Deploy the Application Interface Services (AIS) Server 9.2.4.0.

The Orchestrator Studio 9.2.4 is deployed along with the AIS Server 9.2.4.0 and can be accessed by using the AIS Server URL. Therefore, the Orchestrator Studio 9.2.4 does not require any other set-up or installation steps. The JD Edwards EnterpriseOne Orchestrator Studio 9.2.4 is installed on AIS Server instance.

You can use an existing AIS Server or deploy a new AIS Server 9.2.4.0 instance through the Server Manager for running the orchestrations. It is recommended that you set up two AIS Server instances for an Orchestrator configuration, one for testing orchestrations and one for production. See "Create an Application Interface Services (AIS) Server as a New Managed Instance" in the *JD Edwards EnterpriseOne Tools Server Manager Guide*.

Make sure that the AIS Server is configured with your EnterpriseOne system.

You should set up an additional HTML Server instance for processing AIS Server requests only. This is recommended so that the performance of the EnterpriseOne HTML Server that is used by EnterpriseOne web client users is not impacted by AIS Server requests. For more information, see "Additional EnterpriseOne HTML

Server Instance for Processing AIS Requests" in the *JD Edwards EnterpriseOne Application Interface Services Server Reference Guide* .

Note: You will not be able to login to the Orchestrator Studio 7.3 or previous versions with the Application Interface Services (AIS) Server 9.2.4.0 instance. You will be able to login to the Orchestrator Studio 8.0 with AIS 9.2.4.0, but you will be able to create only V2 orchestrations. Orchestrator Studio 8.0 is intended only for maintenance and migration.

- Ensure that the Orchestration feature is enabled and that all related UDO security is set up properly. For more information, see:
 - *"Managing UDO Feature Security" in the JD Edwards EnterpriseOne Tools Security Administration Guide*
 - *"Define Allowed Actions for UDO Types" in the JD Edwards EnterpriseOne Tools Security Administration Guide*
 - *Managing Orchestrator Studio Security* in this guide
- Ensure that the EnterpriseOne HTML Server and AIS Server keystores are set so that both are using either the demo keystore or the same certificate. If you want to configure your own keystore, see "Configuring EnterpriseOne HTML Server for JSON Web Token (JWT) (9.2.3.2)" or "Configuring EnterpriseOne HTML Server for JSON Web Token (JWT) (9.2.0.5)" in the *JD Edwards EnterpriseOne Tools System Administration Guide* .
- Ensure that the HTML Server is configured as a trusted node for the Security Server. See "Setting Up JD Edwards EnterpriseOne Single Sign-On" in the *JD Edwards EnterpriseOne Tools System Administration Guide* .

Note: Orchestrator Studio 9.2.4 is the latest version which requires a minimum of EnterpriseOne Tools 9.2.4. For instructions about implementing the prior versions of the Orchestrator Studio, see Chapter 2, "Implementing the Orchestrator Studio" in the *JD Edwards EnterpriseOne Tools Orchestrator Guide for Studio Version 8 and Prior* .

Before You Begin

Orchestrator Studio 9.2.4 access requires a separate sign-in with a valid EnterpriseOne user ID and password. Before users can sign in to Orchestrator Studio 9.2.4, an administrator must add each EnterpriseOne user as an allowed user in Work With User/Role Security. See *Managing Orchestrator Studio Security* 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.

Accessing the Orchestrator Studio

To access the Orchestrator Studio 9.2.4:

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

`http://<ais_server>:<port>/studio`

Note: Orchestrator Studio 9.2.4 is not supported on Internet Explorer (IE).

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 that is used for testing, not a production environment.

3. Click the **Login** button.

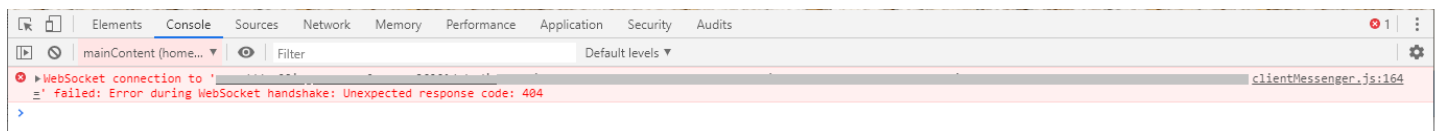
WebSphere WebSocket Support Configuration (Release 9.2.4.3)

Starting with Tools Release 9.2.3.4, the Orchestrator Studio sends notifications during a session timeout. In the case of a timeout, you will be automatically redirected to the Orchestrator Studio login page. This session management functionality relies on the WebSocket technology.

When the Orchestrator Studio is deployed on WebSphere, you must enable the WebSocket functionality using the WebSphere WebSocket configuration.

You can observe the following symptoms if you have not used the WebSphere WebSocket configuration:

- When you access the Orchestrator Studio using a web browser, the console will display the following error message:



- The notification that the user's session has timed out will not be displayed. The user will not be redirected to the login page when a session times out..

WebSphere Configuration

The WebSphere configuration involves the following two steps:

1. Expose the back-end port of the AIS Server as a host alias when you configure virtual hosts for the application.
2. Configure the "Address include list" or the "Hostname include list" option or both the options to restrict communication on the back-end port to only itself or the machine that is hosting the application.

After you complete these two configurations in the WebSphere console, you must bounce the AIS Server using the Server Manager console.

Configuring the Virtual Hosts

Finding the AIS Server Back-End Port

1. Login to the WebSphere console.
2. Navigate to Servers, Server Types, WebSphere application servers.

3. Select the AIS server.
4. Select the Ports.

You can expand the drop-down list to see the ports or click on the option to view all the ports.

5. Search for the WC_defaulthost port and make a note of the port.
If you are using SSL, make a note of the WC_defaulthost_secure port.

Exposing the Back-End Ports as Virtual Hosts

1. Navigate to Environment, VirtualHosts.
2. Select the AIS server that you want to configure.
3. Select Host Aliases.
4. Click New.
5. Enter * in the Host Name field.
6. Enter the port number in the Port field.
Repeat the steps 1-6 for the SSL or non-SSL port.
7. Click Apply.
8. Click Save to save the settings directly to the master configuration.

You will have two or three (if supporting SSL port also) ports listed. The following example displays the configuration for a non-SSL port:

The screenshot shows the WebSphere software console interface. The main window displays the 'Virtual Hosts' configuration page for 'Host Aliases'. The page includes a table with the following data:

Select	Host Name	Port
<input type="checkbox"/>	*	88
<input type="checkbox"/>	*	9130
Total 2		

The left sidebar shows the navigation tree with 'Environment' > 'Virtual Hosts' selected. The right sidebar contains help information, including 'Field help', 'Page help', and 'Command Assistance'.

Including the Address or Host Configuration

1. Login to the WebSphere console.

2. Navigate to Servers, Server Types, WebSphere application servers.
3. Select the AIS server.
4. Select Ports.
You can expand the drop-down list to see the list of ports or click the option to view all the ports.
5. Search for the WC_defaulthost or WC_defaulthost_secure port or both the ports depending on the port that you are using.
6. Click the 'View associated transports' link in the last column.
7. Click WCInboundDefault.
8. Click TCP inbound channel (TCP 2).
9. In the "Address include list" field, enter the IP address of the machine on which the AIS Server is deployed.
10. Click Apply.
11. Click Save to save the configurations directly to the master configuration.

[Application servers](#) > [AIS_88](#) > [Ports](#) > [Transport Chain](#) > [WCInboundDefault](#) > [TCP inbound channel \(TCP_2\)](#)
Use this page to configure a TCP inbound channel for inbound network traffic.

Configuration

General Properties

+ Transport channel name

Port

Thread pool

+ Maximum open connections

+ Inactivity timeout
 seconds

Address exclude list

Address include list

Hostname exclude list

Hostname include list

For information on configuring the Address or Host, see:

https://www.ibm.com/support/knowledgecenter/en/SSAW57_8.5.5/com.ibm.websphere.nd.multiplatform.doc/ae/urun_chain_typetcp.html

Note:

- After you complete these configurations, you must ensure that you bounce the AIS server.
- You have to reapply these configurations after every deployment of an AIS Server component from Server Manager.

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.

Note:

- *Configuring a REST Connector to Invoke a REST Service*
- *Configuring a REST Connector to Transfer Files to a REST Service*

Enabling Orchestrator Studio Single Sign-on (Release 9.2.8.2)

Users can enable Single Sign-On for Orchestrator Studio by configuring the below settings:

1. In the Server Manager Console, select an AIS Server instance, Configuration (left side panel), Advanced (from the drop-down menu), Security Information, Application Interface Services Security Settings.
2. Ensure that **Enable Studio Single Sign-on** check box is selected.
3. Provide a valid **IDCS SSO Sign-Off URL** to be used to terminate the SSO session.

These settings are depicted in the screenshot below:



After these settings are synchronized, Orchestrator Studio can be accessed using Single Sign-On by entering the below URL in a web browser :

`https://<app_gateway_server>:<app_gateway_port>/studio/access.html`

For more information on setting up Orchestrator Studio on App-Gateway refer the following document on [LearnJDE](#):

JD Edwards EnterpriseOne Single Sign-On Using Identity and Access Management with Microsoft Entra ID

Note: The HTML server that the AIS server uses must also be configured with the same OCI Identity Access Management Single Sign-On.

The following conditions will cause a 401 Unauthorized page to display:

- Accessing the application without providing a valid IDCS SSO Sign-Off URL
- Accessing the application using `/studio` or `/login.html`

3 Designing an Orchestration

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.

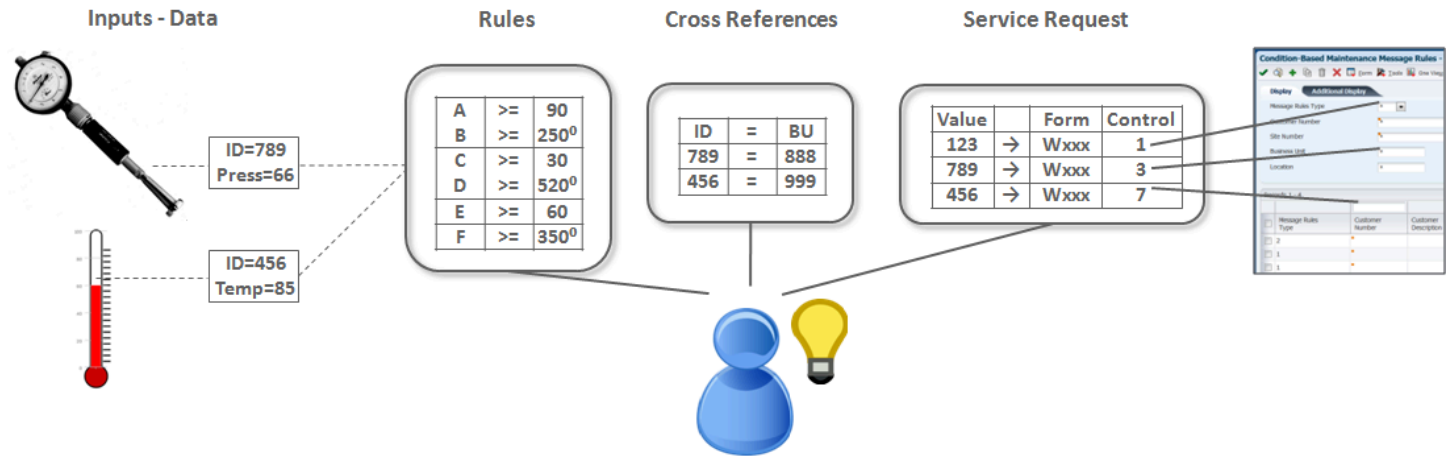
Before you can create an orchestration, you need to perform an analysis to:

- 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 .

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. This graphic shows 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.



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.

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 *this graphic* highlights the data that Company A identified for their orchestration.

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 a Rule* 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 *this graphic* shows the rules and conditions that Company A is using to determine which data should be processed by the orchestration.

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 a Cross Reference* 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 *this graphic* highlights the cross references Company A is using in the orchestration.

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 a Component* section in this guide.

Example: Company A's Service Request Analysis

The "Service Request" area in *this graphic* highlights the data Company A is using for the service request.

4 Configuring Orchestration Components using Orchestrator Studio 9.2.4

Understanding the Orchestrator Studio and Orchestrations

Note: Orchestrator Studio 9.2.4 is the latest version which requires a minimum of EnterpriseOne Tools 9.2.4. If you have not installed Orchestrator Studio 9.2.4, see *Getting Started* chapter in this guide. For instructions for using prior versions of the Orchestrator Studio, see the *JD Edwards EnterpriseOne Tools Orchestrator Guide for Studio Version 8 and Prior*.

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:

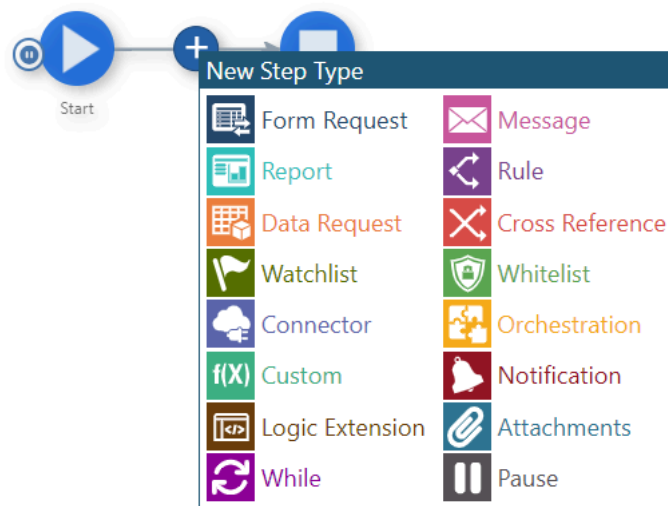
- **Orchestrations.** An orchestration is the master component that provides a unique name for an orchestration process. The orchestration is where you define the inputs or the expected incoming data for the orchestration. It also includes orchestration steps, which are invocations to the other components that are described in this list. When the Orchestrator invokes an orchestration, it processes the steps that are defined in the orchestration.
- **Notifications.** A notification is a process that can run independent of an orchestration. It enables the system to notify users of business events in real time. The notification can contain boilerplate text and a shortcut to an EnterpriseOne application and can be configured to execute a Watchlist or an orchestration.
- **Form Requests.** A service request to perform a business transaction or query data in EnterpriseOne.
- **Data Requests.** A service request to retrieve values from an EnterpriseOne table or business view.
- **Reports.** A service request to invoke an EnterpriseOne report.
- **Watchlists.** A service request to retrieve Watchlist data from EnterpriseOne.
- **Connectors.** A service request to invoke a REST service, database, a notification, or another orchestration.
- **Connections.** A connector service requests to provide a secure access to external resources, such as a REST service, database, or an orchestration or notification on another EnterpriseOne system
- **Custom Requests.** A custom service request to execute a custom process using custom Java or Groovy. Starting with Tools 9.2.4.3, you can call a business function directly from an orchestration.
- **Messages.** A request to send a message about a transaction to EnterpriseOne users or external users.
- **Rules.** A set of conditions against which the input to the orchestration is evaluated to produce a true or false state. A false outcome or a true outcome for a rule can invoke further orchestration steps. You can also nest rules, a process through which an outcome of one rule can invoke a different rule, to produce complex evaluations. You can also use custom Java to define rules.

Starting with Tools 9.2.8.3, you can add a While step to repeat certain steps while a particular condition is met based on an existing rule. See *Adding the While Step*.

- **Cross References.** A set of data relationships that map third-party values to EnterpriseOne values. For example, a serial number of a device can be cross-referenced to a JD Edwards EnterpriseOne Equipment Number when the device is used in service requests.
- **White Lists.** A white list contains an inclusive list of values that are 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 a notification. You can define a schedule by 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.
- **Attachments.** Attachments are used to automate EnterpriseOne transactions by including the options to manage text attachments, and upload or download file attachments (media objects) in transactions.
- **Logic Extensions.** Using logic extensions, you can create business logic to perform operations such as string manipulation, arithmetic calculations, conditions, loops, and even table I/O.
- **Workflow.** Using this web-based user interface, you can design and configure JD Edwards EnterpriseOne workflow processes with minimal dependency on the development client.

This graphic shows the list of the steps you can add to an orchestration. Each step in an orchestration represents one of the components (except Pause and While) listed above.

Starting with Tools Release 9.2.8.3, when you define your orchestration as Stateful, the Pause step is available in the New Step Type window.



Invoking an Orchestration

When a new orchestration is saved in the Orchestrator Studio, the name of the 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 to any EnterpriseOne application that is invoked by the orchestration. See *Orchestration Security Considerations* for more information.

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, open the design page of the component, and then click About from the Manage drop-down list. A pop-up dialog box is displayed listing 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 functionality enables you to give the component a new name and modify it as necessary, and eliminates the risk of breaking other orchestrations where the component is used.

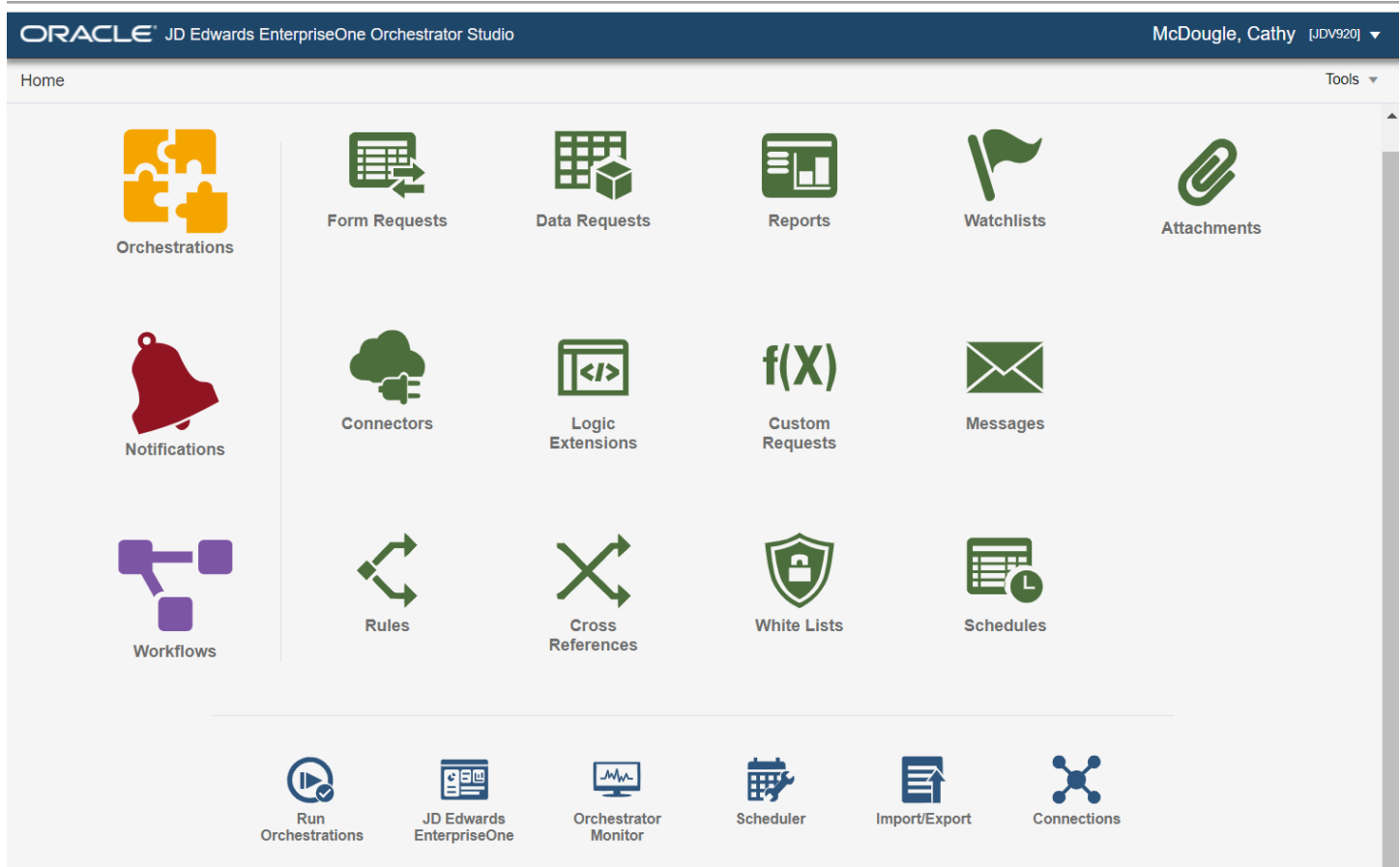
Orchestrations as User Defined Objects

All orchestration components that are 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.

Navigating the Orchestrator Studio

The Orchestrator Studio Home page displays the icons for creating orchestrations and the components that comprise an orchestration. You can hover your mouse over the components and click the info icon to view a description of the component.

The orchestration component icons on the Orchestrator Studio Home page take you to the design pages for creating and modifying each orchestration component.



The icons at the bottom of the Orchestrator Studio Home page provide links for testing orchestrations, accessing the JD Edwards EnterpriseOne web client, the Scheduler page, Orchestrator Monitor, Connections, and importing and exporting orchestration files. When you are in the orchestration or component design pages, you can access these links from the Tools drop-down menu.

When you click the Run Orchestrations, Scheduler, and Import/Export links from the Tools drop-down menu, the corresponding pages are launched in a new window. To navigate back to the parent orchestration, click the Back button on the top menu bar of the child window. (Release 9.2.4.3)

- *Testing an Orchestration*
- *Importing Orchestration Files in the Orchestrator Studio*
- *Exporting Orchestration Components from the Orchestrator Studio*
- *Orchestrator Health and Exception Monitoring*

- *Working with Scheduler*
- *Creating Connection Soft Coding Records for Connector Service Requests*

Orchestrator Studio Features

When you click any of the components icons on the Orchestrator Studio Home page, the corresponding component side panel is displayed. Locator link on the top-left of the Orchestrator Studio enables you to keep track of the components you have used to navigate to your current component. You can click the Home page link in the locator link to return to the Orchestrator Studio Home page. When you click on any component in the locator link, the associated design page is displayed.

The Orchestrations side panel contain the following features:

- **New** button. Create a new component.
- **Restore**. Updates the component list so that the changes to component information is reflected. When using the Restore button, value in the Search field value and drop-down selections are preserved.
- **Sort** drop-down list. Enables you to sort the components in the component list by name, product code, date, and category. For Connectors, you can also sort by the type of connectors such as Rest, Orchestration, Open API connectors, and so on.
- **Sort Order** button. Enables you to sort the components in ascending or descending order.
- **Group** toggle. Enables you to group and view the components in the component list by UDO status.
- **Group** filter drop-down list. Enables you to view 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.
- **Category** filter drop-down list. Enables you to search for an existing component with the category label that you have selected. (Release 9.2.4.2)

The Category drop-down list displays only the categories that you have assigned to orchestrations or orchestration components. This could be the ones that you selected from the list of category values and the ones that you entered in the Category field on the component design page. (Release 9.2.4.2)

- **Close**. Exit the side panel to access the design page.
- **Component list**. Displays a list of existing components. For each of the components, the list displays the component name, category, product code, date and time the component was created, and description. The date and time are displayed based on the user's date and time zone settings.

Note: If a component has a product code of 55, the product code value "55" is not displayed in the component list. (Release 9.2.4.2)

The Orchestrations design page contain the following features:

- **Show List**. Click to display the component side panel.
- **Name**. Name of the component.
- **Description**. Description of the component.
- **Long Description**. Click the button to provide more detail about the component.
- **Product Code**. Select a product code to associate with the component.
- **Version**. All orchestrations created in Orchestrator Studio 9.2.4 are saved as version 3 orchestrations. You cannot change this value.

- **Category.** Select a category to categorize the orchestration or the orchestration component. You can choose from the available category values or enter a new value in the Category field. (Release 9.2.4.2)

Category is useful because you can sort, filter, group, and export orchestrations by the category label.

Even if you do not have the ability to share the category values, you can still enter a new category and assign it to orchestrations or orchestration components. The category values that you have entered and not shared, do not appear in the Category drop-down list.

Note: The Category drop-down list displays only those categories that have the Category Type Code value selected as "ORCH" in the Category Manager.

- **Share.** Click the button if you want to save your category for reuse by others. (Release 9.2.4.2)

Note: You cannot share an existing category.

- **Save.** Saves the orchestration component to a status of "Personal."
- **Delete.** Deletes a "Personal" UDO.
- **Manage.**
 - **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.
 - **Restore.** Restore the component to its original state if you made a mistake and do not want to save your changes.
 - **About.** Takes you to the About page which provides the type, name, description, group, product code, and object name of the selected component. It also displays a list of the orchestrations along with the object name (UDO ID) where the component is used.
- **Canvas.** Provides a graphical representation of an orchestration with all its components. See *Working with the Graphical Representation of an Orchestration*

User Defined Object (UDO) Features in the Orchestrator Studio

Orchestration components are saved and managed as UDOs in EnterpriseOne. The Orchestrator Studio includes UDO options in the Manage drop-down menu that enable you to create orchestration components for your personal use, publish or share orchestration components, and modify shared orchestration components that are created by other users.

Note: The actions that you are allowed to perform in the Orchestrator Studio depend on the UDO security permissions that are granted to you by a system administrator. See *Setting Up UDO Security for Orchestrator Studio Users* 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* .

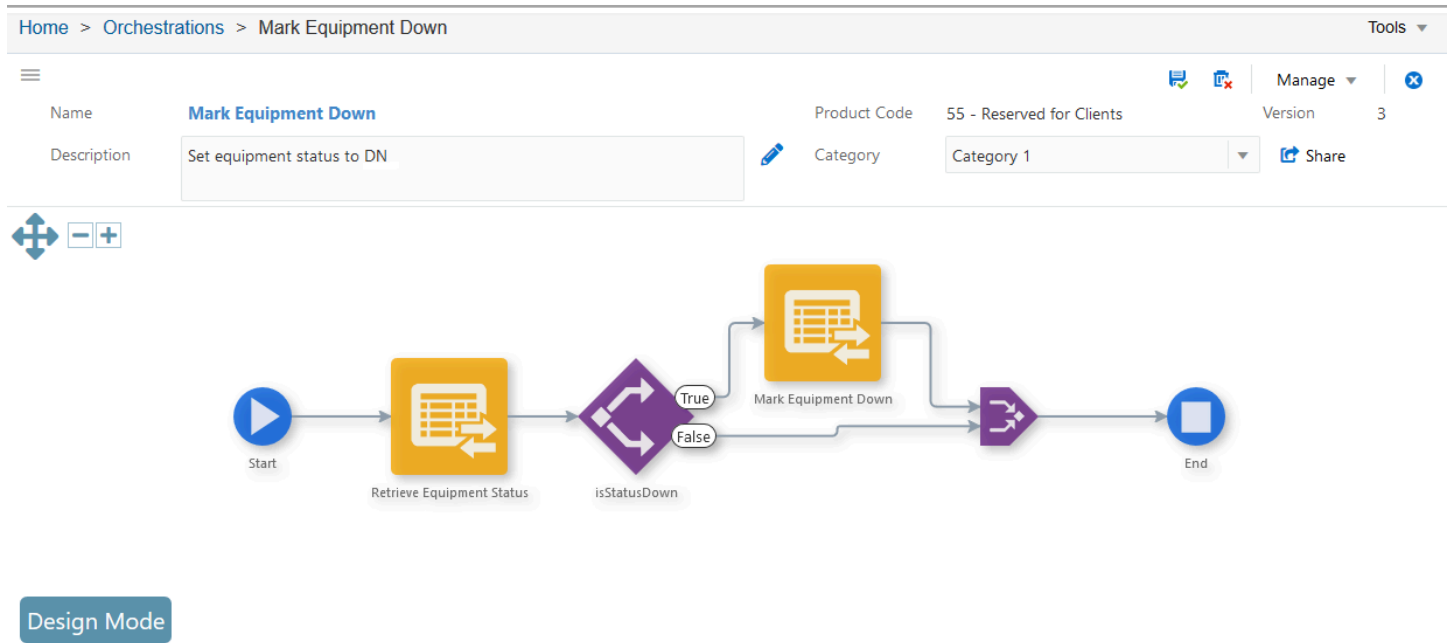
The following table describes the UDO options in the Manage drop-down list of the Orchestrator Studio design pages and the life cycle status enacted by each UDO action.

UDO Options	Description
Save As	Saves the orchestration component to a status of "Personal." Components with a status of "Personal" are components that are being developed 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 the component so that it can be shared. The component status changes to "Pending Approval" in the component list and then changes to "Shared" when the component is approved. If rejected, the 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 that you can modify it. When reserved, no other users can make changes to a UDO. The component status changes to "Reserved."
Unreserve	Cancels the reserved component. This action changes the status of the component back to "Shared."
Notes	Available when the component is in the "Pending Approval" status, this option enables you to add an additional note to send to the approver of the UDO. The Notes option is active only if a note was added the first time the UDO was sent for approval using the "Request to Publish" option. This feature enables you to add an addendum to the original note.

Working with the Graphical Representation of an Orchestration

The following figure provides a graphical representation of an orchestration with all its components. The orchestration is represented in a flowchart enabling you to easily understand the orchestration flow. Each component in the orchestration is represented by a unique icon.

Orchestrations Page



The canvas includes the following features:

- **Zoom to Fit**

The Zoom to Fit icon in the upper-left corner of the canvas enables you to view the entire graphical representation in the window. This functionality helps users to view complex orchestrations that contain multiple components.

Alternatively, you can double-click the canvas to fit the entire orchestration flow within the graphic area. This option positions the orchestration flow in the center of the canvas.

- **Zoom in or Zoom out**

The Zoom in or Zoom out icons in the canvas enable you to zoom in to get a close-up view of the orchestration or zoom out to view the orchestration in reduced size.

- **Start and End Steps**

Start and End are the pseudo steps that are not part of the orchestration but enable you to add components in between them.

When you click the Start node in the orchestration flow, action control icons are displayed at the top of the Start node.

The following figure shows the action control items that are displayed when you click the Start node.

Start Node



- **Run Options (Release 9.2.7.4).** Opens the Run Options window. You can define your Job Queue settings and orchestration monitoring requirements such as input and output information, step details, step input and output, and other run options by using this window.

Indicator Icons

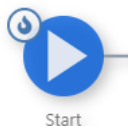
Monitoring icon

The following icon is displayed when monitoring is enabled for the orchestration:



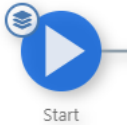
The following icons show the indicator images that are displayed on the orchestration steps representing the job queue of the orchestration:

Fire and Forget icon



The Fire and Forget icon indicates that the job runs in a fire and forget queue.

Stack icon



The Stack icon indicates that the job runs in a single-threaded queue.

Stateful icon (Release 9.2.8.3)



The pause icon indicates that the orchestration is defined as Stateful.

- o **Run Orchestration.** Takes you to the Run Orchestration page to test only the current orchestration. You can click the Run Orchestration icon on the Orchestrator Studio Home page to test all the other personal or shared orchestrations.
- o **Inputs and Values.** Opens Inputs and Values dialog box to add inputs that the orchestration consumes from a calling custom program, a third-party application, or a Cloud service.
- o **Schedule.** Takes you to the Schedule page to assign a schedule to the orchestration.

In the figure below, a Schedule icon on the Start node indicates that the orchestration has an attached schedule.

Schedule

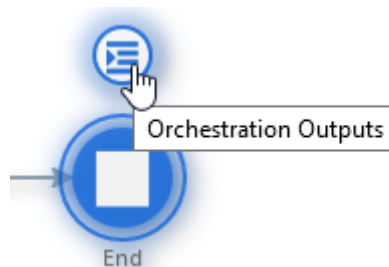


When you click the End icon in the orchestration flow, the following action control icon is displayed at the top of the End node:

- o **Orchestration Outputs.** Takes you to the Orchestration Output page to validate the fields that contain the data you want returned in the orchestration output for output mapping.

The following figure shows the action control item that are displayed when you click the End node.

End Node



- Informational hover help

Hover your mouse over a component in the graphical area to view information about the component. You can find the type of component, component name, description, and product code. Depending on the type of component, you can view other details such as match criteria for a rule and subject and body details for a message.

- Action control items

When you click a box that represents a component, the action control icons are displayed at the top of the box. You can use these action control icons to map a step, edit component, configure transformation, and define error handling.

The following figure shows the action control items that are displayed when you click the any box representing a component in an orchestration.

Action Control icons



- **Choose Step.** Takes you to the component side panel dialog box to choose the component that you want to add to the orchestration flow. The component side panel dialog box displays only those items that are related to the component that you have chosen to include. You can also create a new component item from this dialog box. Broken mapping is indicated by



icon on the component box.

- **Edit.** Takes you to the design page for modifying a particular component. Alternatively, you can double-click the box representing a component to access the design page for editing the component.

Note: The Edit icon is disabled if you have added a local orchestration directly as a step in the orchestration flow. (Release 9.2.4.3)

- **Transformations.** Map orchestration inputs to inputs that are defined in each orchestration component.

The following example shows the Iterate Over icon on the box that represents a component in the orchestration. The Iterate Over icon indicates that you want to pass a data set that is 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.

Iterate and Fire and Forget icons



This fire and forget indicator applies to notification and orchestration steps only and indicates that the individual steps execute asynchronously (on their own thread). If both the iterate and fire and forget options are selected for a step, then the step is submitted to its own queue with the maximum number of threads configured for it.

- o **Error Handling.** Set up error handling for each orchestration component.

Broken mapping is indicated by



icon on the component box. A broken mapping can exist in Transformations and Error Handling. When you open the Transformations and the Error Handling dialog boxes, the Broken Mapping icon is displayed against the rows where the break exists.

- Design Mode
 - o **Add icon.** Click the Design Mode button to add components to the orchestration by clicking the plus sign (+) in the orchestration path.

Design Mode



Alternatively, you can double-click the arrow in the orchestration path to add a component to the orchestration.

- o **Remove Step.** Removes the component from the orchestration.
- o **Cut Element.** Move component within an orchestration. You can paste the component at any location within the orchestration by clicking the Paste Element icon displayed in the orchestration path.

Creating Service Requests

This section contains the following topics:

- [Understanding Service Requests](#)
- [Creating a Component](#)

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 retrieve 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 to EnterpriseOne users through the EnterpriseOne Work Center.

- Connector

Use a connector request to invoke an orchestration, a notification, a REST service or a database. A connector can invoke an orchestration on 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.

Note: With Tools release 9.2.4.3, you do not have to create a connector service request to invoke "local" orchestrations or notifications. You can invoke your own orchestrations and notifications directly as a step in the calling orchestration.

- Custom Request

Use custom Java or Groovy to execute a custom process. With Tools release 9.2.4.3, you can add a custom request to call a business function.

- 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.

Creating a Component

You can use the Orchestrator Studio to create service requests such as form requests, data requests, connectors, and so on along with other components such as connections, white list, rules.

1. On the Orchestrator Studio Home page, click the icon for the type of component that you want to create:

Form Requests

Data Requests

Reports

Watchlists

Connectors

Connections

Custom Requests

Messages

Rules

Cross References

White Lists

Schedules

Attachments

Logic Extensions

The Orchestrator Studio displays the component side panel.

2. On the component side panel, click **New**.

Alternatively, access the Orchestration in design mode and click the plus icon(+) to add a component to an orchestration. Select the component type that you want to create from the New Step Type list and then click the New button.

3. On the component design page, complete these fields:
 - a. **Name.** Enter a name for the component. Do *not* include special characters in the name.
 - b. **Short description** field. Enter a description with a maximum of 200 characters. This description will appear below the component name in the component list.
 - c. (Optional) Click **Long Description** to provide additional details about the purpose of the components.
 - d. **Product Code.** Click the Product Code drop-down list to select a product code to associate with the component. Adding a product code enables the administrator to manage UDO security for orchestration components by product code.
4. With Release 9.2.4.2, you can enter a category for the orchestrations or the orchestration components. The Category drop-down list displays the orchestrator categories if any exist. You can choose from the available categories or enter a new value in the Category field.

Using the Category feature, you can sort, filter, group, and export orchestrations by a category label.

Note: To obtain the Category feature, you must download the Tools 9.2.3.3 electronic software update (ESU).

Note:

- The Orchestrator Studio supports only those categories that have the Category Type Code value as "ORCH" in the Category Manager.
- As Connections are not user defined objects, you cannot add Categories for Orchestrator Connections.

5. Click the Share button if you want to save your orchestrator category for reuse by others.

After you click the Share button, a record is created in the Category Manager with the Category Type Code value as "ORCH".

Note: The Share button is available only if an administrator has given you the permission to add values in the Category Manager application (P980058). If you cannot access the Share button, you can only use the existing orchestrator categories. For more information on Category Manager, see *Working with Categories* in this guide.

6. Click **Save**.

A new component is saved, for the first time as a "Personal" UDO. After configuring the component, you can use the UDO options that are described in the *User Defined Object (UDO) Features in the Orchestrator Studio* section to move the component to the appropriate status.

Starting with Tools Release 9.2.5.4, when you edit a UDO component that can be added to an orchestration, you will see the **Create Orchestration** option in the Manage drop-down menu.

When you click this option, the system creates a new orchestration that contains only the UDO component that you edited. The name, description, product code, and category fields are automatically populated with the same values as the UDO component. The system automatically creates and maps the UDO component's input as the orchestration input, and it returns the UDO component's output as the orchestration output. The new orchestration will then open in the orchestration editor, and you have the opportunity to edit or rename the orchestration before saving it. If you exit the orchestration editor without saving, the orchestration will be discarded. Once you click **Save** to save the orchestration, the orchestration is ready to run. This feature enables you to test a new orchestration component soon after you create it, and you can rapidly promote a standalone orchestration step into a complete and runnable orchestration.

Working with Orchestrator Categories (Release 9.2.4.2)

The Category Manager (P980058) application enables you to view, add, edit, and delete orchestrator categories. You can attach these categories to orchestrations and the orchestration components to group orchestrations with a common purpose or business goal.

Orchestrator categories are those records that have the Category Type Code value selected as "ORCH" in the Category Manager.

The orchestrator categories created in Category Manager appear in a drop-down list in the orchestration component design page and are available for use. In the component design page, you can also create orchestrator categories on the fly and share them for reuse by other users. The Category drop-down list displays only those categories that have Category Type Code as "ORCH". For more information on adding categories in the component design page, see [Creating a Component](#).

Note: Use action security to disable a user's ability to add categories in both the Category Manager and the orchestration component design page. If you disable the ability to add categories in Category Manager, the Share button will not appear in the component design page.

Working with Categories

Use Category Manager to view, add, update, or delete orchestrator categories. Category Manager can be accessed from the navigator menu by selecting EnterpriseOne menus, EnterpriseOne Life Cycle Tools, and then Orchestrator Management.

Note: Categories will be available only if you have downloaded the Tools 9.2.3.3 ESU, and table and application are available.

Adding a Category

To add a category:

1. In Category Manager, select Add.
2. In the grid, provide values for the following columns in the first row:
 - o Category. Required. Enter a description of the category. This field is case-sensitive.
 - o Category Type. Required. For orchestrations and its components, use ORCH.
 - o Language Code. This is an optional field.

Repeat these steps to add multiple categories.

3. Click OK.

Updating a Category

To update a category:

1. In Category Manager, use the header fields or QBE line to find the category you want to update.
2. Select Find.
3. Place a checkmark in the row header of the category that you want to update and click Select.

4. Make the updates and click OK.

Deleting a Category

To delete a category:

1. In Category Manager, use the header fields or QBE line to find the category you want to update.
2. Select Find.
3. Select the row for the category that you want to delete.
4. Select Delete and click OK to confirm that you want to delete the selected item.

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 Component*, 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)*
- *Creating a Form Request with the JD Edwards EnterpriseOne Orchestrator Process Recorder*

Note: 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*.

Loading EnterpriseOne Application Form Fields and Controls

1. Click the **First** button and then complete the following fields in the pop-up window:

- **Application.** Enter the application ID.
- **Form.** Click the drop-down list to select the form.
- **Version.** Enter the version ID.

If you leave the Version field blank, the Orchestrator will use the default version while executing the form request.

Leaving the version field blank with the Application Stack option selected enables you 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.

You can define a version in the form request and still override the version. When adding a form request to an orchestration, you can select the version variable that is available in the list of transformations. This is applicable to form requests with the Application Stack option selected.

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

- **Form Mode.** 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 may be dependent on the form mode as specified in the 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.

- **Bypass Form Processing in Studio.** 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 action on entry or to hide certain fields. These actions can cause the load to fail or prevent certain fields from loading in the Available Actions grid.
- **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 that are defined in a form request. The queries that you can see in this list are based on UDO security permissions.
- **Stop on Warning.** Enable this option if you want the processing to stop if the invoked EnterpriseOne form generates a "Warning" message. Keep this option disabled if you want processing to continue after a warning message.

Prior to Tools Release 9.2.4.3, when running a form service request for some applications such as P48013 you had to press the OK button twice to process the OK-Button Clicked event a second time if you encountered a warning message.

As of Tools Release 9.2.5.4, when the Stop on Warning option is disabled in a form service request, the system automatically runs the OK-Button Clicked event a second time to more closely follow the flow that

occurs when the application is run through the web client. Because of this change, you are not required to press the OK button the second time for applications such as P48013.

- **Run Form Service Request Event.**

This option enables you to configure whether the Form Service Request Event runs when the form is executed from an orchestration. Form Service Request Event is a form design aid (FDA) event that is executed only when an application is run through a form service request from the AIS server.

- **Exclude Summary Row (Release 9.2.6.2).**

This option gives you the control over whether to include the summary row of a grid as part of the output of a form request.

Many EnterpriseOne forms contain grids with a summary row that provides totals for certain columns. For example, Sales Order Detail provides a summary row that totals the Extended Amount for all the orders in the grid. For an end user that summary row can provide some very useful information, but when Orchestrator reads the grid, the summary row might be an unwanted extra row. For example, when writing an orchestration to return the count of order lines in the grid, including the summary row would result in an incorrect count.

- **Turbo Mode.** 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.

2. Click the **Save and Load** button.

The Orchestrator Studio loads the controls and the fields in the grid. The name of the form is displayed above the Order of Execution grid.

3. Click the Form Request Options button and 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 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. The other considerations when selecting values to be returned from forms in an Application Stack are described in *Configuring Form Request Actions*.

- **Return Data From All Forms.** Enabled by default. This option specifies whether to return data from all the forms or just the last form.
- **Run Synchronously.** Enabled by default, this ensures that the form request completes all actions before the Orchestrator executes the next step in the orchestration. This option 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.

You can access more options for configuration. You can configure settings that control how the AIS Server processes a form request in an orchestration at runtime. These options determine how the

form request will be processed. See *Configuring Form Request and Data Request Processing* for more information.

The Orchestrator Studio loads the controls and the fields in the Available Actions grid. The name of the form is displayed above the Order of Execution 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. You can use the Add Form controls such as First, Before, After, and Last to place the additional application forms according your requirement.

You can remove the form by clicking the **Delete Form** icon (X sign).

Configuring Form Request Actions

The Available Actions area is where you specify the EnterpriseOne fields and controls that are used to perform the desired business transaction in EnterpriseOne.

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 the action to the Order of Execution area.

- **Description** (informational only)

This column displays the controls and the fields for each form in a collapsible or expandable parent that is 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.

- **ID** (informational only)

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

- **Return**

Enable Return 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 that are returned from EnterpriseOne applications.

When you select a value to be returned, a Return Form Data action is added to the order of execution. Ensure that this action follows the steps that are needed to populate the returned fields in the form, and make sure that this action occurs before an action that clears the data or moves to another page.

Only one Return action is allowed per form when configuring a form request in the Orchestrator Studio. If you need the orchestration 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 values 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.

Starting with Release 9.2.5, you can enable the **Return** toggle for the Include Count field in Form request to include the row count as a variable. You can also enable the **Return** toggle for the Include Count field to count the number of records in the grid.

Starting with Tools Release 9.2.8.2, you can reorder the outputs of a form request by clicking the **Edit Returns** icon on the Return Form Data action and by dragging and dropping the rows. The system preserves the order that you set when you **Save** the changes and reflects this order in Orchestration Outputs when the Form

Request is added to an orchestration. This feature improves readability and enables orchestrations to easily integrate with user interfaces or third-party systems that might be sensitive to the order of outputs.

Note: Order of name-value pairs in JSON is not important or guaranteed but in most cases, the orchestration output reflects the order set at design time. If you use Output Array to File to create a CSV, XML, or JSON file, the system reflects the order of grid data specified at design time.

- **Variable**

In each row, the Variable column is editable only if Return is enabled. Use this column to enter a variable name for the return value. When you add the form request to an orchestration, the variable name appears in the orchestration inputs grid. As a result, the returned value is available for mapping to the subsequent steps in the orchestration.

You can also use this field to retrieve a data set from an application grid. See *Retrieving and Passing Data Sets in an Orchestration* for more information.

- **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 the 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 Row** (row)

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

In the Order of Execution table, you can enter a variable in the Input column of this row. This action enables you to map an orchestration input to this variable. Alternatively, you can enter a value in the Default Value column to select a specific row. To select the first row in the grid, add Select Row and leave these columns blank or set the Default Value to 1 when adding this action to the Order of Execution.

(Release 9.2.5.2) You can enter a comma delimited list of row numbers such as 1,2,4 as value for Select Row. In the previous releases it was possible to pass a single row number only. You can enter the row numbers as a Default Value or pass them into the Form Request through the variable defined for the Select Row action.

If the grid has columns that do not have a QBE, you will see the **Filter** check box in the Description column when you add a **Select Row** action or add a **Row Number for Update** action.

If you select this check box, a Filter button appears in the Input column in place of the variable name and the Default Value field is cleared and disabled.

When you click the Filter button, a popup window is displayed showing all the columns in the grid that do not have a QBE.

Note: To filter columns with a QBE, continue to use the SetQBEValue action.

You can then enter the filter criteria on one or more columns to find the rows you want to select or to update.

This enhancement eliminates the need to use custom logic to update rows in a grid with no QBE.

For example, if you are on a previous release, to update all the sales order lines on an order that are at a certain status, you have to fetch and return the sales order lines as a step. You then have to pass the output variable from that step to a custom step that will parse the .output response and apply filters through custom looping

logic to find the rows that have to be updated and build an array. Finally, you have to run Sales Order Entry using the array to update the appropriate rows.

As of Tools Release 9.2.5.2, all these steps can be completed using the **Filter** option in the description column.

- **Row Number for Update** (row)

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

(Release 9.2.5.2) The **Filter** option as described in the **Select Row** section is available for columns from the grid that do not have a QBE.

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

Configuring the Order of Execution

After actions are added to the Order of Execution grid, you can reorder them by dragging and dropping the rows. You can also delete actions using the Delete (X) button.

You can enter or modify values in the Input and Default Value columns in the Order of Execution area. Use the following features to configure the order in which the actions have to be executed.

- **Input**

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

Instead of a mapped variable, you can enter a literal value in the Default Value column. You can also 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.

You can also use this column 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 default value. You can also add a default value to use if the mapped value returns a null.

For a check box or a radio button, you can enable (slide the button to right) 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 check box and radio button, you can set an input variable in addition to the default value.

For a combo box control (a list of items in a drop-down menu), the Orchestrator Studio displays a drop-down menu in the Default Value column in which you can select the appropriate item from the menu.

Starting with Tools Release 9.2.6.3, you can select **<Space>** from the Default Array drop-down menu to clear out the fields in the form request.

- **Edit Returns** (left arrow icon in the Action column)

Use this feature to view and edit all the return values. You can also use this feature to retrieve data from "hidden" fields or grid columns that are not displayed in the Available Actions area. Hidden fields appear only after 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 the hidden fields and the grid columns. The control ID (AIS Id) is displayed in the Advanced Options section in the help pop-up window.

To edit the existing return values and retrieve a hidden field:

- a. In the Order of Execution area, in the Return Form Data row, click the left arrow in the Action column.

A dialog box appears displaying the control IDs, description, and associated variable names of any return field that is already selected in the Available Actions area. You can add a variable name in the Value column for an existing returned value and also remove a returned value.

- b. Enter the control ID of the hidden field in the ID field. Optionally, you can enter a variable name to represent the return value in the associated Variable field.

For multiple controls, use the Add button to add more controls, making sure that the order of the values matches in each field.

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

- c. Click OK.

Populating Multiple Grids with Repeating Inputs (Optional)

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

When you add mappings to a grid to the Form Request, a grid level grouping is created with a default value of `GridData` in Input column of the Order of Execution table. This value is the array name that 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 for each grid to distinguish the grids.

Creating a Form Request with the JD Edwards EnterpriseOne Orchestrator Process Recorder

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.

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.

Starting with Tools Release 9.2.4, if an application contains versions, the Process Recorder saves the versions and displays it against the form in a Form Request.

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. You can select return values from any form while recording the form request.

Starting with Tools Release 9.2.5, the Process Recorder contains an option to record the assertion values.

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 can click the Refresh Cache button on the Run Orchestrations page to access the form requests.

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.

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* .

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. (Release 9.2.6) Enable the **Stop on Warning** option if you want the processing to stop if the invoked EnterpriseOne form generates a "Warning" message. Keep this option disabled if you want processing to continue after a warning message. By leaving this option disabled, you can avoid recording the second press of the OK button when a warning is encountered during the recording.
4. 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.

5. 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 and click on control, 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.

Note: 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.

6. 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.

7. If you want to record the assertion values in the Process Recorder, enable the **Capture Assertions** option. (Release 9.2.5)
 - a. Click **Return Values**.
 - b. Enter value for the control and then click on controls in the form that you want returned for the controls. You can see that the values are recorded in the **Value** field of the Return Controls section. If you add a control by mistake, click it again to remove it from the list.
 - c. Click the **Operator** drop-down menu and select the operator value as required in the Return Controls section.
 - d. Similarly, click the grid headers or the QBE (Query by Example) controls of the grid and then enter or select the values that you want returned for the columns in the grid. You can see that the values are

recorded in the **Value** field of the Return Columns section. If you add a column by mistake, click it again to remove it from the list

Note: If you click the grid controls and add the columns for the Return Columns section in the process recorder, the values of the first column are added as assertion values by default. You can change these default values. You can also select different rows in the grid to add specific values. You can see the row number of the selected row when you view this form request from the Orchestrator Studio.

- e. Click the **Operator** drop-down menu and select the operator value as required in the Return Columns section.
 - f. 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.
8. 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**.
 9. To complete the recording, enter a name, product code, and description for the form request.
 10. 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](#).

Configuring a Data Request

This section contains the following topics:

- [Understanding Data Request](#)
- [Configuring a Data Request to Return Field Data](#)
- [Configuring a Data Request with Data Aggregation](#)
- [Viewing and Copying JSON Code of a Data Request](#)
- [Configuring Form Request and Data Request Processing](#)

Understanding 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 a business view and return aggregate amounts.

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

When you add a data request to an orchestration, all the defined fields are automatically added to the orchestration as inputs. As a result, you can map the returned data to subsequent steps in an orchestration.

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.

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, and the business analyst can then map these inputs to the subsequent orchestration steps.

- In order to have the returned values be available in subsequent steps of an orchestration, the Variable column for each of the return value in the Return Fields and Variable Names grid must be populated. Otherwise, the return value is just available for mapping in the orchestration outputs, but not available as an input.

Starting with Release 9.2.5, you can enable the **Return** toggle for the Include Count field in Data request to include the row count as a variable. You can also enable the **Return** toggle for the Include Count field to count the number of records in the grid.

To configure a data request to return field data:

1. On the Orchestrator Studio Home page, click the **Data Requests** icon.
2. Create a data request as described in [Creating a Component](#).
3. On the Data Request design page, in the **Table/View Name** field, enter the name of the table or the business view.

If you do not know the table or business view name, but you know the application that uses the table or business view, click the Get View from Form button.

In the dialog box that is displayed, enter the application ID and select the form. The associated business view is then retrieved and displayed in the Table/View Name grid.

4. Click the **Load** button.

The Orchestrator Studio loads all the 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.

5. Define the filtering criteria for the data request:
 - a. In the grid on the left, click the **Filter** icon next to the field or fields that contain data on which you want to filter on.
The Orchestrator Studio displays each field in the Filter Criteria area on the right.
 - b. In the **Filter Criteria** section, for each field in the Operator box, select the appropriate operand and in the adjacent field, enter a literal value or a variable.
 1. A variable appears in the field by default. If you modify the variable name, make sure that the syntax includes the \$ sign and braces, for example:

`${Address Number 1}`

If you do not enter a variable using `${}`, the literal value that you enter will be used directly in the filter criteria.

2. If the field is a date, you can use the drop-down arrow at the end of the row to set a special value, such as today plus or minus the number of days, months, or years that you specify.

If you want to add multiple values for date, select the "is in list" operator and click the List button in the Values column. Then click Add and set the value.

- 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 Name 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 that are defined in a data request. The queries that you can see in this list are based on UDO security permissions.

- d. (Release 9.2.5.4) Enable the **Include Empty Query Values** option if you want to include the empty values in the query when the data request runs.

For the existing data requests, this option is enabled by default and therefore all the defined fields for the query are always included even if they are empty. If you disable this option, any empty query values at runtime will not be included in the query (potentially resulting in more records returned).

6. Specify the data you want returned:

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

The Orchestrator Studio adds each field to the Return Fields and Variable Names section 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 literal 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.

(Release 9.2.6) Enable the **Associated Description** option if you want to return the description associated with the variable. If you enable this option, the associated description is available as an orchestration variable in subsequent orchestration steps and for inclusion in the orchestration output.

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.

7. 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.

You can use 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*.

Configuring a Data Request with Data Aggregation

To configure a data request to return aggregate amounts:

1. On the Orchestrator Studio Home page, click the **Data Requests** icon.
2. Create a data request as described in [Creating a Component](#)
3. On the **Data Request** design page, in the **Table/View Name** field, enter the name of the table or business view.

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 the fields from the primary business view that is associated with the form.

4. Click the **Load** button.

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

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

This step enables the aggregation features in the grid to the left.

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.

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

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

7. Set up conditions for filtering data:

- a. For each field in the **Filter Criteria** area, select the appropriate operand and in the adjacent field, enter a literal 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 braces in the syntax, for example:

```
#{Address Number 1}
```

If the field is a date, you can use the drop-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 Name 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 that are defined in a data request. The queries that you can see in this list are based on UDO security permissions.

- c. (Release 9.2.5.4) Enable the **Include Empty Query Values** option if you want to include the empty values in the query when the data request runs.

For the existing data requests, this option is enabled by default and therefore all the defined fields for the query are always included even if they are empty. If you disable this option, any empty query values at runtime will not be included in the query (potentially resulting in more records returned).

8. Click the **Aggregate** icon next to the fields that contain the data for the aggregation.
9. In the pop-up window, select the type of aggregation that you want to perform.

The aggregation options that are displayed depend on whether the field contains a numeric value or a string.

10. (Optional) In the **Aggregations** section, slide the toggle button to right if you want the response to include a count of the records that are returned. To use the returned count in a subsequent orchestration step, enter a variable name in the adjacent field.
11. (Optional) Use the following features in the grid to the left to further define the data aggregation:

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

The Orchestrator Studio 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 default value or variable.
- o **Group By** icon. Click this icon next to any field that you want the aggregate results grouped by. In the Group By section on the right, you can enter a variable name.

Starting with Tools release 9.2.3.4, if you are grouping data by a date field, you can configure the format of the date to be displayed in the output. In the Date Format drop-down field, you can either enter a simple date format or select any of the following formats:

- Date - Milliseconds: Displays time in milliseconds.
- Date - User's Format: Displays the EnterpriseOne user's preferred date format.
- Date - Calendar Quarter: Uses the four-digit year and one-digit quarter format, for example 2020-1 for Q1-2020.
- Date - Year: Uses the four-digit year format.
- Date - Year-Month: Uses the four-digit year and two-digit month format.
- Date - MM/dd/yyyy
- Date - dd/MM/yyyy
- Date - yyyy/MM/dd

For more information on SimpleDateFormat, refer to the following Java documentation:

<https://docs.oracle.com/javase/8/docs/api/java/text/SimpleDateFormat.html>

Using the Group By option may result in multiple rows being returned, one for each unique combination of a group. For example, you might want the system to return the 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: When Data Set Variable Name is not used, the variables are only supported in the first row of a response. If Data Set Variable Name is populated, these variables are used to reference the individual columns within the array. If a query contains multiple rows, only the values from the first row will be available as variables.

(Release 9.2.6) Enable the **Associated Description** option if you want to return the description associated with the variable. If you enable this option, the associated description is available as an orchestration variable in subsequent orchestration steps and for inclusion in the orchestration output.

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.

- o **Order By** icon. For any field that is used for aggregation of return data or for group by action on return data, click this icon to arrange return data in ascending or descending order.

Starting with Tools Release 9.2.8.2, you can reorder all the selected columns in each section of the data request, including the Returns and Aggregations, by dragging and dropping the rows. The system preserves the order that you set when you **Save** the changes and reflects this order in Orchestration Outputs when the Data Request is added to an orchestration. This feature improves readability and enables orchestrations to easily integrate with user interfaces or third-party systems that might be sensitive to the order of outputs.

Note: Order of name-value pairs in JSON is not important or guaranteed but in most cases, the orchestration output reflects the order set at design time. If you use Output Array to File to create a CSV, XML, or JSON file, the system reflects the order of grid data specified at design time.

You can use the Options button to configure the settings that control how the AIS Server processes a data request at runtime. See [Configuring Form Request and Data Request Processing](#).

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.

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 a data request in an orchestration at runtime.

For form requests, these settings are available from the Form Request Options button on the Form Requests design page.

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

Page Size Considerations

You can use page size for data services and form services (application stack) to indicate the number of records that the AIS Server fetches at a time through the HTML Server. Using this functionality may help when the data or form request expects a large data set. The data request or the form request, instead of requesting all of the records in a single operation, can request a chunk at a time, store (buffer) the record on the AIS Server, and then return the complete set of records in the response to the caller.

For a form service, it is important to consider where to place the "Return Form Data" action when you want the orchestration to return grid data. If your application stack contains multiple forms with grids, and you want to use the paging option to return all the data for a grid, you must place the "Return Form Data" action after the Find action.

Note: It is unnecessary to use the paging setting if you do not expect to get large numbers of records or have not had any issues in getting large record sets. While this setting helps with the large data sets, it may slow down requests for smaller data sets. A request that works fine to return 1000 records in one response would run much slower if the page size is set to 100. The slowdown is because the request would be executing ten times the number of AIS to HTML server communications.

Processing Settings

The settings include:

- **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 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. The other considerations when selecting values to return from forms in an Application Stack, are described in *Configuring Form Request Actions*.

- **Return Data From All Forms.** Enabled by default if Application Stack is not enabled. This option specifies whether to return data from all forms or just the last form.
- **Run Synchronously.** Enabled by default, this option ensures that the form request completes all actions before the Orchestrator executes the next step in the orchestration. This option 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.

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

- **Maximum Records.** 0 is for All Records.
- **Maximum Records Variable (Release 9.2.8.2).** Disabled by default, this option enables the form requests and data requests to be able to accept a variable number of rows to be returned. For example, you could create an orchestration to return a list of all sales orders at a certain status, and provide an input variable for the number of rows to be returned. An application that invokes the orchestration, such as a mobile application, could then allow the user to specify the number of rows to return.
- **Page Size.** 0 is for Disabled. For a form request, page size is only available if the form request is defined as an application stack.

As of release 9.2.5, the value of Page Size is set to 1000 and hence the disk caching is used by default.

The page size value works directly with the max records value:

Max Records	Page Size	Behavior
0 - All records	0 - Paging is off	System will attempt to get all records in one request.
0 - All records	>0 - Size of page defined	System will attempt to get records in sections for the defined page size until it gets all records.
>0 - Set number of records	0 - Paging is off	System will attempt to get the max number of records defined in one request.

Max Records	Page Size	Behavior
>0 - Set number of records	>0 - Size of page defined	System will attempt to get records in sections for the defined page size until it gets to the maximum number of records defined.

- **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 that are 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. The default format is Grid Data output type for version 2 orchestrations.
 - **Grid Data.** Returns grid data in simplified name-value pairs.
 - **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.
- **Stop on Warning** check box (form requests only). Select this check box if you want the processing to stop if the invoked EnterpriseOne form generates a "Warning" message. Do not select 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.** Select this check box to enable caching of the service request response. If the Enable Caching by Default option is enabled in the Server Manager, this parameter is ignored and all responses for Read operations is cached.
- **Caching: Force Update.** Select 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 the Server Manager, then this parameter for the individual service request is ignored.
- **Caching: Time Stored - Milliseconds.** Enter the 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 the Server Manager.

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 that are defined for the report in the Batch Versions program. Alternatively, you can configure a report service request 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 report service request to use the default settings or override the settings.

A report service request can launch EnterpriseOne reports that are designed with report interconnects. Report interconnects are inputs that are added to a report at design time. These inputs 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 toggle 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 who is initiating the orchestration must be authorized in the Security Workbench to perform these overrides as well.

Starting with Tools Release 9.2.6.3, you can use an orchestration to run a report (or to locate an existing report) and save the output files during the orchestration execution. The report output files can then be delivered in the response of the orchestration itself.

You can configure a message service request to :

- Execute a report
- Download an existing report

Executing a Report

To configure a report service request to execute a report:

1. On the Orchestrator Studio Home page, click the **Reports** icon.
2. Create and name the report service request as described in *Creating a Component*.
3. Select the **Execute Report** option. This option is selected by default for new reports.
4. On the Reports design page, in the **Report Name** field, enter the ID of the report that you want the report service request to invoke.

The Orchestrator Studio loads all the versions that are associated with the report in the Report Version drop-down list.
5. Click the **Report 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.
6. (Release 9.2.5.2) Enable the **Variable** toggle if you want to use the report version as a variable. When you enable the Variable toggle, the name of the variable is populated with the value `<Report Request UDO Name>_version` by default.

For example, for a report that has many versions, you can build a single report request and pass a version into that step from an orchestration input, or even from a value that is derived in a previous orchestration step. Using this option, a single report step can serve to run any version of the report, giving your orchestrations more flexibility.
7. (Release 9.2.8.2) Click the **Queue** drop-down list and select a job queue for the report. If you want to set a variable for the queue, you can use the standard `${variableName}` syntax. This variable is listed in step transformations and can be mapped to an orchestration input, the output of a previous step, or a string literal. During runtime, the report is run on this specific queue based on the value provided in the Queue.
8. Enable the **Fire and Forget** option if you want the report to run asynchronously.

Note: The Download Report section is disabled if you enable the Fire and Forget option.

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

Disable the **Fire and Forget** option if you want the report to run synchronously. When a report runs synchronously, the orchestration will wait until the report is done before continuing to the next step. There are two AIS configuration settings to control how long the orchestrator will wait and how often it will check during the wait time for the report to complete.

- o Check Status Interval (milliseconds) - Default Value 3000 (3 seconds)
- o Launch Report Maximum Synchronous Wait Time (milliseconds) - Default Value 1800000 (30 minutes)

You have to consider these settings especially when running a lengthy report synchronously.

9. Select one of the following options to determine how to run the report:
 - o **Blind Execution.** Select this option if you want the report service request to invoke the batch version by using the settings that are defined in EnterpriseOne for the report version.
You can review the settings that are defined for the version by expanding the Data Selection, Data Sequencing, and Processing Options sections.
 - o **Report Interconnects.** If a report is defined with report interconnects, select this option and enter the values you want to pass in the available fields in the report. This option is disabled if no report interconnects were defined for the report.
 - o **Override Version.** Select this option to override the settings in EnterpriseOne for a version of the report, and then perform these steps:
 - i. Expand **Data Selection** and **Data Sequencing** sections to modify existing criteria, or click the Add button to define new criteria.
 - ii. Expand the **Processing Options** section and change the settings as desired.
10. (Release 9.2.6.3) Expand the Download Report section and complete these fields:

Note: The Fire and Forget option must be disabled to access the Download Report section.

- a. **Download Immediately:** Enable this option if you want to download the report immediately after it runs.
- b. **File Type:** Select the file type you want from this drop-down menu.
The available file types are:
 - PDF
 - CSV
 - OSA (Release 9.2.8)
 - ETEXT - BI Publisher Only
 - HTML - BI Publisher Only
 - EXCEL - BI Publisher Only
 - XML - BI Publisher Only
 - RTF- BI Publisher Only
 - PPT - BI Publisher Only
- c. **Overwrite Existing File:** Enable this option to use the original file name throughout the processing. Disable this option to assign a unique file name to each file.
- d. **Keep Temp Files:** Enable this option to save files in the root temporary directory. The files will remain in the root temporary directory indefinitely. Disable this option to save the files in the user's session temporary directory. These files will be deleted when the user session ends.

- e. **File Name Variable:** Enter the file name variable. The downloaded file's name will be stored with this file name variable, which can be used in later steps to access the file in the temporary directory..
- 11. 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 that is defined in the Enterprise Server configuration settings.
- 12. To modify the print properties that are 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* .

- 13. 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 higher value for receiving 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).

Downloading an Existing Report (Release 9.2.6.3)

To configure a report service request to download an existing report:

1. On the Orchestrator Studio Home page, click the **Reports** icon.
2. Create and name the report service request as described in *Creating a Component*.
3. Select the **Download Existing Report** option and complete these fields:
 - a. **Job Number:** Enter the job number variable.
 - b. **Execution Server:** Enter the execution server name.
 - c. **File Type:** Select the required file type from the drop-down menu.

The available file types are:

- PDF
 - CSV
 - OSA (Release 9.2.8)
 - ETEXT - BI Publisher Only
 - HTML - BI Publisher Only
 - EXCEL - BI Publisher Only
 - XML - BI Publisher Only
 - RTF - BI Publisher Only
 - PPT - BI Publisher Only
- d. **Overwrite Existing File:** Enable this option to use the original file name throughout the processing. Disable this option to assign a unique file name to each file.
 - e. **Keep Temp Files:** Enable this option to save files in the root temporary directory. The files will remain in the root temporary directory indefinitely. Disable this option to save the files in the user's session temporary directory. These files will be deleted when the user session ends.
 - f. **File Name Variable:** Enter the file name variable. The downloaded file's name will be stored with this file name variable, which can be used in later steps to access the file in the temporary directory.
4. Click **Save** to save your changes.

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 the [JD Edwards EnterpriseOne Applications One View Watchlists Implementation Guide](#) .

1. On the Orchestrator Studio Home page, click the **Watchlists** icon.
2. Create and name the Watchlist service request as described in [Creating a Component](#).
On clicking the **New** button, the grid displays all the 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.
3. Search and select the Watchlist that you want this service request to access.
On the Watchlists design page, the **Watchlist Information** grid displays information about the Watchlist. To select a different Watchlist, click the **Select Watchlist** button to return to the list of Watchlists.
4. In the **Outputs** section, select the Watchlist data that you want returned.
5. In the **Advanced Outputs** section, select any additional details about the Watchlist that you want returned.
6. Click the **Force Watchlist To Update** check box if you want the service request to refresh the Watchlist data in EnterpriseOne before returning the Watchlist data. (Recommended)

Configuring a Message Request

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

- Workflow distribution lists that can be used 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.
- Links to EnterpriseOne applications.
- Links to EnterpriseOne reports and BI Publisher reports.
- Links to URLs.
- Links to notifications and orchestrations.

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 to a message request. You can include variables in the recipient fields, the subject and body fields, a message template, and in links.

Starting with Tools Release 9.2.6.3, you can use the message editor to create a message body to include rich formatting of text enabling you to use typefaces, emphasis, and colors of your choice. You can create layouts for messages that include links and images that are interspersed with text. You can also include tables and formatted content from external sources such as Microsoft Word.

To configure a message request:

1. On the Orchestrator Studio Home page, click the **Message** icon.

2. Create and name the message request as described in *Creating a Component*.
3. On the Message design page, to enter recipients (To, Cc, or Bcc), select a recipient type:
 - o Select **Address Book** to send a message to a single EnterpriseOne user. Enter the address book number of the EnterpriseOne user. The corresponding address book description is displayed next to the Address Book field.
 - o Select **Contact** to send a message to an individual in a user's contact list. Enter the address book number of the user and then the number of the contact.

When you enter a valid address book number of the user and the contact number, the corresponding description of the contact is displayed next to the Contact Number field.

- o 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 the structure type of the distribution list.

Depending on the parent address number that you enter, the customized structure type variables: `${tvariable}`, `${ccvariable}`, or `${bccvariable}` are available for selection in the Struct Type drop-down list.

For more information about structure types, see *Structure Types* in the *JD Edwards EnterpriseOne Tools Workflow Tools Guide*.

- o Select **Email** to enter an email address for the recipient.
 - o Select **Logged in User** to send a message to the user who is currently logged in.
4. In the **Subject** and **Body** fields, enter text, variables, or a combination of both. To insert variables, see step 11.
 5. (Release 9.2.6.3) Enable the **Plain Text** option to include a plain text in the message.

Disable the **Plain Text** option to add rich formatting to your existing messages. This option is disabled by default for new messages.

6. (Release 9.2.6.3) To format the messages:
 - a. Enter the message in the body field and use the options such as Paragraph, Bold, Italics, Underline, Strikethrough, Bulleted and Numbered List, and so on from the toolbar to format your messages. You can use the Insert Table and Insert Image options to insert tables and images.

You can also insert action links within your formatted messages. The link IDs defined for Data Dictionary, Application Links, Other Links, and Report Attachments are displayed in the Placeholder drop-down menu. To add an action link to your message, keep the cursor where you want to insert the

placeholder, and click the Placeholder drop-down menu and then select the required action link identifier. Additionally, you can add a horizontal line in your message by clicking the Horizontal Line icon. Starting with Tools Release 9.2.6.4, you can add a hyperlink to the text in the body of the message. Highlight the relevant text, and then click the Link icon in the toolbar, enter the required address, and click the Save icon.

Starting with Tools Release 9.2.7.4, the following new options are available in the editor:

- Subscript and the Superscript icons to add mathematical formulas. For example, $y = x^{2n} + z^{3n}$, $a_1 = a_2 + a_3$, and so on.
 - List Properties option in the Numbered List drop-down to change the properties of the numbered list order.
- b. To use an existing template, click the Template button, select the template from the left-hand panel in the Templates window and place the template at the position you want by using the **Insert at Top**, **Insert at Cursor**, or **Insert at Bottom** buttons.
- Note:** The templates are created using the Work With Media Object Templates(P98TMPL) application.
- c. Optionally, you can click the **Preview** button to verify the formatted message. Click the **Mobile** icon in the Preview window to verify how the message appears on a mobile device.

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

Note: [Understanding Rich Formatting of Messages Learning Path](#)

7. To include boilerplate text from a message template in the data dictionary:
- a. Expand the **Data Dictionary** section.
 - b. If the Plain Text option is disabled, the system displays the **DD Link ID** field. Enter a unique ID in the DD Link ID field and click the **Apply** button. This data dictionary link ID is displayed in the Placeholder drop-down menu enabling you to add it as a placeholder in your formatted message. You will not be allowed to change the DD Link ID after you click the Apply button. The system displays an error message if you enter a duplicate ID.

Note: The character limit for the DD Link ID field is 50 and special characters are not allowed in this field except underscore (_).
 - c. In the Data Dictionary field, enter the name of the message template data item and click **Load**.
 - d. If the message template contains variables, use the grid in the right to replace the variables with a default value or a variable. For each variable used in the message template, a row is added to the adjacent grid. In the grid, slide the Literal toggle to right to replace the variable in the template with a literal value. Slide the Literal toggle to left to select an existing notification, orchestration, or a watchlist input variable from the drop-down list.
8. To include a link to an application:
- a. In the **Application Links** section, click **Add** and expand the section. To set up task tracking, see [Setting Up Task Tracking \(Release 9.2.8\)](#).

- b. If the Plain Text option is disabled, the system displays the Application Link ID field. Enter a unique ID in the Application Link ID field and click the **Apply** button. This application link ID is displayed in the Placeholder drop-down menu enabling you to add a placeholder in your formatted message.

You will not be allowed to change the Application Link ID after you click the Apply button. The system displays an error message if you enter a duplicate ID.

Note: The character limit for the Application Link ID field is 50 and special characters are not allowed in this field except underscore (_).

- c. Complete the **Application**, **Form**, and **Version** fields to specify the form that you want the application link to launch.
- d. If needed, you can define a personal form, query, or watchlist to be used when the application opens.

If the user receiving the message does not have view access to that particular personal form, query, or watchlist, the application will open without it.

- e. If the Plain Text option is enabled, you can see the **Pre Text**, **Link Text**, and **Post Text** fields. Use these fields to define the text that appears before, as, and after the link, respectively, in the message.

If the Plain Text option is disabled, you can see the **Link Text** field. Use this field to define the text that appears as the link in the message.

- f. In the grid, you can use variables to pass data to the application when the application is launched from the shortcut.
- g. Click the Add button in the Application Links section and repeat these steps to include multiple application links in a message.
- h. Click the Remove button (X) at the end of the New Shortcut section header to delete the application link that is added.

9. To include other links (for example, to launch an orchestration or notification):
 - a. In the **Other Links** section, click Add and expand the section.
To set up task tracking, see *Setting Up Task Tracking (Release 9.2.8)*.
 - b. If the Plain Text option is disabled, the system displays the **Other Link ID** field.
Enter a unique ID in the Other Link ID field and click the **Apply** button. This other link ID is displayed in the Placeholder drop-down menu enabling you to add it as a placeholder in your formatted message.
You will not be allowed to change the Other Link ID after you click the Apply button. The system displays an error message if you enter a duplicate ID.
Note: The character limit for the Other Link ID field is 50 and special characters are not allowed in this field except underscore (_).
 - c. Select the type of link you would like to add from the **Type** drop-down menu. Valid values are Orchestration, Notification, or URL.
 - d. Depending on the type you have selected, either select the orchestration name, notification name, or enter the URL.
 - e. In the **Link Text** field, enter the text you would like to appear as link in the message.
This link text appears in the message.
 - f. In the **Pre Text** and **Post Text** fields, enter the text you want to appear before and after the link text.
The Pre Text and Post Text fields are not displayed if the Plain Text option is disabled.
 - g. Use the grid to work with orchestration input, notification input, or a key for the URL, depending on which type of link you are using. In the Value column, you can enter either a variable or a default value as the input.
 - h. Click the Add button in the Other Links section and repeat these steps to include multiple links in a message.
Click the Remove button (X) at the end of the GroupBy section header to delete the links that were added.
10. To include report output as an attachment:
 - a. In the **Attachments** section, click **Report Attachments**, and then click **Add**.
 - b. If the Plain Text option is disabled, the system displays the **Report Link ID** field.
Enter a unique ID in the Report Link ID field and click the **Apply** button. This report link ID is displayed in the Placeholder drop-down menu enabling you to add it as a placeholder in your formatted message.
You will not be allowed to change the Report Link ID after you click the Apply button. The system displays an error message if you enter a duplicate ID.
Note: The character limit for the Report Link ID field is 50 and special characters are not allowed in this field except underscore (_).
 - c. In the grid, enter **Link Text**, **Job Number**, **Server**, and **File Type** for the report output that you want to attach. The report output can be generated at this stage by an orchestration or it can be previously generated outside of any orchestrations or notifications. Either standard EnterpriseOne reports or BI

Publisher reports can be attachments. You can define the Link Text, Job Number, and Execution Server as variables.

- d. Enable the **Send As Link** toggle to attach the reports as links in the message. By default, the toggle turned off. Therefore the attachments are sent as files in the message.
- e. Click the Add button in the Attachments section and repeat these steps to include multiple attachments in a message.

Click the Remove button (X) at the end of the each row in the grid to delete the reports that were added.

- 11. 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.

Ensure the syntax includes the \$ sign and braces, for example:

```
${creditmanager}
```


12. (Release 9.2.6) To include file attachments (media objects) in the message:

Note: Including attachments in messages is only supported when media objects are stored in the database. For more information see [Uploading Media Object Files to Database](#).

- a. In the **Attachments** section, click **File Attachments**, and then click **Add**.
- b. In the File Attachment window, click the Search icon next to the Structure Name field, and then search for and select the required value.

Depending on the value in the Structure Name field, the values in the Description and Key Value fields are automatically populated in the Keys table. You may delete the automatically populated variable value in the Key Value field and enter a literal value.

Note: To determine the structure used in an EnterpriseOne application, access the required application using the Fast Path or open the application from the Navigator menu. Search for a record that contains attachments. In the Attachment Manager, click the (i) information icon to see the structure name in the Attachment Information section. For example, Attachment Information Name: GT0801 Key: 6001 Count: 9

Note: [Click here to view an OBE of this feature.](#)

- c. Click the **Which Attachments** drop-down menu and select the required option. The available options are:
 - **All:** Choose this option to select all the attachments present in the structure.
 - **Sequence:** Select this option to specify the sequence number of the attachment present in the structure. The Sequence field is displayed when you select this option. In the Sequence field, enter a sequence number or a sequence number variable.

Attachment	
* Structure Name	ABGT
Which Attachments	Sequence
Sequence	
Keys	
Description	Key Value
Address Number (AN8)	\${mnAddressNumber}

- **Default Image:** Choose this option to select the default image attachment.
- **First:** Choose this option to select the first attachment present in the structure.

The Attachment Type drop-down menu is displayed when you select this option. From the Attachment Type drop-down menu, select any of these options as required:

Note: JD Edwards EnterpriseOne supports these attachment types: File Attachment, Text Attachment, and URL Attachment.

- o **File :** Choose this option to select the first file attachment.

- **File Extension:** Choose this option to filter the first attachment by extension. The File Extension drop-down menu is displayed when you select this option and you can enter the required extension type. For example, if you enter `.pdf`, the first PDF file is added as an attachment in the message.

Attachment

* Structure Name Q

Which Attachments ▼

Attachment Type ▼

File Extension ▼

Keys

Description	Key Value
Address Number (AN8)	\${mnAddressNumber}

- **Image:** Choose this option to select the first image attachment.
- **Text:** Choose this option to select the first text attachment.
- **URL:** Choose this option to select the first URL attachment.
- **Last:** Choose this option to select the last attachment in the structure.

The Attachment Type drop-down menu is displayed when you select this option. From the Attachment Type drop-down menu, select any of these options:

- **File :** Choose this option to select the last file attachment.
- **File Extension:** Choose this option to select the last attachment by extension. The File Extension drop-down menu is displayed when you select this option and you can enter the required extension type. For example, if you enter `.pdf`, the last PDF file in the structure is added as an attachment in the message.
- **Image:** Choose this option to select the last image attachment.
- **Text:** Choose this option to select the last text attachment.
- **URL:** Choose this option to select the last URL attachment.
- **All of Type:** Choose this option to select all the attachments of a particular type to the message.

The Attachment Type drop-down menu is displayed when you select this option. From the Attachment Type drop-down menu, select any of these options:

- **File :** Choose this option to select all the file attachments present in the structure.
- **File Extension:** Choose this option to select the attachments by extension. The File Extension drop-down menu is displayed when you select this option and you can enter the

required extension type. For example, if you enter `.pdf`, all PDF files in the structure are added as attachments in the message.

- o **Image:** Choose this option to select all the image attachments.
- o **Text:** Choose this option to select all the text attachments.
- o **URL:** Choose this option to select all the URL attachments.

d. Click **OK**.

13. (Release 9.2.8.3) You can pass in a data set (array) and format that data as a table in the message. To add a data table:

- a. Click the Data Tables section, and then click **Add**.
- b. In the Add Data Table window, the option Include Header Row is selected by default. Deselect this option if you do not want a header row for your table.
- c. Click **Add**.
- d. In the Data Set Name field, enter a name, and click **Apply**. The system displays the value in the Data Set Name field in the header area of the Data Tables section.

Note: The value you enter in the Data Set Name field must match the data set name in the orchestration step that is the source of the data. For example, the value in the Data Set Name field must be the same as the data set name defined in the data request, form request, or connector output. If there is no matching data set from a previous step, you will not be able to map the individual fields for the table in the transformation for the message step.

- e. Click **Edit**. An Edit Data Table window is displayed along with the table template.
- f. Edit the header text and variables in the table as required.

You can click the table and select the displayed options, such as Column, Merge Cells, and so on, to change the table properties. You can also use the Font Color, Font Size, Paragraph, Bold, Italics, Underline, Strikethrough, Bulleted and Numbered List options from the toolbar to format your text in the table.

- g. Click **OK**. The system creates a table link ID in the Placeholder drop-down list enabling you to add it as a placeholder in your formatted message.

Note: You can click Add in the Data Tables section to define more tables.

Note: [Click here to view an OBE of this feature.](#)

14. Click the **Preview** button to preview the message. The preview dialog displays the subject and the body of the message including the data dictionary boilerplate text, links to applications, URLs, orchestrations, notifications, and reports.
15. Click **Save** to save your changes.

Note: The Message service requests created in Orchestrator Studio 9.2.4 can be loaded and used in the previous version of the Orchestrator Studio.

Setting Up Task Tracking (Release 9.2.8)

Tasks are records that are tracked in the F980070 and F980071 tables. These records can be automatically created by enabling task tracking for actions in a message. Each task also has a set of historical state records to track when, and who updated the task.

In Messages, you can set due dates for tasks that are assigned to recipients through orchestration messages by enabling the Task Tracking option. The tasks can then be monitored to ensure their timely completion and analysis.

You can enable the Task Tracking option in the Application Link and the Other Link sections of the Message.

In the Task Tracking window, you can define a due date and the tracking state values. The system adds the defined due date, and the tracking state values as links to the Placeholder drop-down list in the body field of the message. You can then add these links to your messages, enabling the recipient to access the task in the Task Tracking (P980070) application and to click and perform the required actions directly from the message.

When you enable task tracking for any action, the system provides several variables to map the inputs of the action. The variables include the **Task Tracking ID** and a variable for each possible task tracking state. For example, you can map these variables into the action link if you have designed an orchestration or an application that updates the task state automatically. When a recipient clicks the action link in the message, the system uses the tracking ID and process the task associated with that action.

To enable task tracking:

1. Click **Add** and expand the Application Link or the Other Link section.
2. Click the **Task Tracking** icon.

The Task Tracking window is displayed.

Note: You must define and apply a Link ID before setting up task tracking.

3. Enable the **Track Task** option.

The system displays a blue dot next to the Task Tracking icon to indicate that the task tracking is enabled for the message.

4. In the Description field, enter a description. The description entered in this field is included in the task record when the message is sent.
5. Select one of these options to set the task due date:

Note: The system adds the defined due date as a link to the Placeholder drop-down list in the body field of the message. This enables you to add the due date link to your message.

- **No Due Date:** Select this option if you do not want to setup a due date for task tracking.
- **Use Date Rule:** Select this option to define a custom date rule. You can define your date rule by selecting the number of minutes, hours, days, weeks, months, and years. You can also select a specific time from the Time Due field, and then select the required time zone from the Time Zone drop-down list. Enable the **Allow Due Date Override** option if you want to allow the recipient to override the due date.
- **Use Date Variable:** Select this option to enter variable as a value in the Due Date Variable field. Enable the **Allow Due Date Override** option if you want to allow the recipient to override the due date.

-
6. In the Tracking Link Options section, you can define links for updating the tracking state (status) of the task to your message.

Select one of these options:

- **No Link:** Select this option if you do not want to provide any links in the message. The message recipient can use the Task Tracking application (P980070) to update their tasks. See *Understanding the Task Tracking Application*.
- **Link to Task Tracking:** This option enables you to update the state of the task by adding a single application link to the Edit Task Tracking (W980070C) application.

When you select the Link to Task Tracking option, the system enables the Link Text field and a list of tracking state values. Enter a value in the Link Text field and select the desired options from the following tracking state values: Approved, Closed, Completed, Delayed, In Process, Rejected, Sent, and Viewed.

The values that you select in the Link to Task Tracking option will be listed in the Task State field in the Edit Task Tracking (W980070C) application when the user clicks the link from the message.

- **The Link for Each State:** This option enables you to add multiple action (task tracking states) links in the message. The system defines one link for each task state that you select. When the recipient clicks the

link in the message, an orchestration will process the required change to that specific state. For example, Approved or Rejected.

When you select the Link for Each State option, the system enables a list of tracking states. Select the desired tracking state values from the list: Approved, Closed, Completed, Delayed, In Process, Rejected, Sent, and Viewed.

In the following example, the tracking state values, Completed and In Process are selected in the Task

Task Tracking Track Task

Description

Confirm your job profile information and indicate you have completed the review (employee \${employeeNo})

No Due Date Use Date Rule Use Date Variable

Due Date Rule *
3 Days

Time Due 11:55 PM Time Zone America/Denver

Allow Due Date Override

Tracking Link Options ⓘ

No Link Link to Task Tracking Link for Each State

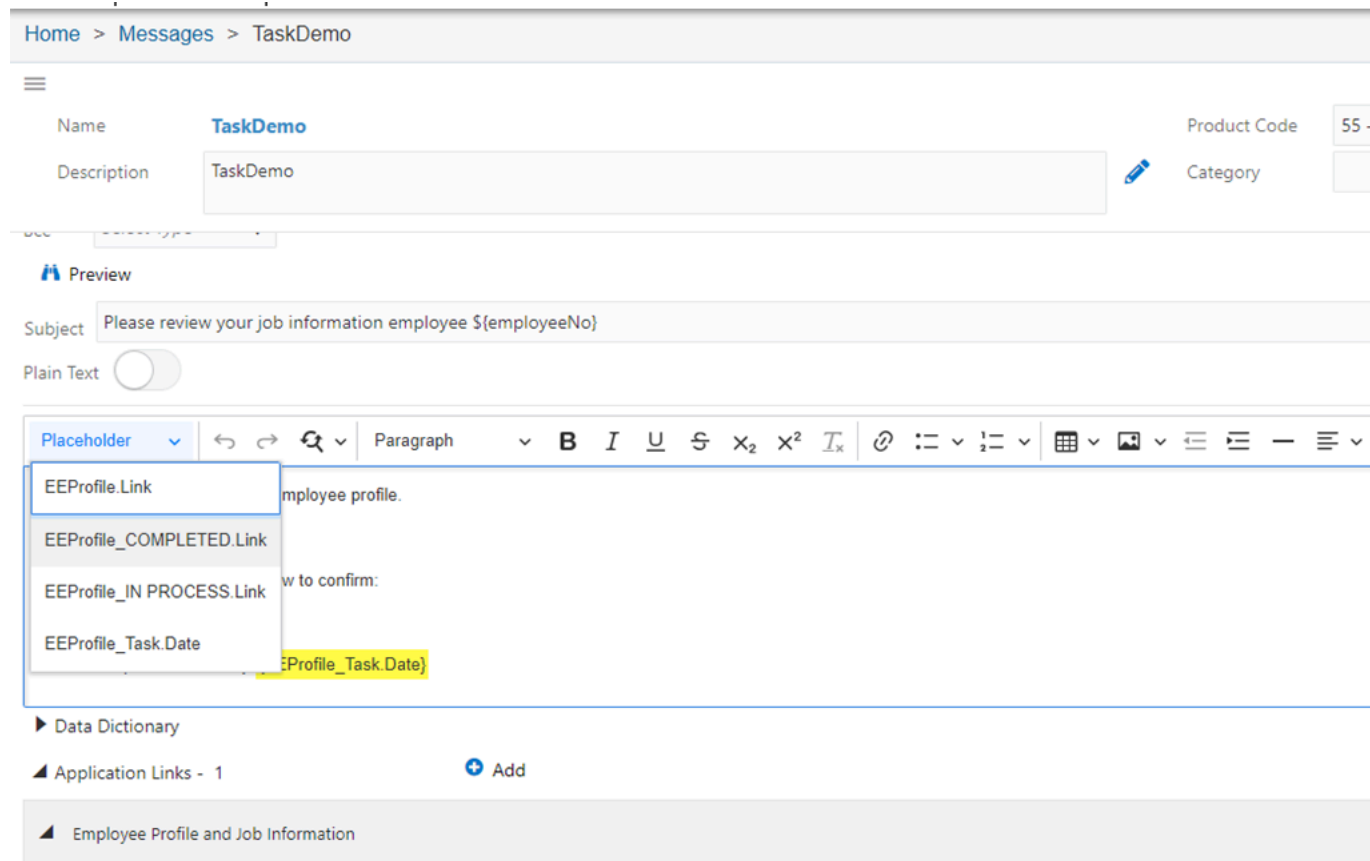
Select from Available Tracking States

Approved Closed Completed Delayed
 In Process Rejected Sent Viewed

Tracking window.

The system adds the defined date rules and the selected tracking state values to the Placeholder drop-down list in the body of the message as shown in the

following screenshot. You can then select and add these links to your message.



Note: The Orchestrator Studio provides these standard tracking state values: Approved, Closed, Completed, Delayed, In Process, Rejected, Sent, and Viewed. The tracking state values are UDCs that can be customized. You can customize them using the **H981/TT** UDC code. Make sure to clear the caches on the HTML Servers after making the changes. Otherwise, the Orchestrator Studio will not reflect the modified tracking state values. Also, when customizing, you can choose if the state is Terminal or not by including Terminal in the Special Handling code. The system will not allow you to make further changes to a task once it is at a Terminal state. To learn more about how to customize UDCs, see *Customizing UDC Types*.

7. Click **Close**.
8. (Release 9.2.8.3) In the **Task Tracking ID Array Name** field , enter the name of the array of the task tracking IDs, which can be used in a subsequent step or in the orchestration output. This field is displayed only if you have links with Task Tracking enabled.

Note: *Click here to view a recording of this feature.*

Note: *Click here to view a recording of this feature.*

Note: *Click here to view an OBE of this feature.*

Understanding the Task Tracking Application (Release 9.2.8)

The JD Edwards EnterpriseOne provides the Task Tracking (P980070) application to review tasks and make changes to the value in the State field or Due Date field of an assigned task. You can also add and delete tasks manually using the application.

Using JD Edwards Orchestrator Studio, you can add a link to the Task Tracking application in Messages by enabling the Task Tracking option. See *Setting Up Task Tracking (Release 9.2.8)*.

In the Task Tracking application, you can view the list of tasks in a table with details such as Tracking ID, Description, Task State, Task Closed, Sent Date, Due Date, Last Changed Date, and so on.

When you open the Task Tracking application, the system displays all the open tasks associated with the current user (shown in the Address Number field) by default, and the tasks that are in the Terminal state are not displayed.

You can filter the tasks by using these options:

- **View All Tasks:** Select this option to remove the current user filtering and to display all the tasks.
- **Open Tasks:** Select this option to view only the tasks that are in an open state. Deselect this option to view all the tasks, including the terminal state tasks.
- **Sent to My Email:** Select this option to find the tasks sent directly to the displayed email address in the Email field associated with the current user.

You can also sort the tasks by using the Sent Date and Description options from the Sort By drop-down list.

When you select a particular task from the table, the system displays the **Edit Task Tracking** window. You can view further details about the task in the **Task Details** and **Task State** tabs, and review the task history in the History table. In the **Task State** tab, you can use the State drop-down list to change the value of the State field. Or, you can right-click the task in the Task Tracking application, select **Change State** to access the Task State window, and then change the value in the **New State** drop-down list. You must click **OK** to save your changes.

You can also update the **Due Date** field depending on the configuration of the task.

You can add media object attachments while changing the state of the task. To add an attachment while changing the state, click the **Add Attachments** button, add the media object, and then click **Save**. You can view the attachments added for each state change made by any user in the History table, and you can edit the attachments if you are the listed user that made the state change.

Configuring a Connector Service Request

This section contains the following topics:

- *Understanding Connector Service Requests*
- *Before You Begin Configuring a Connector Service Request to Invoke an Orchestration or a Notification*
- *Configuring an Open API Connector to Invoke a REST Service*
- *Configuring a REST Connector to Transfer Files to a REST Service*
- *Configuring a Database Connector*
- *Configuring a Connector to Transfer Files Using File Transfer Protocol (FTP)*

Understanding Connector Service Requests

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

- Invoke another orchestration or a notification on an AIS Server on another EnterpriseOne system in a multisite operation.
- Invoke a REST service. Referred to as a REST connector, this connector enables outbound REST calls to external systems through 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.

You can also configure a REST connector to invoke REST services from your local AIS server. By using the Local AIS Call connection, you can run a REST service on the current AIS server by using the existing session established to run the calling orchestration

- Connect to a database. Referred to as a database connector, this connector 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 enables data that is not appropriate for transaction tables to be stored for analysis, archiving, or integration.
- Retrieve a file from or send a file to a known location using either the File Transfer Protocol (FTP) or the Secure File Transfer Protocol (SFTP). This connector is referred to as an FTP connector. 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.
- List all the REST services that can be performed on Server Manager using the Open API connector.
- Invoke REST services from your local AIS server. The session that is already established to execute the orchestration is reused for the current AIS REST call.

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.

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

Note: With Tools release 9.2.4.3, you do not have to create a connector service request to invoke "local" orchestrations or notifications. You can add your own orchestrations and notifications directly as step in the calling orchestration. However, a connector is required to invoke an orchestration or a notification on an AIS Server on another EnterpriseOne system in a multisite operation. (Release 9.2.4.3)

1. Click the Connectors icon on the Orchestrator Studio Home page.
2. On the Connectors side panel, click the **New** drop-down menu and select Orchestration.
3. Create and name the connector service request as described in [Creating a Component](#).
4. On the Connectors design page, select a connection to the AIS server from the Orchestration drop-down list.

After you select a connection, the Orchestrator Studio displays the URL of the AIS server next to the Orchestration field.

5. In the **Orchestration/Notification** drop-down list, select the appropriate option depending on whether you want to invoke an orchestration or a notification.

Depending on your selection, a pop-up dialog box is displayed with a list of orchestrations or notifications.

6. Search for and select the orchestration or notification that you want to invoke.

You can click the **Search** button to reopen the pop-up dialog box that displays a list of notifications or orchestrations. The Search dialog box lists the available orchestrations or notifications, depending on your selection in the previous step. The list is categorized by these UDO status: Personal, Pending Approval, Reserved, and Shared.

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

7. Enable the **Use Object for Input** toggle if you want to map the entire object as an input to the orchestration connector. This action enables you to pass arrays to an orchestration that is called through a connector. (Release 9.2.4.3)
8. 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 called orchestration will run asynchronously. The Output grid is hidden because the REST service response is not processed.

(Release 9.2.4.4) The **Number of Threads** option is displayed when you enable the Fire and Forget toggle. This option enables you to select the number of threads for an iterated step for an iterated step.

For example, if an array of 1,000 records is sent as the iterator for a fire and forget orchestration and the value in the Number of Threads option is 10, only 10 orchestrations will be executed at one time. The remaining orchestrations will be in a waiting state until one of the 10 threads is available again. Similarly, the remaining set of orchestrations are processed until all of them are completed.

9. In the **Input** column, you can modify the variable names as you want or map a default value by entering a value (without the `{ }` notation).
10. For an orchestration connector, use the **Output** grid to specify the values you want to be returned from the called orchestration.

For a version 3 orchestrations, the Orchestration Connector outputs are used for variables only. 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 calling orchestration.

You have to populate the Variable Name column and that name will be available for mapping in the orchestration.

Configuring an Open API Connector to Invoke a REST Service

You can configure an Open API connector to invoke a REST service. In the Open API connector, when you specify the API call and set the HTTP methods that are populated in the system according to the allowed API call, the Pathing and Parameter fields are automatically filled with values as required for the REST service.

Note: The Open API connectors only support OpenAPI 2.0 standard (Swagger 2.0), and not the OpenAPI 3.0 standard version.

You can test the connector and view the JSON response body of the REST service. You can also use Groovy, JRuby, Jython scripting to refine the data in the connector output.

Starting with Tools Release 9.2.5.5, the variables defined inside the JSON strings in the body of the request should include the `esc` notation. The `esc` notation ensures that the string values are properly escaped for use as JSON strings at runtime.

For example, you should define the JSON input as follows in the body of the REST connector:

```
{
  "value": "$esc{myVariable}"
}
```

instead of using this traditional method:

```
{
  "value": "${myVariable}"
}
```

In the above examples, a variable called `myVariable` is defined. In the first example, the value inside the variable will be escaped for use as a JSON string at runtime. An invalid JSON may be created if you use the second example, and the runtime value may have invalid characters and they will not be escaped.

To configure an Open API connector to invoke a REST service:

1. Click the Connectors icon on the Orchestrator Studio Home page.
2. On the Connectors side panel, click the **New** drop-down menu and select Open API.
3. Create and name the connector service request as described in *Creating a Component*.
4. On the Connectors design page, click the **Open API** field and select a connection.

The URL to the REST service for the Open API connection is displayed.

5. Click the **API** drop-down list and select the API from the list of all the available APIs as defined in the Open API documentation. The HTTP methods will be populated as applicable to the selected API call.
6. Click the **HTTP Method** drop-down list and select the method as required.

After you select the HTTP Method value, if available in the documentation, the corresponding values for the selected API are displayed in the API Description and API Summary fields. Also, the Pathing, Parameters, Headers, and Body sections are automatically filled with the appropriate values.

Note: Pathing section is read-only and hence you cannot modify this section. The value in the Pathing section is displayed based on the API definition. You can change the values defined in the Parameter and the Body sections if you want to hard-code the values that are passed.

Starting with Tools Release 9.2.8.2, the Orchestrator Connector step is enhanced to allow files to be sent to external REST APIs either as a multipart request or as a "direct" request (without multipart boundaries in the request body). If you select the **File Upload** option in the Headers section, the system displays the **Binary Body** option. You can enable the **Binary Body** option to send the file as the body of the HTTP request. When you select the **Binary Body** option, you can add an appropriate header value for Content-Type. If the Content-Type header is not defined, the system attempts to find the matching MIME type based on the file name. If the MIME type is not found, then the system uses `application/octet-stream`.

7. 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.

(Release 9.2.4.4) The **Number of Threads** option is displayed when you enable the Fire and Forget toggle. This option enables you to select the number of threads for an iterated step.

8. (Release 9.2.8.2) In the Output section, you can enable the **Include Response Status** option, and enter the variable. This option enables the connector step (REST or OpenAPI) to assign the returned HTTP status code to an orchestration variable. Having the HTTP status code as an orchestration variable then gives you more control over an intelligent reaction to the HTTP status. For example, if a third-party system returns an HTTP status of 500, the orchestration might pass that status into a rule, which in turn sends a notification message indicating the failure.
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.

- o **Manipulate Output** (advanced). Use Groovy, JRuby, or Jython scripting to modify 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 that is displayed in the output. See *Groovy Template for Manipulating Output from a REST Connector Response* for more information. Click the Show Shortcut Command icon (?) to view the commands to edit the script.
 - o **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 brackets ([]).
10. To test the connector:
 - a. Click the **Test** button.
 - b. If the connector includes parameters, enter values for the parameters in the pop-up box.

Clicking **Execute** displays the REST service response.
 - c. Click the **Show Output** button that is displayed at the end of the response to view the modified output.

Note: The Pathing, Parameter, Header, and Body sections are prefilled with optional values. You can remove the optional values per your requirement and save the connector.

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 scripting to refine the data in the connector output. For example, you can configure a script to change 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. Click the **Connectors** icon on the Orchestrator Studio Home page.
2. On the Connectors side panel, click the **New** drop-down menu and select REST.
3. Create and name the connector service request as described in *Creating a Component*.
4. On the Connectors design page, click the **REST** field and select a connection.
5. Click the **HTTP Method** drop-down list and select the appropriate method.
6. 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.

(Release 9.2.4.4) The **Number of Threads** option is displayed when you enable the Fire and Forget toggle. This option enables you to select the number of threads for an iterated step.

7. (Release 9.2.5.2) Enable the **Use Connection Security** option if you want to use the security defined in the connection.
8. 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 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.

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

Parameters are appended to the endpoint after a question mark (?) and are separated by an ampersand (&). You can include variables by specifying a value inside `{}`.

10. 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.

Starting with Tools Release 9.2.6.4, the File Upload and Multipart toggle options are available in the Headers section. When you select the File Upload option, the system enables the File section and the Parts section of the request where you can configure files (reports or files from the temporary directory) as input to the REST service in addition to any text-based input to the request.

When you select the Multipart option, the system enables you to set the Content-Type to multipart/form-data. When you select this option, the system enables only the Parts section to allow for multipart request without sending a file.

Starting with Tools Release 9.2.8.2, the Orchestrator Connector step is enhanced to allow files to be sent to external REST APIs either as a multipart request or as a direct request (without multipart boundaries in the request body). When you select the **File Upload** option, the system displays the **Binary Body** option. You can enable the **Binary Body** option to send the file as the body of the HTTP request. When you select the **Binary Body** option, you can add an appropriate header value for Content-Type. If the Content-Type header is not defined, the system attempts to find the matching MIME type based on the file name. If the MIME type is not found, then the system uses `application/octet-stream`.

11. In the **Body** section (which is not used or displayed if the HTTP method is GET or TRACE), if required by the HTTP method, specify a payload to send to the REST service.

Note: Starting with Tools Release 9.2.5.5, the variables defined inside the JSON strings in the body of the request should include the `esc` notation. The `esc` notation ensures that the string values are properly escaped for use as JSON strings at runtime. For example, you should define the JSON input as follows in the body of the REST connector:

```
{
  "value": "$esc{myVariable}"
}
```

instead of using this traditional method:

```
{
  "value": "${myVariable}"
}
```

In the above examples, a variable called `myVariable` is defined. In the first example, the value inside the variable will be escaped for use as a JSON string at runtime. An invalid JSON may be created if you use the second example, and the runtime value may have invalid characters and they will not be escaped.

12. (Release 9.2.5.2) Expand the **Response** section and in the **Header** section, enter the name of the header field in the **Output** column and a variable name in the **Variable Name** column. By defining a variable, you can pass a header value from the response of the REST service to the subsequent orchestration steps. For example, after successful authentication, a REST service may return an authentication token in the response header. You can pass that token as a variable to subsequent orchestration steps to maintain an authenticated session.
13. In the **Manipulate Body** section, use scripting to manipulate the output of the REST call. For example, if you only need certain data in the output, you can use a script to refine the data displayed in the output. Depending on the scripting language you choose, a template will be displayed and you can modify it to manipulate the output. For Groovy, see [Groovy Template for Manipulating Output from a REST Connector Response](#) for more information.
14. (Release 9.2.8.2) In the **Output** section, you can enable the **Include Response Status** option, and enter the variable. This option enables the connector step (REST or OpenAPI) to assign the returned HTTP status code to an orchestration variable. Having the HTTP status code as an orchestration variable then gives you more control over an intelligent reaction to the HTTP status. For example, if a third-party system returns an HTTP status of 500, the orchestration might pass that status into a rule, which in turn sends a notification message indicating the failure.

15. In the **Output** section, use the following sections to identify the outputs:

- o **JSON Output.**

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 the value by adding the proper index of the array inside brackets ([]).

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.

To understand how to define the name and category as objects in the array, see the following section *Defining Array Outputs from Connectors (Release 9.2.5.2)*.

- o **Output to a File** (Release 9.2.6.2)

Use this option to accept files from the response from external REST calls and to save and name the file output for use in subsequent orchestration steps.

Complete the following fields:

- **File Name:** Enter a file name or a variable.
- **Overwrite Existing Files:** Enable this option if you want to use the original file name for all the files throughout the processing. Disable this option to assign a unique file name to each file.
- **Keep Temp Files:** Enable this option to save the files in the temporary directory. The files saved in the temporary directory will remain there indefinitely. Disable this option if you want to save the files in the user's session temporary directory. The system deletes these files when the session ends.
- **File Name Variable:** Enter the file name variable that can be referred in the subsequent orchestration steps.

Note: You can test the response by using the Test button. When you click the Test button, the system will call the service and download the output file.

16. To test the connector:

- a. Click the **Test** button.
- b. If the connector includes variables, enter values for the variables.

If test is successful, the Orchestrator Studio displays the response.

- c. Click the **Show Output** button that is displayed at the end of the response to apply the instructions that are defined in the Output section, including any manipulation specified in the script.

Use the results to validate the path to the output values that are specified in the Output section.

Defining Array Outputs from Connectors (Release 9.2.5.2)

The external service can return data either as simple variables, such as strings or numbers, or as an array such as a list of customer names, account numbers, and addresses. This feature enables you to map those returned arrays to orchestration variables, which can then be acted upon and iterated over by subsequent orchestration steps.

As of Tools Release 9.2.5.2, you can enable the **Array** option in the Body section of Response to indicate that the data returned is an array and you can define the output sets you want in the array.

Enter the name in the **Variable Name** field for the array and specify the objects you want to be returned in the array. Enter a value in the **Named Object** column if you want to pass the entire array as an object to a connector or an orchestration. You can then pass this object for `jde_object` in the Input column in the Transformations window, instead of iterating over the array directly. This step enables the called orchestration to define how and when the iteration happens.

As shown in the following example, you can define the name and the category as objects in the array by entering these values in the Variable Name column:

The screenshot shows the configuration of a variable in the Orchestrator Studio. The variable is named "array" and is of type "array". The "Variable Name" column contains the value "arrVar". The "Named Object" column contains the value "array". The "Output" column contains the value "array". The "Body" section is expanded, showing a table with columns for "Output", "Variable Name", and "Named Object".

Output	Variable Name	Named Object
array	arrVar	array
array	name	name
array	category	category

When you run this example, you can see the name and category objects defined for the array in the output.

```
{
  "array": [
    {
      "name": "Pet1",
      "category": {
        "id": 219,
        "name": "Category 1"
      }
    },
    {
      "name": "Pet2",
      "category": {
        "id": 190,
        "name": "Category 2"
      }
    },
    {
      "name": "Pet3",

```



```

"category": {
  "id": 898,
  "name": "Category 3"
}
}
]
}

```

You can also define the array shown in the preceding example using the dot notation within the array objects as shown in the following screenshot:

Body

Output	Variable Name	Array	Named Object
array	arrVar	<input checked="" type="checkbox"/>	
array name	name		
array category.id	category		
array		<input type="checkbox"/>	

You will get the following output when you use the dot notation. When you use the dot notation, the system pulls the values from inside the object (category) up to the top level of each array element (not inside their own objects).

```

{
  "array": [
    {
      "name": "Pet1",
      "category.id": 219
    },
    {
      "name": "Pet2",
      "category.id": 190
    },
    {
      "name": "Pet3",
      "category.id": 898
    }
  ]
}

```

Similarly, in the body of the Response section you can define multiple number of arrays and the output sets you want in each array.

Configuring a REST Connector to Invoke REST Services from a Local AIS Server

You can configure a REST connector to invoke REST services from your local AIS server. By using the Local AIS Call connection, the session that is already established to execute the orchestration, is reused for the current AIS REST call.

So, a new session is not created. The need to set up a connection to connect to your local AIS server is eliminated by using the Local AIS Call connection.

1. Click the **Connectors** icon on the Orchestrator Studio Home page.
2. On the Connectors side panel, click the **New** drop-down menu and select REST.
3. Create and name the connector service request as described in [Creating a Component](#).
4. On the Connectors design page, click the REST field and select Local AIS Call.
For details regarding the HTTP Method field and the Pathing, Parameter, Headers, Body, and Output sections, see [Configuring a REST Connector to Invoke a REST Service](#).
5. Click Save.

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 any steps that generate or save files to the AIS Server, you can create a REST connector to transfer the generated file to an external system.

1. Click the **Connectors** icon on the Orchestrator Studio Home page.
2. On the Connectors side panel, click the **New** drop-down menu and select REST.
3. Create and name the connector service request as described in [Creating a Component](#).
4. On the Connectors design page, click the **REST** field and select a connection
5. Click the **HTTP Method** drop-down list and select **POST**.
6. 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. (Release 9.2.4.4) The **Number of Threads** option is displayed when you enable the Fire and Forget toggle. This option enables you to select the number of threads for an iterated step.
7. 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.
8. 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:
 - o **Job Number**. Enter `${ variablename }` to use a variable for the job number. When you add this connector to an orchestration, you can map the job number of a report that is returned from a report service request to this variable. Alternatively, instead of a variable, you can enter an actual job number.
 - o **Execution Server**. Enter the server where the report was generated. Enter `${ 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. Alternatively, instead of variable, you can enter the server name.
 - o **File Type**. Select the file type of the output.

For File, complete these fields:

- o **Source File Name**. If the file is in the AIS temporary directory, enter just the name of the file. If the file is in another accessible file location on the AIS server, enter the fully qualified path of the file.
Click the **Use Temporary File Location** option to include a file from the temporary directory on the AIS Server

- **Remove File.** This option is enabled by default, and removes the file from the temporary directory on the AIS Server after the file is transferred.
9. If the REST API that is being called requires additional information, you can use the table at the bottom of the File section to include additional details.

Note: The variable notation `${variable}` is supported for multi-part REST connectors. Add one or more variables in any of the entry fields to pass values to the connector. The inputs displayed in the test window depict the variables you entered (Release 9.2.5).

Configuring a Database Connector

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

1. Click the **Connectors** icon on the Orchestrator Studio Home page.
2. On the Connectors side panel, click the **New** drop-down menu and select Database.
3. Create and name the connector service request as described in *Creating a Component*.
4. On the Connectors design page, click the **Database** field and select a connection.

The Orchestrator Studio displays an edit area that contains a script template corresponding to the selected scripting language (Groovy, JRuby, or Jython). Click the **Show Shortcut Command** icon (?) to view the commands to work with the script.

5. In the **Input** grid, enter the inputs that you want to pass to the database connector.
6. 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.
7. If you intend to return a set of records, enter a name for the data set in the **Data Set Variable Name** field.

You have to then define the column names (fields) for the records you want to return (such as name, location, and so on).

Member Names - Variable Name grid is applicable only used if you enter a value in the Data Set Variable Name field.

8. The **Member Name** column correspond to the field in the table in the database. The **Variable Name** column is the name that you would use to refer to the corresponding field when passing it to the other orchestration steps.
9. Click **Save**.

Configuring a Database Connector Using DADriver (Release 9.2.7)

Starting with Tools Release 9.2.7, you can configure a database connector in the Orchestrator Studio that uses the JD Edwards EnterpriseOne Data Access Driver (DADriver). The DADriver is a read-only type 4 JDBC driver and it enables read-only connectivity to the EnterpriseOne database. Using this connector, you can read data from the EnterpriseOne database by using complex SQL statements and joined tables.

For more information about the DADriver and the supported SQL grammar, see:

Using the JDBC Driver

For information about the settings for JDBj.ini, see:

Create an Application Interface Services (AIS) Server as a New Managed Instance

Note: You must have knowledge of the Groovy, JRuby, or Jython scripting language to create a database connector to read data from the EnterpriseOne database. The Connector design page provides a template for the selected scripting language that you can use as a basis for creating a script for a database connector.

1. Click the **Connectors** icon on the Orchestrator Studio Home page.
2. On the Connectors side panel, click the **New** drop-down menu and select Database.
3. Create and name the connector service request as described in *Creating a Component*.
4. On the Connectors design page, click the **Database** field and select the **EnterpriseOne Database (Read-only)** connection. Selecting this option enables you to connect to the DADriver that provides a built-in, preconfigured, and read-only connection to the EnterpriseOne database.

Note: The system uses the same login credentials used for the Orchestrator Studio to connect to the DADriver. For executions outside of the Orchestrator Studio, the credentials used to run the orchestration are also used for the DAD driver.

The Orchestrator Studio displays an edit area that contains a script template corresponding to the selected scripting language (Groovy, JRuby, or Jython). Click the **Show Shortcut Command** icon (?) to view the commands to work with the script.

In the edit area, you can add advanced database operations like complex queries and table joins.

5. In the **Input** grid, enter the inputs that you want to pass to the database connector.
6. 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.
7. If you intend to return a set of records, enter a name for the data set in the **Data Set Variable Name** field.

You must then define the column names (fields) for the records you want to return (such as name, location, and so on).

Member Names - Variable Name grid is applicable only used if you enter a value in the Data Set Variable Name field.

8. The **Member Name** column correspond to the field in the table in the database. The **Variable Name** column is the name that you would use to refer to the corresponding field when passing it to the other orchestration steps.
9. Click **Save**.

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

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.

Using the temporary directory on the AIS server is optional. You can also enter a fully qualified path to any directory on the server that the FTP server has access to. This applies to send and receive file.

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.
- Retrieve data from a CSV file. The data is imported as an array. As a result, you can 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. Click the **Connectors** icon on the Orchestrator Studio Home page.
2. On the Connectors side panel, click the **New** drop-down menu and select FTP.
3. Create and name the connector service request as described in *Creating a Component*.
4. On the Connectors design page, click the **FTP** field and select a connection.

If no connections are available for an FTP connector, ask your system administrator to create one. See *Creating a Soft Coding Record for an FTP Server Connection*

5. Click the **Report** option and complete these fields:
6. 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 you want. Alternatively, you can delete the variable (including the special characters) and replace it with a server name.
 - **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 that is returned from a report service request to this variable. You can modify the variable name if you want. Alternatively, you can delete the variable (including the special characters) and replace it with an actual job number.
 - **File Type.** Select the type of file that the FTP connector will transfer. The default is PDF for standard EnterpriseOne reports. If you are transferring an embedded BI Publisher report, select one of the BI Publisher file types.
 - **Path Extension.** Enter the name of the subdirectory on the third-party FTP server where you want the report output to be sent. The FTP Connection already contains information for the base folder.
7. Click **Save**.

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

1. Click the **Connectors** icon on the Orchestrator Studio Home page.
2. On the Connectors side panel, click the **New** drop-down menu and select FTP.
3. Create and name the connector service request as described in *Creating a Component*.

4. On the Connectors design page, click the **FTP** field and select a connection to the server from which you want to transfer a file.
If no connections are available for an FTP connector, ask your system administrator to create one. See *Creating a Soft Coding Record for an FTP Server Connection*
5. Select the **File** option.
6. Click the **Type** drop-down list and select Receive.
7. In the **Source File Name** field, enter the name of the file that you want to retrieve.
8. In the **Target File Name** field, enter the name of the target file, if you want it to be saved to the temporary directory on the AIS Server, with a name that is different from the name of the source file.
9. Click the **Use Temporary File Location** check box to save the file to the temporary directory on the AIS Server.
10. In the **Path Extension** field, enter the name of the subdirectory on the third-party FTP server from where you want to retrieve the file.
Since Receive was selected in the Type field, the Path Extension field applies to the file that you want to retrieve.
11. Enable or disable the **Remove Source File** toggle. This toggle is enabled by default, and applies to the source file. It removes the file that you retrieved from the FTP server after it is transferred.
12. Click **Save**.

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

1. Click the **Connectors** icon on the Orchestrator Studio Home page.
2. On the Connectors side panel, click the **New** drop-down menu and select FTP.
3. Create and name the connector service request as described in *Creating a Component*.
4. On the Connectors design page, click the **FTP** field and select a connection to the server from which you want to transfer a file.
If no connections are available for an FTP connector, ask your system administrator to create one. See *Creating a Soft Coding Record for an FTP Server Connection*
5. Select the **File** option.
6. Click the **Type** drop-down list and select Send.
7. In the **Source File Name** field, enter the name of the file that you want to transfer.
8. Click the **Use Temporary File Location** check box if the file you want to send is in the temporary directory on the AIS Server.
Since Send was selected in the Type field, the Use Temporary File field applies to the file that you want to send.
9. In the **Target File Name** field, enter the name of the target file that will be created on the FTP server.
10. In the **Path Extension** field, enter the name of the subdirectory on the third-party FTP server where you want the file output to be sent.
11. Enable or disable the **Remove Source File** toggle. This toggle is enabled by default, and removes the file from the temporary location on the AIS Server after the file is transferred to an external server.
12. Click **Save**.

Configuring an FTP Connector to Import Data from a CSV File

Use an FTP connector to retrieve data from a CSV file on an FTP server. The data is imported as an array. As a result, you can 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. Click the **Connectors** icon on the Orchestrator Studio Home page.
2. On the Connectors side panel, click the **New** drop-down menu and select FTP.
3. Create and name the connector service request as described in *Creating a Component*.
4. On the Connectors design page, click the **FTP** field and select a connection to the server from which you want to transfer a file.

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

5. Select the **File** option.
6. Click the **Type** drop-down list and select **Receive**.
7. Enable the **Import CSV** option.

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

8. Enable the **Import First Row Only** option if you want to include only the first row (Release 9.2.5).
9. If the CSV file has a header row, enable the **CSV Has Headers** option to exclude the header row.

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

10. Complete the following fields:
 - o **Delimiter.** Enter the delimiter used in the CSV file to separate values.
 - o **Source File Name.** Enter the name of the CSV file.
 - o **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.
 - o **Remove Source File.** Enable or disable the Remove Source File toggle. This toggle is enabled by default. When enabled, this toggle removes the file from the temporary location on the AIS Server after the data is imported.
 - o **Data Set Variable Name.** Enter a name to refer to the array created from the CSV data.

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

Note: This field is available only when the **Import First Row Only** option is not selected. The system will not create an array when the **Import First Row Only** option is selected.

- o **Header Object (Release 9.2.4.4).** Enter the name of the header object to indicate how you want to group the data. See [Defining the Header Fields \(Release 9.2.4.4\)](#).
 - o **Detail Object (Release 9.2.4.4).** Enter the name of the detail object to indicate how you want to group the data. See [Defining the Header Fields \(Release 9.2.4.4\)](#).
11. Click the **File** button to select a model CSV file for defining the columns in the CSV file that you want to import.

You can use a CSV file that is a copy of the CSV file from which the FTP connector will import data, or you can use a different CSV file if it has the same column names.

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

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 that are 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.

12. Click **Save**.

Defining the Header Fields (Release 9.2.4.4)

You can designate one or more columns individually as a Header Field when you import the CSV file as an array through an FTP connector.

You can designate columns of data within the CSV files as indicators of grouped data, thereby giving the flat CSV data a certain amount of structure. For example, a single CSV file might contain data representing multiple sales orders, with each sales order having multiple items. The CSV rows for a single sales order can be grouped together as long as the value in each of the designated group header fields remains the same. A row containing a designated header field with a different value indicates a new group. Using this feature, you can create more hierarchical inputs for your orchestrations.

When you choose a header field, the Header Object and Detail Object fields will appear on the left. The data set variable name is used for the array name. The array will contain one header object for each group of rows containing the designated header fields. Each header object will also contain an array of detail data taking the name of the detail object.

The following example shows the output generated by converting a CSV file that had six lines and five columns into an array that has three SO Header objects, each containing a different number of lines in the SO Grid array.

An orchestration can now iterate over the SO Array in a step to call another orchestration to create a sales order with an object as input. This example will result in three sales orders with three, one, and two lines, respectively.

```
{
"SO Array": [
{
"SO Header": {
"Customer": "4242",
"Branch": "30",
"SO Grid": [
{
"ID": "1",
"Item_Number": "210",
"Quantity_Ordered": "5"
},
{
"ID": "2",
"Item_Number": "210",
"Quantity_Ordered": "4"
},
{
"ID": "3",
```



```
"Item_Number": "210",  
  
"Quantity_Ordered": "3"  
  
}  
  
]  
  
},  
  
{  
  
"SO Header": {  
  
"Customer": "4243",  
  
"Branch": "30",  
  
"SO Grid": [  
  
{  
  
"ID": "4",  
  
"Item_Number": "220",  
  
"Quantity_Ordered": "2"  
  
}  
  
]  
  
}  
  
},  
  
{  
  
"SO Header": {  
  
"Customer": "4242",  
  
"Branch": "20",  
  
"SO Grid": [  
  
{  
  
"ID": "5",  
  
"Item_Number": "210",  
  
"Quantity_Ordered": "2"  
  
},  
  
{  
  
"ID": "6",
```

```
"Item_Number": "210",  
  
"Quantity_Ordered": "1"  
  
}  
  
]  
  
}  
  
}  
  
]  
  
}
```

Configure an FTP Connector to Read Directory Content Attributes (Release 9.2.8.3)

You can create an Orchestrator FTP connector to read the file attributes in a given directory in an FTP site. By using this type of a connector, the orchestrations can automatically read and iterate over a list of files in a directory even if the file names change or are unknown at the time the orchestration runs.

1. Click the **Connectors** icon on the Orchestrator Studio Home page.
2. On the Connectors side panel, click the **New** drop-down list and select FTP.
3. Create and name the connector service request as described in [Creating a Component](#).
4. On the Connectors design page, click the **FTP** field and select a connection to the server from which you want to get the directory contents.

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

5. Select the **Directory Contents** option.
6. In the **Path Extension** field, enter the name of the subdirectory on the third-party FTP server from where you want to read the file attributes.

The system displays the path starting at the root directory next to this field.

7. In the **Data Set Variable Name** field, enter the name for the data set array.
8. In the **Attributes** section, select all the options you want the system to return. The available attributes are Name, Type, Date, Length, Path, Absolute Path, Is Directory, Is File, and Is Link.

Note: The system returns only the attributes of the files. Attributes of the sub-directories and the links in the folder are not included.

9. Click **Save**.

Configuring a Custom Service Request

This section contains the following topics:

- [Configuring a Custom Service Request with Java \(Advanced\)](#)
- [Configuring a Custom Service Request with Groovy \(Advanced\)](#)
- [Configuring a Custom Business Function Service Request \(Release 9.2.4.3\)](#)

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 [Creating Custom Java for Orchestrations](#)

1. Click the **Custom Requests** icon on the Orchestrator Studio Home page.
2. On the Custom Requests side panel, click the **New** drop-down menu and select **Java**.
3. Create and name the custom request as described in [Creating a Component](#).
4. On the Custom Requests design page, in the **Fully Qualified Class** field, enter the Java class.

This is the custom Java class that you created for the service request.

5. In the grid, complete the following fields:
 - o **Input.** Enter the name of the field in the custom Java class.
 - o **Value.** Enter the input variable name. If the input variable name has a Boolean attribute and you pass in "true", then you could use a default value.
 - o **Default Value.** Enter a default value if there is no mapped value. You can also add a default value to use if the mapped value returns a null value.
6. (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.
7. If you make a mistake while configuring any of the fields, you can click Restore from the Manage drop-down menu to return to the last saved state of the custom Java.
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.

Configuring a Custom Service Request with Groovy (Advanced)

You can create a service request that uses Groovy, JRuby, or Jython scripting languages to execute a custom process or to route data into another database. To use the script for a service request, the Orchestrator Studio provides an edit area that contains a sample script with instructions on how to modify the script. You can click the Show Shortcut Command icon located above the edit area to view the commands to work with the script. Chapter 10, ["Using Scripting Languages"](#) provides information on how to work with sample scripts.

Note: You can also configure a customer service request using JRuby and Jython. See [Installing Optional Scripting Languages on the AIS Server \(Release 9.2.5.4\)](#).

To configure a custom service request with Groovy:

1. Click the **Custom Requests** icon on the Orchestrator Studio Home page.
2. On the Custom Requests side panel, click the **New** drop-down menu.
3. From the Script drop-down list, select **Groovy**.

You can also select JRuby or Jython as your scripting language.

4. Create and name the custom request as described in [Creating a Component](#).
5. On the Custom Requests design page, configure the script to perform the desired action.
6. 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 using the commented out example code.

7. Click the **Load Outputs** button.

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

8. (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 service request name appears in the orchestration inputs grid. As a result, the returned value is 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.

9. To test the script:
 - a. Enter a value in the **Test Value** column for one or more inputs. If there is any syntax error in the script, details of the error will be displayed in a pop-up window.
 - b. Click the **Test** button.

The Orchestrator Studio executes the script using the inputs you entered. If the execution is successful, the system 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 dialog box is displayed after execution. If you see the warning log, but not the debug log, it indicates that the logging level is set-up to the error level and not the debug level. At runtime, log statements are included in the AIS Server log. These statements can be used for debugging script issues.

10. Click the **Save** button.

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

Configuring a Custom Business Function Service Request (Release 9.2.4.3)

To create a custom service request to call a business function.

1. Click the **Custom Requests** icon on the Orchestrator Studio home page.
2. On the Custom Requests side panel, click the **New** drop-down menu and select **Business Function**.
3. Create and name the custom request by following the steps described in *Creating a Component*.
4. On the Custom Requests design page, in the **Function Object Name** field, enter the name of the business function object.
5. From the **Function Name** drop-down list, select a business function.

The Orchestrator Studio loads all the fields included in the template for the selected business function into the grid.

6. Enable the **Fire and Forget** toggle if you want the business function to execute without waiting for a response. If the Fire and Forget toggle is enabled, the output columns are not editable.

(Release 9.2.4.4) The **Number of Threads** option is displayed when you enable the Fire and Forget toggle. This option enables you to select the number of threads for an iterated step.
7. Use the following columns and toggle options in the grid to configure the business function service request:
 - o **ID, Name, and Description** (informational only)

This column displays the ID, name, and description of the field in the business function.
 - o **Required** (informational only)

This column indicates if the field is set as a required field.
 - o **Input**

Enable the Input toggle to mark the field as the input to the business function service request.
 - o **Input Variable**

For each field to which you want to map an orchestration input, enter a variable name in the Input Variable column. When you add this custom request to an orchestration, you map the orchestration input to this variable.
 - o **Default Value**

Use this column to enter a literal value as the input.
 - o **Return**

Enable the Return toggle for the fields that contain values that you want returned in the orchestration response.

Note: The Return toggle is available only for the fields that allow output.
 - o **Output Variable**

Use this column to enter a variable name for the return value. When you add the business function custom request to an orchestration, the variable name appears in the orchestration inputs grid. As a result, the returned value is available for mapping to the subsequent steps in the orchestration.

In each row, the Output Variable column is editable only if the Return toggle is enabled.
8. Click the **Save** button.

The Orchestrator Studio saves the business function service request. You can then add this custom service request as a step in the orchestration to call a business function directly from the orchestration.

Configuring a Custom Service Request to Create an Array (Release 9.2.7.4)

You can create a custom service request that creates an orchestration array from a CSV file or a JSON array.

This custom service request enables you to mass-process data by consuming CSV files and JSON array data in orchestrations. You can pass the CSV or JSON to an orchestration called from an EnterpriseOne page, a form extension, a mobile device, or any client that can pass a CSV or JSON array as input or get it from another orchestration step.

To configure a custom service request with an array:

1. Click the **Custom Requests** icon on the Orchestrator Studio Home page.
2. On the Custom Requests side panel, click the **New** drop-down menu and select **Create Array**.
3. Create and name the custom request as described in *Creating a Component*.
4. Select one of these options:

- o **CSV File**

Complete the following fields:

- i. **Data Set Variable Name:** Enter a name to refer to the array created from the CSV data. Later, you can use this data set variable name as a reference to map data in the data set to a subsequent orchestration step.
- ii. **Delimiter:** Enter the delimiter used in the CSV file to separate values.
- iii. **CSV Has Headers:** If the CSV file has a header row, enable this option to exclude the header row. This action ensures that at runtime, the Orchestrator will not import the column headers as data in the array.
- iv. **Sample File:** Click this button to upload a sample CSV file. This will auto-populate the column table with a row for each column in the CSV. If the CSV has headers, the header value is used for each column name. You can click the X icon to delete a row in the grid.
- v. **Column Table:** Each row in this table represents a column in the CSV file. The Header Field option allows you to identify columns to break on. If you select one or more columns as a Header Field, you will need to specify Header Object and Detail Object to reference the array of objects created based on the level break.

- o **JSON File or JSON String**

Complete the following fields:

- i. **Data Set Variable Name:** Enter a name to refer to the array created from the JSON array. Later, you can use this data set variable name as a reference to map data in the data set to a subsequent orchestration step.
- ii. **JSON Path to Array:** Enter the path to the array using JSON dot notation. The grid displays the column names from the CSV file in the Member Name and Variable Name columns. You can click the X icon to delete a row in the grid.
- iii. **Column Table:** The Member Name column should match a value inside the JSON array. The Variable Name is used to reference the column in the orchestration.

5. Click the **Save** button.

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

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

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

Note: [Click here to view an OBE of this feature.](#)

Creating Connection Soft Coding Records for Connector Service Requests

This section contains the following topics:

- *Understanding Connection Soft Coding Records*
- *Creating a Soft Coding Record for an Orchestrator Connection*
- *Creating an Open API Connection*
- *Creating a Soft Coding Record for a REST Connection*
- *Creating a Soft Coding Record for a Database Connection*
- *Creating a Soft Coding Record for an FTP Server Connection*
- *Creating a Soft Coding Record for an Enterprise Server AIS Connection (Release 9.2.5)*
- *Authenticating Orchestrations Through Public Key Certificates (Release 9.2.4.4)*

Understanding Connection Soft Coding Records

Connector creates soft coding records to provide a secure access to external resources, such as a REST service, a database, or an orchestration or a 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 resources on an 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.
- **Open API Endpoint.** A connection to an external system where the documentation of external REST services (Open API - 2.0) reside. Open API connectors only support the OpenAPI 2.0 standard.
- **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. REST connections also support the transfer 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.
- **FTP.** A 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 or SFTP.

When you create a connection soft coding record, you associate it with an EnterpriseOne user, role, or *PUBLIC. This association enables the Orchestrator Studio user who is creating a connector service request to see the REST service, database, or orchestrations that are available through the connection. The user must be authorized to run the originating orchestration, which is the orchestration on the local system that will call the external orchestration, REST service, or database.

A soft coding record also includes the credentials of the user who is authorized to invoke the external resource, such as an orchestration, REST service, database, or directory on the external system.

Creating a Soft Coding Record for an Orchestrator Connection

1. On the Orchestrator Studio Home page, click the **Connections** icon.
2. On the Connections side panel, click the **New** button and select Orchestration from the drop-down list.
3. On the Connection design page, 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 respective location in the description of each of the soft coding records

The Type of connection is displayed next to the Description field.

- **User/Role.** Enter the user who is authorized to run the originating orchestration, which is 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 of the Orchestrator directory on the AIS Server where orchestrations are stored, for example:

```
https://<aisserverdomain>:<port>/jderest/orchestrator
```

Note: Oracle strongly recommends that all external calls use the SSL protocol (https) with a certificate from a reputable certificate authority. Oracle recommends using SSH file transfer protocol, otherwise referred to as Secure FTP (sFTP). The system does not support FTPS.

5. On the **Security** tab, enter the credentials for accessing the external system.
 - Security Policy

Select the authentication type that is 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 existing environment and role that are 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 the environment JDV920C1 and the role ADMIN.
 - Override Role
6. If the connection is being made 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. Slid the toggle to the right to enable the proxy.
7. Click **Save**.

Creating an Open API Connection

1. On the Orchestrator Studio Home page, click the **Connections** icon.
2. On the Connections side panel, click the **New** button and select **Open API** from the drop-down list.
3. On the **Connection** design page, complete these fields:
 - Name. Enter a unique name for the Open API 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.

The Type of connection is displayed next to the Description field.
 - User/Role. Enter the user who is authorized to access the connection. The user can be an EnterpriseOne user, role, or *PUBLIC.
 - Environment. Enter the environment where the connections reside.
4. On the **Service Information** tab, in the OpenAPI Catalog field, enter the catalog URL.
5. The Endpoint field contains the execution endpoint of the API. This field is automatically populated from the API definition. If there are multiple values, you must enter the correct endpoint. The system displays the first endpoint by default.

This endpoint is used during the execution time and the catalog endpoint is used during the design time.

The values for the Title, Description, Version, and Documentation fields are displayed automatically.

Note: Open API connectors only support the OpenAPI 2.0 standard (Swagger 2.0).

Note: As of release 9.2.5, Open API connectors support the OpenAPI 3.0 standard and the OpenAPI 2.0 standard.

6. On the **Security** tab, enter the credentials for accessing the external REST services.
 - Security Policy

Select the authentication type as Basic Authorization for accessing the Server Manager Console.
 - User
 - Password
7. When a proxy server is present, connection is made through that 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. Slide the toggle to the right to enable the proxy.
8. Click **Save**.

Note:

- Server Manager REST services can be hosted on the same server on which the Server Manager Console is hosted or on a different server.
- The credentials that are used for accessing the REST services and the Open API documentation are same as mentioned in the Security section.

Creating a Soft Coding Record for a REST Connection

1. On the Orchestrator Studio Home page, click the **Connections** icon.
2. On the Connections side panel, click the **New** button and select **REST** from the drop-down list.
3. On the **Connection** design page, complete these fields:
 - Name. Enter a unique name for the REST 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 respective location in the description of each of the soft coding record.
The Type of connection is displayed next to the Description field.
 - User/Role. Enter the user who is authorized to run the originating orchestration, which is 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.

Note: Oracle strongly recommends that all external calls use the SSL protocol (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:

This type of security allows orchestrations to exchange REST calls and data with Oracle Cloud services and other third-party systems.

 - i. In the Token Endpoint URL field, enter the address of the server that will provide the OAuth token.
 - ii. In the OAUTH Client ID field, enter ID of the client.
 - iii. In the OAUTH Secret field, enter the value for client secret you have received after registering the application.
 - iv. If the client requires additional header parameters, in the OAUTH Parameters area:
 - a. Click **Add**. The system adds a new row in the OAUTH Parameters table.
 - b. Enter a name and value for the parameter in the table.
 - v. (Release 9.2.5.2) Enable the **Use Parameters** option if you want to send the client ID and client secret as form parameters in the request body. Do not enable this option to pass the client ID and client secret in the Basic Authentication header.

- o (Release 9.2.8.2) For OCI API Key-Based Authentication:

Note: You must have permissions enabled in Oracle Cloud to access each of the Oracle Cloud Infrastructure (OCI) services you want to call. Each OCI service requires different permissions associated with it in OCI. Try to use the OCI web interface to run the service first, and then the system will display a message if additional permissions are required to use that service. Use this security policy to authenticate to external services provided by Oracle Cloud Infrastructure (OCI). This security policy enables the use of Oracle Cloud Infrastructure API Signature Version 1 to authenticate to services such as Oracle Document Understanding. To know more about the available OCI services, see <https://docs.oracle.com/en-us/iaas/api/>

- i. In the OCI Configuration File field, copy and paste the OCI configuration file details from your OCI account.
- ii. In the Private Key File Name field, drag the OCI private key (.pem) file from your system and drop it on the field name, or click and then upload the key file.

Note: You can generate and download an OCI private key in your OCI user account from Oracle Cloud.

- iii. Click **Save**. The system adds the OCI authentication information as record in the Soft Coding Record Table (F954001).

6. If the connection is being made to an external server, on the **Proxy** tab, specify a proxy server for accessing the external server:
 - o Proxy Host. Enter the URL of the proxy server.
 - o Proxy Port. Enter the port number of the proxy server.
 - o Use Proxy. Slide the toggle to the 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. Click **Save**.

Creating a Soft Coding Record for a Database Connection

Note: For a database connection to work, the JDBC driver for the database must be accessible in the AIS Server class path.

1. On the Orchestrator Studio Home page, click the **Connections** icon.
2. On the Connections side panel, click the **New** button and select **Database** from the drop-down list.
3. On the **Connection** design page, complete these fields:

- Name. Enter a unique name for the database 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 respective location in the description of each of the soft coding records.
The Type of connection is displayed next to the Description field.
 - User/Role. Enter the user who is authorized to run the originating orchestration, which is 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** area, complete these fields:
- Connection. Enter the URL to the database.
Note: 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 that the driver is specified on the AIS Server class path.
5. Click **Save**.

Creating a Soft Coding Record for an FTP Server Connection

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 Orchestrator Studio Home page, click the **Connections** icon.
2. On the Connections side panel, click the **New** button and select **FTP** from the drop-down list.
3. On the **Connection** design page, complete these fields:
 - Name. Enter a unique name for the FTP 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 respective location in the description of each of the soft coding records.
The type of connection is displayed next to the Description field.
 - User/Role. Enter the user who is authorized to run the originating orchestration, which is 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** 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.
 - Connection Type. Choose the connection type. Valid values are Anonymous, Username/Password, and SSH Key.
 - User. Enter the user who is authorized to access the directory on the FTP server.

- Password. Enter the password of the user who is 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.
 - Passphrase. If you chose SSH Key as the connection type, this field appears. If the associated OpenSSH key file requires a passphrase, enter the passphrase here.
 - SSH File. If you chose SSH Key as the connection type, this field appears. You can drop a OpenSSH key file into the box to upload the file.
5. Click **Save**.

Creating a Soft Coding Record for an Enterprise Server AIS Connection (Release 9.2.5)

1. On the Orchestrator Studio Home page, click the **Connections** icon.
2. On the Connections side panel, click the **New** button and select **Enterprise Server AIS** from the drop-down list.
3. On the **Connection** design page, complete these fields:
 - Name. The value **AIS_CONNECTION** is included in the software.
 - 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 respective location in the description of each of the soft coding records.
The Type of connection is displayed next to the Description field.
 - User/Role. Enter *PUBLIC if you are configuring one AIS server for the environment. Alternatively, a specific user ID or role can be used to route requests to another AIS server for that user or role.
 - Environment. Enter the environment that the AIS server being configured is associated with.
4. On the **Enterprise Server AIS** tab, in the Endpoint field, enter the URL of the AIS server, for example:
https://<ais_server>:<port>
5. Click **Save**.

Note:

You can copy this connection to another User/Role or Environment. The name of the connection will be saved as **AIS_CONNECTION** as the value of the Name field is included in the software.

Authenticating Orchestration Through Public Key Certificates (Release 9.2.4.4)

Authentication to an external REST service is established through X.509 public key certificates to securely integrate the EnterpriseOne Orchestrator with third-party systems and cloud services.

Using X.509 for an External REST Connector in Orchestration

Perform these steps to use an X.509 Client Certificate while invoking an external REST call through an orchestration:

1. Ensure that your certificate is in the following two Java keystore formats:
 - Java KeyStore (JKS)

- Public-Key Cryptography Standard (PKCS12)

Note: You can use external tools such as Java keytool or OpenSSL to convert your certificate to one of the accepted formats. You can use these tools to convert your certificates from the formats such as DER (.crt, .cer, or .der) or PEM (.pem) along with their key files into JKS (.jks) or PKCS12 (.pfx, .p12) formats.

2. Create a REST connection and include the location of the keystore file and the file password.
3. Use this REST connection to create a connector to call the external REST service.

Creating a REST Connection with a Certificate

1. On the Orchestrator Studio Home page, click the **Connections** icon.
2. On the Connections side panel, click the **New** button and select **REST** from the drop-down list.
3. On the **Connections** design page, 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 respective location in the description of each of the soft coding records.

The type of connection is displayed next to the Description field.

 - User/Role. Enter the user who is authorized to run the originating orchestration, which is the orchestration on the local system and calls 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 service URL.
5. Click the **Security** tab and enter the security settings that are required for the service.

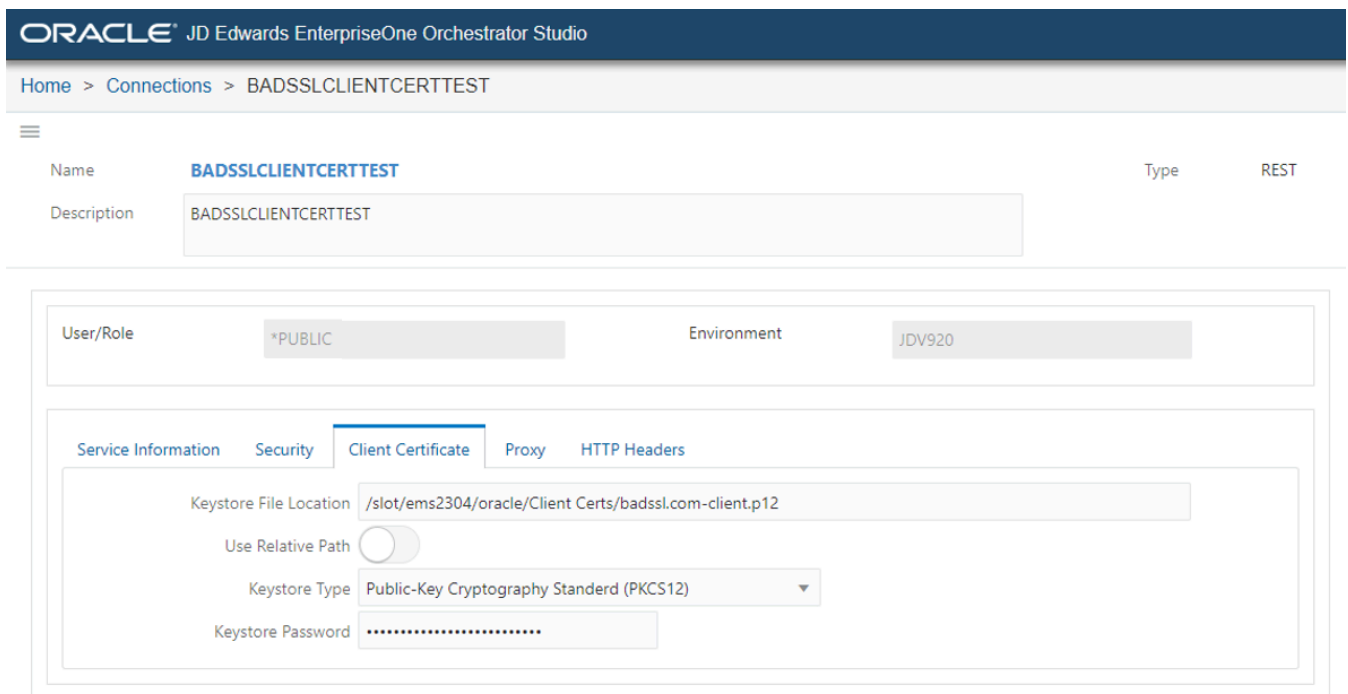
6. Click the **Client Certificate** tab and complete these fields:

- Keystore File Location. Enter the fully qualified path to the keystore on the AIS server where the orchestration is running.

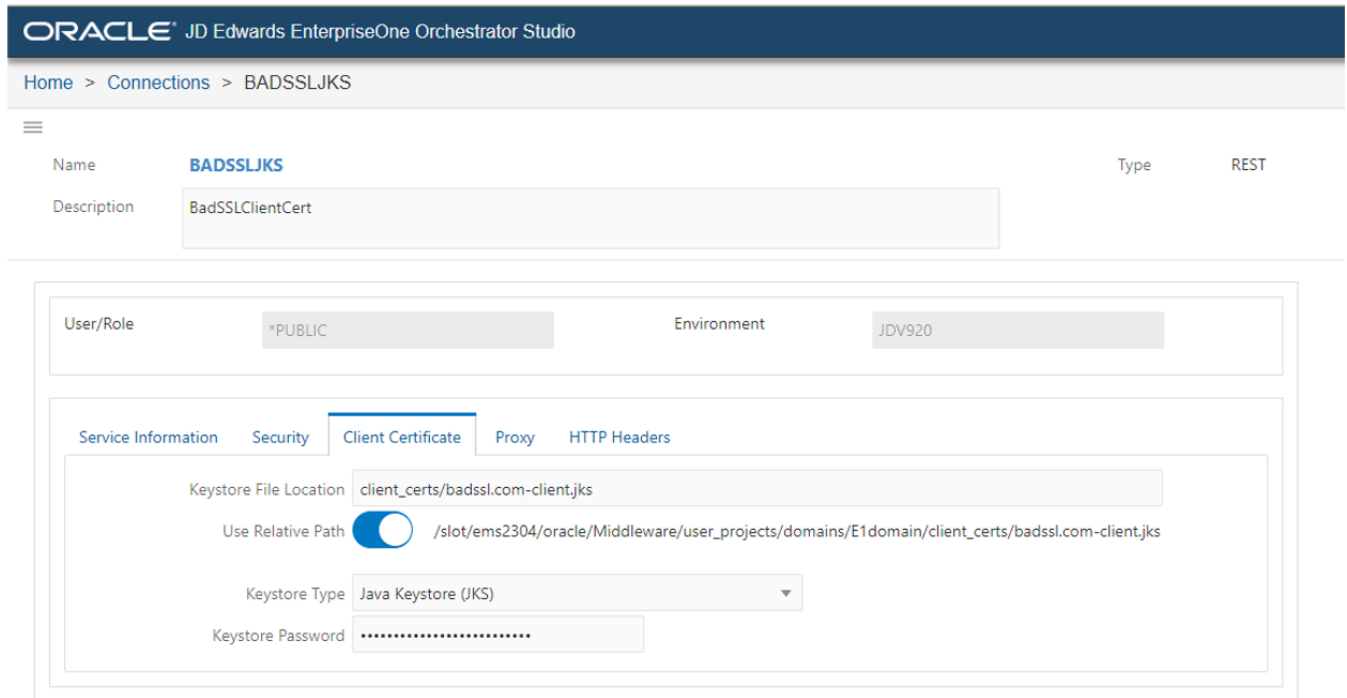
Or

Enter the relative path from the domain root of the AIS server. If you want to choose this option, you must slide the Use Relative Path toggle button to the right. The system displays an example of where the relative path is located on the AIS host server when you enable the Use Relative Path option.

- Keystore Type. Enter the type of the keystore file. For example,
 - PKCS12 (.p12, .pfx)
 - JKS (.jks)
- Keystore Password. Enter the password of the keystore.



Or



7. On the **Proxy** tab, specify a proxy server for accessing the external server if required.
8. On the **HTTP Headers** tab, enter the details to add a header record if required.
9. Click the **Save** button to save the record.

Troubleshooting

With AIS deployed on WebSphere, if you run a REST connector with a configured client certificate, the system generates the following error message:

```
"message": {
  "statusCode": 500,
  "error": "java.io.IOException: Received fatal alert: protocol_version"
}
```

To resolve this issue, you must configure the AIS server instance in WebSphere to specify the correct TLS version. Add the following settings in the Generic JVM arguments field in Java Virtual Machine.

- Djdk.tls.client.protocols=TLSv1.2
- Dhttps.protocols=TLSv1.2

Creating Rules

This section contains the following topics:

- [Understanding Rules](#)
- [Creating a Rule](#)
- [Creating a Custom Rule with Java \(Advanced\)](#)
- [Creating a Custom Rule with Groovy \(Advanced\)](#)

Understanding Rules

A rule contains conditional logic that the Orchestrator uses to evaluate conditions, and determine whether the conditions are true or false. These conditions determine how the orchestration processes the incoming data. You can define a simple 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 True or False nodes in the orchestration path 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.

Creating a Rule

Create a rule to define conditions for an orchestration.

To create a rule:

1. On the Orchestrator Studio Home page, click the **Rules** icon.
2. On the Rules side panel, click the **New** drop-down menu and select **Standard**.
3. Create and name the rule as described in [Creating a Component](#).
4. On the Rules design page, enter a name for the rule in the Rule field. Do *not* include special characters in the name.
5. In the Match Type drop-down menu, select one of the following values:
 - **Match All**. Select this option if all the conditions in the rule must be met.
 - **Match Any**. Select this option if any of the 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, Boolean, 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: >, <, >=, <=, =, !=, starts with, ends with, contains, is between, is in list. The "is in list" operator must be a string of values delimited by the pipe character (|). Example values: v1|v2|v3.

- o Literal. Slide the toggle to the right to indicate a literal value.
- o Value 2. If you have enabled the Literal toggle, manually enter a value.

Starting with Tools Release 9.2.6.3, from the Value 2 field drop-down menu, you can select:

- **<Null>** if the Literal option is enabled and the value in the Operator field is = or !=.
- **<Blank>** if the Literal option is enabled and the Rule Type is selected as String.

A string is considered blank if it is an empty string ("") or if it contains only one or many spaces.

- o Literal Value Type. If you have enabled the Literal toggle, click the drop-down menu and select the format of the literal value: string, numeric, or date.
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.

A new rule is saved for the first time as a "Personal" UDO. Thereafter, you can use the UDO links 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.

Creating a Custom Rule with Java (Advanced)

You can create a 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 *Creating Custom Java*.

To create a custom Java rule:

1. On the Orchestrator Studio Home page, click the **Rules** icon.
2. On the Rules side panel, click the **New** drop-down menu and select **Java**.
3. Create and name the rule as described in *Creating a Component*.
4. On the Rules design page, 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.

5. Complete the following fields in the grid:
 - o Input. Enter the name of the field in the class.
 - o Value. Enter a variable for the input name. If the attribute is a boolean and you pass in "true", then you could enter a default value.
 - o Default Value. Enter a default value if there is no mapped value. You can also add a default 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 Restore from the Manage drop-down menu to retrieve the last saved state of the custom Java rule.
7. Click the **Save** button.

Creating a Custom Rule with Groovy (Advanced)

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

Note: You can also create a custom rule using JRuby and Jython. See *Installing Optional Scripting Languages on the AIS Server (Release 9.2.5.4)*.

To create a custom rule with Groovy:

1. On the Orchestrator Studio Home page, click the **Rules** icon.
2. On the Rules side panel, click the **New** drop-down menu and select **Groovy**.
3. Create and name the rule as described in *Creating a Component*.
4. On the Rules design page, configure the script to perform the desired action.

The Orchestrator Studio displays an edit area that contains a sample groovy script with instructions on how to work with the script. See *Using Apache Groovy for Custom Service Requests, Rules, and Manipulating Output* for more information about the sample Groovy script. Click the Show Shortcut Command icon (?) to view the commands to edit the script.

5. In the **Input** grid, enter the names of the inputs.
6. 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 pop-up is displayed after execution. At runtime, log statements are included in the AIS Server log. These log statements can be used for debugging script issues.

7. Click the **Save** button.

Creating Cross References

This section contains the following topics:

- *Understanding Cross References*
- *Creating a Cross Reference*

Understanding Cross References

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

Creating a Cross Reference

To create a cross reference:

1. On the Orchestrator Studio Home page, click the **Cross References** icon.
2. On the Cross References side panel, click the **New** button.
3. Create and name the cross reference as described in *Creating a Component*.
4. On the Cross References design page, in the **Object Type** drop-down list, select a cross-reference object type.

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

5. (Release 9.2.6.4) In the Third Party App ID field, enter the field name to further refine the mappings between EnterpriseOne and third-party systems.

You can use the Third Party App ID field to define multiple sets of cross-references and then distinguish them. For example, you might be integrating to two or more external systems that use the same inbound value to be cross-referenced. This option enables you to define mappings that may have the same EnterpriseOne or third party value, but have a different cross-reference value for each of the external systems.

This is an optional field and can be left blank. You can enter a literal value such as `systemA`, or you can enter a variable name using the `${variableName}` syntax. If you enter a variable name, then the variable name becomes a step input, and the orchestration calling it can either leave it blank for no filtering or pass in a value to perform filtering on the app id.

6. (Release 9.2.6.4) In the Cross Reference Direction section, you can select the Inbound or Outbound options to specify the direction of the cross-reference.

If you select the Inbound (third-party value) option, the cross-reference will return its corresponding EnterpriseOne value. Conversely, if you select the Outbound (EnterpriseOne value) option, the cross-reference will return its corresponding third-party value. This option enables you to map data between EnterpriseOne and external systems bidirectionally.

7. Complete the **Input Key (Third-party Value)** and **Output Key (EOne Value)** columns to map the inputs to EnterpriseOne fields:
 - o Input Key. The inputs can be one or many values. The inputs must correspond to the values or pipe (|) delimited values in the Third Party Value column in P952000. See *Setting Up Cross References and White Lists in EnterpriseOne (P952000)* for more information.
 - o Output Key. The outputs can be one or many values. The outputs must correspond to the values or pipe (|) delimited values in the EOne Value column in P952000. See *Setting Up Cross References and White Lists in EnterpriseOne (P952000)* for more information.
8. If you enter any value by mistake, you can click the **Remove** (X) button to delete the entry.
9. Click the **Save** button, which saves the cross reference 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 that are described in the User Defined Object (UDO) Features in the Orchestrator Studio section to move the cross-reference to the appropriate status.

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

Creating White Lists

This section contains the following topics:

- [Understanding White Lists](#)
- [Creating a White List](#)

Understanding White Lists

A white list contains a list of IDs that are permitted in the Orchestrator. When a white list is added to an orchestration, only inputs with IDs that are 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 field should have the value 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.

Creating a White List

1. On the Orchestrator Studio Home page, click the **White Lists** icon.
2. On the White Lists side panel, click the **New** button.
3. Create and name the white list as described in [Creating a Component](#).
4. On the White Lists design page, in the **Object Type** field, select the object type from the drop-down list.

The value you select 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, which is stored in P952000, is a group of records to which a name is assigned. 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 [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.

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

Creating Attachments (Release 9.2.6)

This section contains the following topics:

- Understanding Attachments
- Creating an Attachment

Understanding Attachments

Attachments are used to automate EnterpriseOne transactions by including the options to manage text attachments, add URL attachments, and upload or download file attachments (media objects) in transactions.

Note: *Click here to view a recording of this feature.*

Note: *Click here to view an OBE of this feature.*

Creating an Attachment

Using Attachments, you can perform the following actions for a specified key:

- List files, texts, and URLs.
- Get the text in a sequence or get all the text attachments.
- Add either a text or an URL type of attachment.
- Update the text.
- Upload a file from the temporary directory of your AIS Server.
- Download a file.
- Delete a file.

Listing an Attachment

To list file, text, or URL attachments:

1. On the Orchestrator Studio Home page, click the **Attachments** icon.
2. On the Attachments side panel, click the **New** button.
3. Create and name the attachment as described in *Creating a Component*.

4. Click the **Search** icon next to the Structure Name field, and then search for and select the structure name.

Depending on the value in the Structure Name field, the values in the Description and Key Value fields are automatically populated in the Keys table. You may delete the automatically populated variable in the Key Value field and enter a literal value.

Note: To determine the structure used in an EnterpriseOne application, access the required application using the Fast Path or open the application from the Navigator menu. Search for a record that contains attachments. In the Attachment Manager, click the (i) information icon to see the structure name in the Attachment Information section. For example, Attachment Information Name: GT0801 Key: 6001 Count: 9

5. In the Action section, select the **List** option.
6. In the Input section, select the required **Type**. The options available are: File, Text, and URL.
7. In the Output section, enter a value in the **Data Set Variable Name** field.

Note: This is an optional step. If you enter a variable name, the array will be available to iterate over in subsequent orchestration steps.

8. Click **Save**.

Getting a Text Attachment

To get a text attachment:

1. On the Orchestrator Studio Home page, click the **Attachments** icon.
2. On the Attachments side panel, click the **New** button.
3. Create and name the attachment as described in *Creating a Component*.
4. Click the **Search** icon next to the Structure Name field, and then search for and select the structure name.

Depending on the value in the Structure Name field, the values in the Description and Key Value fields are automatically populated in the Keys table. You can delete the automatically populated variable in the Key Value field and enter a literal value.

5. In the Action section, select **Get Text**.
6. In the Input section, if you want to get all the text attachments, enable the **Get All** option.

If you do not want to get all the text attachments, enter the required value or a sequence in the **Sequence** field.

7. (Optional) In the Output section, enter a value in the **Data Set Variable Name** field.
8. (Release 9.2.7.2) Enable the **As HTML** option to return the text along with the HTML tags.

This option is disabled by default. When this option is disabled, the system returns the text without the HTML tags.

9. Click **Save**.

Adding Text or URL Attachment

Starting with Tools Release 9.2.7.3, you can use the text attachment editor to include rich formatting of text enabling you to use typefaces, emphasis, and colors of your choice in the text attachments. You can create layouts for the text that include links and images that are interspersed with text. You can also include tables and formatted content from external sources such as Microsoft Word.

To add a text or URL attachment:

1. On the Orchestrator Studio Home page, click the **Attachments** icon.
2. On the Attachments side panel, click the **New** button.
3. Create and name the attachment as described in *Creating a Component*.

4. Click the **Search** icon next to the Structure Name field, and then search for and select the structure name. Depending on the value in the Structure Name field, the values in the Description and Key Value fields are automatically populated in the Keys table. You can delete the automatically populated variable in the Key Value field and enter a literal value.
5. In the Action section, select the **Add** option.
6. In the Input section, these attachment types are available:
 - o **Text:** Select this option to add text. Enter the value or a variable in the Item Name field and enter the text or the variable you want to add in the **Text** field.

Note: If the Item Name field is left blank, the system assigns a default item name.

(Release 9.2.7.3) Enable the **Plain Text** option to include a plain text in the text attachment.

Disable the **Plain Text** option to add rich formatting to your existing text attachments. This option is disabled by default for new text attachments.

To format the text attachment:

- i. Enter the text in the area provided and use the options such as Paragraph, Bold, Italics, Underline, Strikethrough, Bulleted and Numbered List, and so on from the toolbar to format your text. You can use the Insert Table and Insert Image options to insert tables and images. Additionally, you can add a horizontal line in your text by clicking the Horizontal Line icon. To add a hyperlink to the text, highlight the relevant text, and then click the **Link** icon in the toolbar, enter the required address, and click the Save icon.
- ii. To use an existing template, click the Template button, select the template from the left-hand panel in the Templates window and place the template at the position you want by using the **Insert at Top**, **Insert at Cursor**, or **Insert at Bottom** buttons.

Note: The templates are created using the Work With Media Object Templates(P98TMPL) application.

Starting with Tools Release 9.2.7.4, the following new options are available in the editor:

- Subscript and the Superscript icons to add mathematical formulas. For example, $y = x^{2n} + z^{3n}$, $a_1 = a_2 + a_3$, and so on.
- List Properties option in the Numbered List drop-down to change the properties of the numbered list order.
- Page Break icon to end the current page and start a new page.

- o **URL:** Select this option to add an URL. Enter the URL you want to add in the **URL** field.

(Release 9.2.8.3) In the **Item Name** field, you can assign a name to a URL attachment that is different from the path to the URL itself.

7. (Optional) In the Output section, enter the value in the **Sequence Variable** field.
8. Click **Save**.

Updating Text Attachment

To update a text attachment:

1. On the Orchestrator Studio Home page, click the **Attachments** icon.

2. On the Attachments side panel, click the **New** button.
3. Create and name the attachment as described in *Creating a Component*.
4. Click the **Search** icon next to the Structure Name field, and then search for and select the structure name.
Depending on the value in the Structure Name field, the values in the Description and Key Value fields are automatically populated in the Keys table. You can delete the automatically populated variable in the Key Value field and enter a literal value.
5. In the Action section, select **Update Text**.
6. In the Input section:
 - a. In the Sequence field, enter the sequence number or a variable.
 - b. In the Item Name field, enter the value or a variable.

Note: If the Item Name field is left blank, the system assigns a default item name based on the name of the file.
 - c. Next to Text Action, select **Append** or **Replace** as required.
If you select the **Append** text action, the system displays the Append On New Line option.
 - d. (Release 9.2.7.2) Enable the **Append On New Line** option to append the new text in a new line.
This option is enabled by default. When this option is disabled, the new text is appended to the end of the last line.

Note: If you manually edit the text media object (in the EnterpriseOne Attachment Editor), the system always includes a new line at the end when the text is saved. Therefore, a subsequent append operation appears on a new line.
 - e. Enable the **Plain Text** option to include a plain text in the text attachment. In the Text field, enter the text or the variable you want to append or replace.
(Release 9.2.7.3) Disable the **Plain Text** option to add rich formatting to your text attachments. This option is disabled by default for new text attachments.
For more information on rich formatting of text attachments, see *Adding Text or URL Attachment*.
7. Click **Save**.

Uploading a File

You can upload a file from the session or the root temp directory of your AIS Server.

1. On the Orchestrator Studio Home page, click the **Attachments** icon.
2. On the Attachments side panel, click the **New** button.
3. Create and name the attachment as described in *Creating a Component*.
4. Click the **Search** icon next to the Structure Name field, and then search for and select the structure name.
Depending on the value in the Structure Name field, the values in the Description and Key Value fields are automatically populated in the Keys table. You can delete the automatically populated variable in the Key Value field and enter a literal value.
5. In the Action section, select **Upload File**.
6. In the Input section, enter the respective values or variables in the Item Name and File Name fields.
File Name is the name of the file as it resides in the session or root temp directory of your AIS server. It can be a variable, which allows you to dynamically pass in the file name from the orchestration input or from a previous step.

Note: If the Item Name field is left blank, the system assigns a default item name based on the name of the file.

- (Optional) In the Output section, enter the respective values in the **Sequence Variable** and **Item Name Variable** fields.

Note: If the Item Name Variable field is left blank, the system uses a default item name based on the name of the file.

- Click **Save**.

Downloading a File

You can download a file to the session or the root temp directory of your AIS Server.

- On the Orchestrator Studio Home page, click the **Attachments** icon.
- On the Attachments side panel, click the **New** button.
- Create and name the attachment as described in *Creating a Component*.
- Click the **Search** icon next to the Structure Name field, and then search for and select the structure name.

Depending on the value in the Structure Name field, the values in the Description and Key Value fields are automatically populated in the Keys table. You can delete the automatically populated variable in the Key Value field and enter a literal value.

- In the Action section, select **Download File**.
- In the Input section:
 - To download all the attachments, enable the **Download All** option.
 - In the Sequence field, enter the sequence number or a variable for the file you want to download.
 - Enable the **Overwrite Existing File** option to use the original file name for all the files throughout the processing.

If you do not enable this option, a unique name is assigned automatically to each file.

- Enable the **Keep Temp Files** option to save the files in the root temporary directory. The files will be saved in this directory indefinitely.

If you do not enable this option, the files are saved in the temporary directory of the user's session. These files are deleted when the session ends.

- (Optional) In the Output section, enter a value in the **File Name Variable** field.

Deleting an attachment

To delete an attachment:

- On the Orchestrator Studio Home page, click the **Attachments** icon.
- On the Attachments side panel, click the **New** button.
- Create and name the attachment as described in *Creating a Component*.
- Click the **Search** icon next to the Structure Name field, and then search for and select the structure name.

Depending on the value in the Structure Name field, the values in the Description and Key Value fields are automatically populated in the Keys table. You can delete the automatically populated variable in the Key Value field and enter a literal value.

- In the Action section, select the **Delete** option.
- In the Input section, enter a sequence number or a variable in the **Sequence** field.
- Click **Save**.

Creating Logic Extensions (Release 9.2.6)

This section contains the following topics:

- Understanding Logic Extensions
- Navigating the Logic Extensions Page
- Understanding the Logic Extensions Design Page
- Creating a Logic Extension

Understanding Logic Extensions

The Orchestrator Studio provides a web-based user interface to create logic extensions. Using logic extensions, you can create business logic to perform operations such as string manipulation, arithmetic calculations, conditions, loops, and even table I/O. Using the logic extensibility framework, you can use the familiar syntax of the JD Edwards Named Event Rules language to create custom logic. The logic extensions component can be added as a step in the orchestration.

Logic extensions are created and managed as user-defined objects (UDOs), and therefore they inherit all the lifecycle management and security capabilities of the UDO framework. Logic extensions are ready to run immediately as part of an orchestration.

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





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

Note: [Click here to view an OBE of this feature.](#)

Navigating the Logic Extensions Page

The Logic Extensions page displays a list of icons for creating logic extensions. Hover over each icon to view the associated description.

The Data Structure tab is displayed by default when you start creating a new logic extension. The left panel of the page contains the following tabs:

-  Data Structure
-  Variables
-  Logic
-  Test

Data Structure

In the Data Structure tab, you can define the inputs and outputs of the logic extension. You can add a data dictionary item, a Boolean(Release 9.2.6.3), an array(Release 9.2.7), or a control (Release 9.2.7.3) to your data structure.

Starting with Tools Release 9.2.7, the logic extensions can accept arrays as inputs and return arrays as outputs for efficient iterative processing. The input arrays could come from orchestration inputs or from the results of a previous orchestration step.

The Data Dictionary tab contains a table with these columns:

- **ID:** This field displays an automatically generated unique identification number for the data dictionary item.
- **Name:** This field displays the name of the data dictionary item. This is an editable field.
 - Note:** The maximum allowed length for a name is 26 characters. If you enter a name longer than 26 characters, the system truncates the name to the first 26 characters while processing.
- **Dictionary:** This field displays the data dictionary number.
- **Data Type:** This field displays the data type of the data dictionary item.
 - Note:** The following data types are allowed in the Data Structure tab: Character, Boolean, Date, Integer, String, Variable String, JDE UTime, Identifier, and Numeric.
 - Note:** For a control, the following selections are allowed: Form Control(default) and Form Control List.
- **Required:** Enable or disable this field as required.
- **IO Type:** Select Input Only, Output Only, or Both options in this field.
- **Output Variable:** If the IO Type is defined as Output Only or Both, the same value that is in the Name field is displayed by default in the Output Variable field. This is an editable field. You can change this value and use it as an identifier for output when you process the logic extension.

A Menu icon is displayed next to the ID field to indicate that you can drag and drop the data dictionary items to reorder them.

Variables

In the Variables tab, you can define variables to use in the logic extension. You can add a data dictionary item, a Boolean (Release 9.2.6.3), and an array (Release 9.2.7) as a variable.

The Variables tab contains a table with the following column names:

- **ID:**This field displays an automatically generated unique identification number for the variable.
- **Name:**This field displays the system generated data dictionary, Boolean, or Array name. This field is editable.
 - Note:** The maximum allowed length for a name is 26 characters. If you enter a name longer than 26 characters, the system truncates the name to the first 26 characters while processing.
- **Dictionary:**This field displays the data dictionary number.
- **Data Type:** This field displays the data type of the data dictionary item.
 - Note:** The following data types are allowed in the Variables tab: Character, Boolean, Date, Integer, String, Variable String, JDE UTime, Identifier, and Numeric.You can drag and drop the variables to reorder them.

Logic

Use the Logic tab to create the user-defined logic. This tab contains all the available logic statements. See [Understanding the Logic Extension Design Page](#)

Test

In the Test tab you can specify the input values and then click the **Test** button to see the output and response of the logic extension.

Home > Logic Extensions > untitled

Name: Demo
Description: This is for demo.
Product Code: 55 - Reserved for Clients
Category: [Empty]

Name	Input	Output
AddressNumber	enter test input value	
Boolean2	enter test input value	
Array3	True False	
Array4	[List Icon]	
Array6		[List Icon]
Array7	[List Icon]	[List Icon]

Response

(Release 9.2.6.3) When you test a logic extension that contains a Boolean data structure, the system enables you to select True or False in the Input field.

(Release 9.2.7) When you test a logic extension that contains an array, only the array parent (not the array columns) is displayed in the Test tab. Depending on the IO type of the array, you can see the list icon in the Input, Output, or both the fields to indicate an array. You can click the list icon to view the table with all the columns.

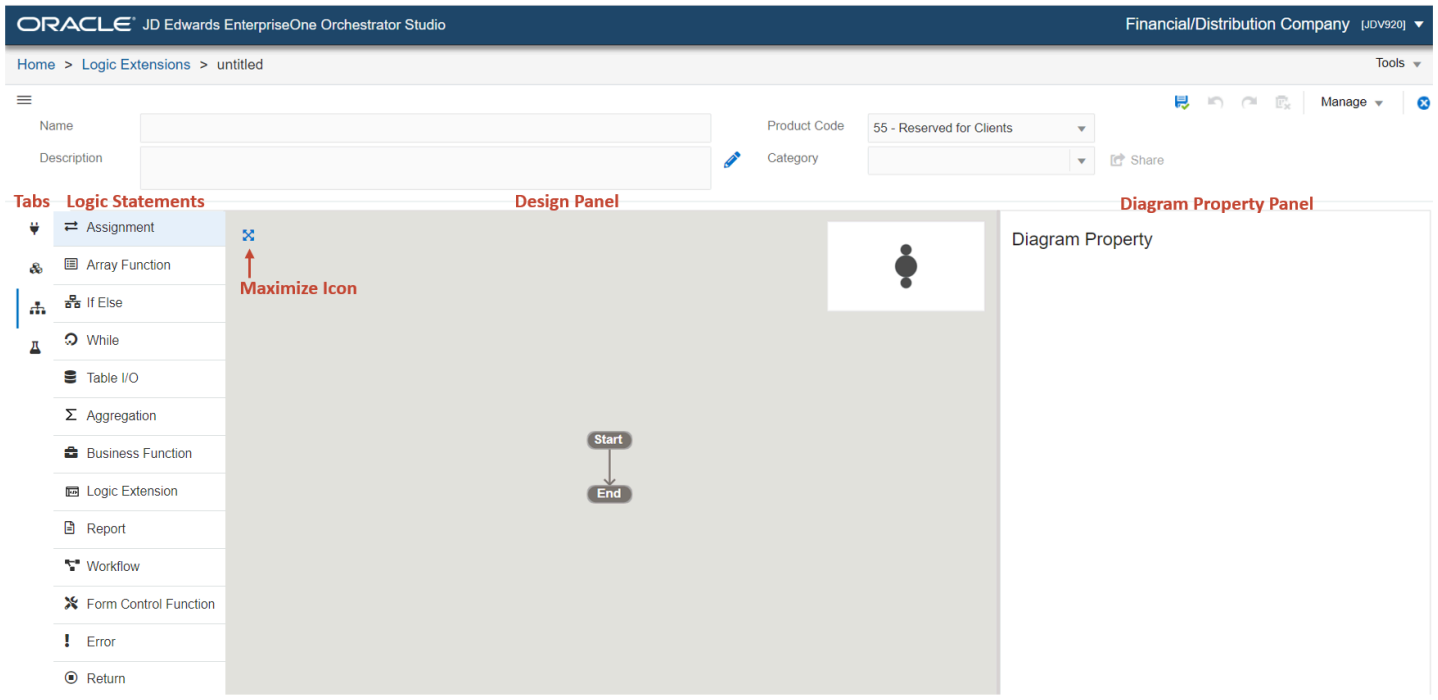
Depending on your logic extension, when you click the list icon in the Input field, the system displays an editable table or an empty table.

When you click the list icon in the Output field, the system displays a read-only table containing the data from the execution.

See [Testing the Logic Extension Using the Diagnostic Mode \(Release 9.2.6.3\)](#)

Understanding Logic Extensions Design Page

The Logic tab of the Logic Extensions page is used to design the logic extensions. You can see the following subpanels in this tab:



- **Logic Statements** : Use this panel to open the Data Structure, Variables, Logic, and the Test tabs.

- **Logic Statements:** This panel displays the available logic statements that you can use to create your logic extensions. You can drag and drop the logic statements from this panel to the desired location in the Design Panel.

The various types of logic statements include:

- Assignment block
 - Array Function (Release 9.2.7)
 - Conditional statements, such as If/Else
 - While loops
 - Table I/O operations
 - Aggregation (9.2.7)
 - Business Function (Release 9.2.6.3)
 - Logic Extension (Release 9.2.7.3)
 - Report (Release 9.2.6.3)
 - Workflow (Release 9.2.6.4)
 - Form Control Function (Release 9.2.7.3)
 - Error statements
 - Return statements
- **Design Panel:** This panel displays the flow chart representation of the logic extension. For a new logic extension, the system displays only a Start node and an End node. You can start building your logic extension by hovering over the line between the Start and End and clicking the + icon to insert a logic. You can also drag and drop the logic from the Logic Statements panel to the required location in the design panel.
 - **Diagram Property Panel:** The Logic Extensions page displays a Diagram Property panel at the right-hand side. This panel contains the following fields corresponding to the selected logic in the Design Panel:
 - Title: This is a read-only field. This field contains the name of the logic. For example, Assignment Block, Report, Table I/O, and so on.
 - Label: This is an editable field. It displays the title of the logic by default.
 - Description: Enter the required description in this field.

Note: For better understandability, it is recommended to change the label and add a description for all the logic statements in the Diagram Property panel.

Creating a Logic Extension

To create a logic extension:

1. On the Orchestrator Studio Home page, click the **Logic Extensions** icon.
2. On the Logic Extensions left panel, click the **New** button. The Data Structure tab is displayed.
3. Create and name the logic extension as described in [Creating a Component](#).
4. Follow the steps in the [Defining Data Structure](#), [Defining Variables](#), and [Building a Logic Extension](#) sections.

Defining Data Structure

In the Data Dictionary tab, you can add data dictionary items, Boolean, and arrays to the data structure.

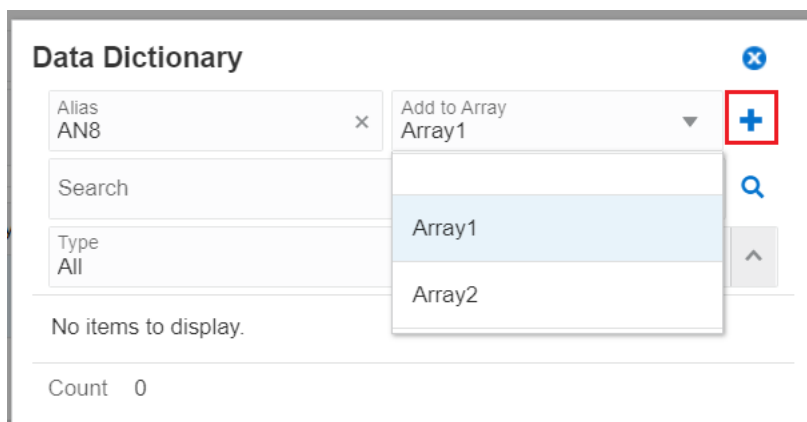
1. Access the Data Dictionary tab in the Logic Extensions page. This tab is displayed by default when you start creating a new logic extension.
2. To add a data dictionary item:
 - a. Click the **+ Data Dictionary** button.
 - b. In the Data Dictionary window, enter the required data dictionary alias in the Alias field and then click the **+** icon. You can also use the Search, Type, and Size fields to filter and find the required data dictionary item.
 - c. The Name field is editable. Change the value in the Name field if required.
 - d. The Required option for all the data dictionary items is disabled by default. Enable this option if required.
 - e. Select the value from the IO type drop-down list. If you select the IO type as **Output Only** or **Both**, you can enter an output variable name.
3. To add a Boolean:
 - a. Click the **+Boolean** button.
 - b. The Name field is editable. Change the value in the Name field if required.
4. (Release 9.2.7) To add an array:
 - a. Click the **+Array** button.

Note: When you add an array as a data dictionary item, the system adds an array variable in the `<Arrayname><id>.length` format in the Variables tab. These `<Arrayname><id>.length` variables are not editable but you can reorder them.

- b. The Name field is editable. Change the value in the Name field if required.
 - c. The Required option is disabled by default. Enable this option if required.
 - d. Select the value from the IO type drop-down list. If you select the IO type as **Output Only** or **Both**, you can enter an output variable name.
5. Click **Save**.

The system enables you to add members to the newly added array. When you add a data structure after adding an array, an option to add the data dictionary item to the array is displayed in the Data Dictionary window.

For example, see the following screenshot. When you add an AN8 data dictionary item after adding Array1 and Array2 in the Data Dictionary tab, you can see the Add to Array drop-down menu and you can choose to add AN8 to Array1 or Array2. To do so, select the required array and then click the Add "+" icon.



When you click the **+Boolean** button after adding an array, the system displays the Add to Array window. You can either choose the array name from the Array field to add the Boolean to an array or leave this field blank.

The values in the Required field and the IO type field for all the data dictionary items added to the array is inherited from the parent array. For example, if you define an array as Input Only and enabled the Required option, all the data dictionary items added to that array will have the IO type as Input Only and Required option enabled.

6. (Release 9.2.7.3) To add a control:

- a. Click the **+Add Control** button.
- b. The Name field is editable. Change the value in the Name field if required.
- c. The Required option is disabled by default. Enable this option if required.
- d. In the Data Type drop-down list, Form Control is selected by default. To select an array of form controls, click the Data Type drop-down list and select Form Control List.

7. Click **Save**.

To delete a data dictionary item, click the **X** icon at the end of a row.

Note: The system deletes all the data dictionary items with in an array when you delete an array type data dictionary item.

Duplicating a Data Dictionary Item

To duplicate a data dictionary item, you can click the drop-down menu next to the Name field and select either Duplicate as Data Structure, Duplicate as Variable, or Duplicate into Array (Release 9.2.7). The system automatically generates a unique ID with a value in the Name field and the Output Variable(if applicable) for the duplicated item. The Name and Output Variable fields are editable.

Note:

- You have to manually change the name for the new duplicated data structure to avoid the duplicated name validation error.
- You cannot duplicate an array into an array. When you click the drop-down menu next to an array Name field, you will see the Duplicate as Data Structure and Duplicate as Variable options. The Duplicate into Array option is disabled.

Defining Variables

In the Variables tab, you can add data dictionary items, Boolean, and arrays as variables.

To add a data dictionary item as a variable:

1. In the Logic Extensions page, click the Variables tab.
2. To add a data dictionary item as a variable:
 - a. Click the **+ Data Dictionary** button.
 - b. In the Data Dictionary window, enter the required data dictionary alias in the Alias field and then click the **Add +** icon. You can also use the Search, Type, and Size fields to filter and find the required data dictionary item.

Note: (Release 9.2.7) The system displays the Add to Array drop-down menu in the Data Dictionary window when you add a data dictionary item after adding an array. You can choose to add the data dictionary item into an array by selecting the array name from the Add to Array drop-down menu.

- c. The Name field is editable. Change the value in the Name field if required.
3. To add a Boolean as a variable:
 - a. Click the **+Boolean** button.

Note: (Release 9.2.7) The system displays the Add to Array window when you add a Boolean after adding an array. From the Array drop-down menu, select an array name, and then click Ok. If you don't want to add the Boolean into an array, choose the blank option from the Array drop-down menu and then click OK.

- b. The Name field is editable. Change the value in the Name field if required.
4. (Release 9.2.7) To add an array as a variable:
 - a. Click the **+Array** button.
 - b. The Name field is editable. Change the value in the Name field if required.

Note: When you add an array as a variable, the system adds another array variable in the `<Arrayname><id>.length` format in the Variables tab.

5. Click **Save**.

Duplicating a Variable

To duplicate a variable item, you can click the drop-down menu next to the Name field and select either Duplicate as Data Structure, Duplicate as Variable, or Duplicate into Array (Release 9.2.7).

Note:

- You have to manually change the name for the new duplicated variable to avoid the duplicated name validation error.
- You cannot duplicate an array into an array. When you click the down arrow icon next to an array Name field, you will see the Duplicate as Data Structure and Duplicate as Variable options. The Duplicate into Array option is disabled.



Building a Logic Extension

In the Logic tab, you can add the logics to build your logic extension.











Note: You can add the logic statements such as the assignment block, conditional statements, table I/O, return statement, business functions, or reports in any order and any number of times depending on how you want to build your logic extension. The steps in this section explain how to add the logic statements in the order that they are displayed in the design panel, just for your understanding of the process. When you add a logic statement in the design panel, it is recommended to change the label and description of the logic using the Diagram Property panel for better understandability. Select **Create Orchestration** from the **Manage** drop-down menu to add your logic extension to a new orchestration.



Generic Functions

- You can either click the **+** icon between the nodes to add a logic statement using the context menu, or you can drag and drop from the logic statements panel to the required location in the design panel.
- You can also drag and drop the logic statements from one position to another in the design panel.
- You can resize the width of the Property Diagram panel by dragging the left side edge of the panel.

- You can maximize and minimize the design panel by clicking the Maximize or Minimize icon. If you add many logic statements in the design panel, the system automatically adjusts (zooms in) the design panel to display the full diagram (all nodes) of the logic extension.
- To cut, copy, delete, paste, or disable a logic, right-click the logic statement and then select the required option from the context menu. You can also select a logic and then click the Action menu in the Diagram Property panel to perform these operations.
- An error  icon is displayed, if the mapping is incomplete or there is an error when you create logic extensions.
- In the Diagram Property panel you can click the edit  icon to create mappings or build a criteria for the conditional logic statements.

This table lists the available objects in the mapping window:

Available Objects	Icons	Description
Data Structure		Add the data dictionary item. The system displays the list of data dictionary items added in the Data Structure tab.
Variable		Add the variable. The system displays the list of variables added in the Variables tab.
Array		Add an array.
Literals		Add a literal value.
System Literals		Add system literals such as user ID, language preference, country code, and so on.
Expressions		Add numeric expressions such as the plus sign (+), minus sign (-), multiplication symbol (X), and so on.
Math Functions		Enables you to perform various complex mathematical operations and calculations such as return the absolute value of a number, round a number up to the next whole number, return the value rounded by the rounding rule, and so on.
Text Functions		Add text functions to append two strings together, return the length of a string, convert a given text to lowercase, and so on.
Date Functions		Add number of days, add a specified number of months, return the number of days between dates, and so on.
System Constants		Add system constants to indicate a success, a warning, an error, and so on. For example, <code>add days (Date, Numeric)</code> , <code>add months (Date, Numeric)</code> , <code>date today()</code> , and <code>last day (Date)</code> .

System Variables		Add system variables for error status or file input/output status.
UTC Functions (Release 9.2.7)		Enables you to return date and time value adjusted for the time zone of the user. For example, <code>get year (JDE U Time, String)</code> , <code>get month (JDE U Time, String)</code> , <code>get day (JDE U Time, String)</code> , and <code>get hour (JDE U Time, String)</code> .

Adding an Assignment Block

Use an assignment block to assign a field with a fixed value or a mathematical expression. You can also use an assignment block to calculate a value.

Note: When you create an expression, calculate only the data items that are of the same numerical scale or data type. For example, do not calculate different currencies or decimal figures that represent different decimal values because the result might compromise data integrity.

You can access and modify the available objects in the assignment block to implement business logic. When you select or add an assignment block in the logic extension design page, the Assignment Block panel is displayed at the right hand side. You can click the **Launch Assignment Mapping Wizard** icon next to Assignments to open the Assignment Builder window and using this window you can map the target to the source. The target can be a data structure item, a variable, or a Boolean. The source can be a Boolean data type data structure or a variable, or a literal (if the target is Boolean).

To add an assignment block:

1. On the design panel, hover over the location where you want to add an assignment block and then click the **+** icon.
2. Select **Assignment** from the Action menu. The system displays the Assignment Block panel on the right.
3. In the Assignment Block panel, edit the label and enter the description for your assignment block.
4. Click the **Assignment Mapping Wizard** icon next to Assignments. The Assignment Builder window is displayed.

The Assignment Builder window contains two panels:

- Assignment Panel (left panel): Displays an assignment table with the target and source information.
 - Object Selector Panel (right panel): Filter and select the target and source values using this panel. You can click the Close or Open icons to hide or view this panel.
5. In the Assignment Builder window, in the left panel, click the **Target** icon and select a value from the right panel using the Quick Selection drop-down menu or click the Data Structure, Variables, or Literal icons to select the required value.

Once you select the Target, the system automatically highlights the Source icon in the right pane.

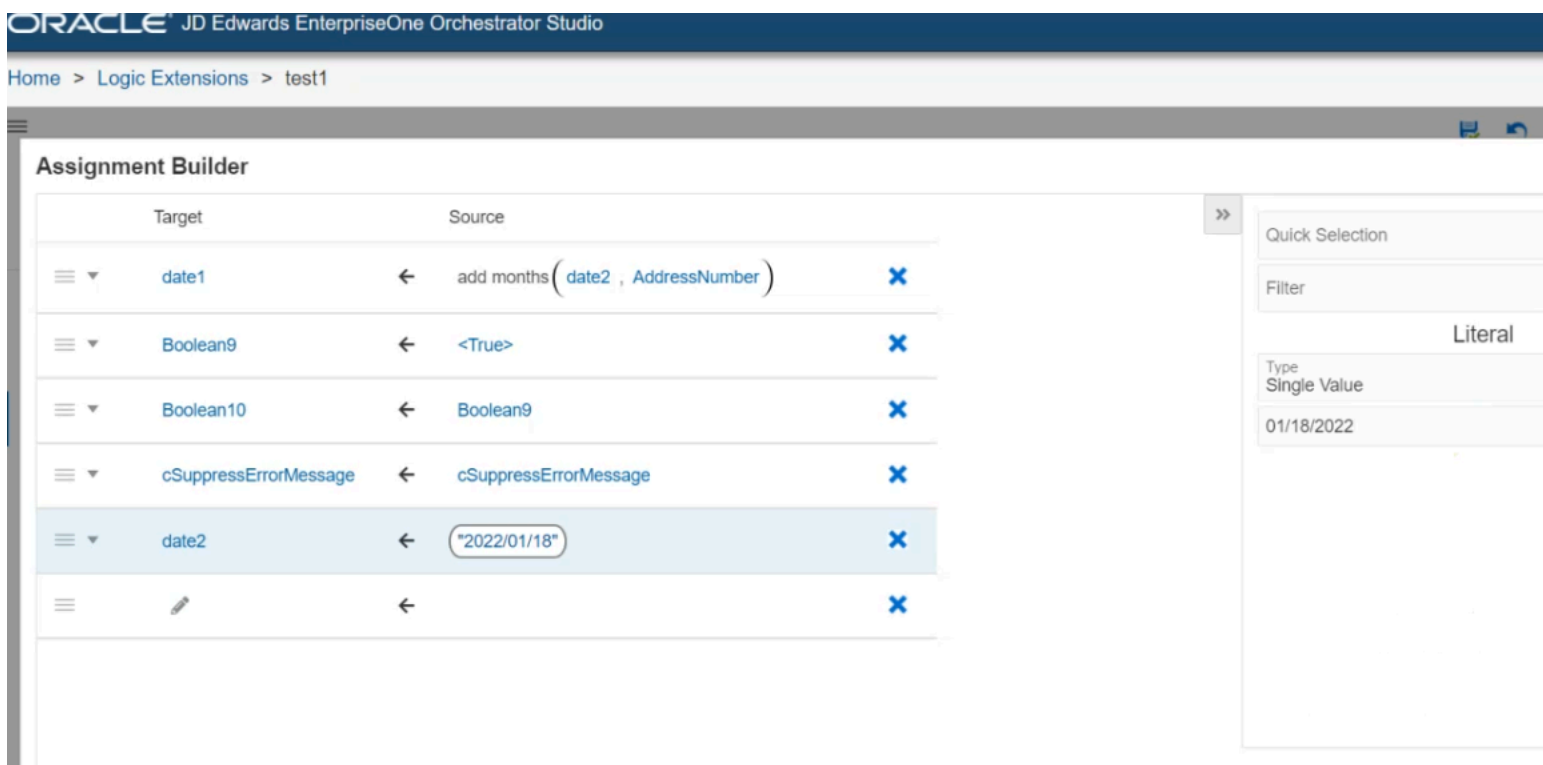
6. Select the required values or functions for the Source using the right panel. You can select values using the Quick Selection drop-down menu or click the Data Structure, Variables, or Literal icons to select the required value.

A new entry row is automatically added at the end of the assignment table if the table is empty or if the last row is completely defined.

Note: Repeat Steps 5 and 6 to add more mappings. If you want to delete a mapping, click the **X** icon in the left panel.

7. Click the **Close** icon to close the Assignment Builder window. Your mappings are displayed in the Assignment Block panel.
8. Click **Save**.

(Release 9.2.6.3) You can cut and copy an individual assignment from one assignment block and paste the assignment on another assignment block. In the Assignment Builder window, click the drop-down icon, and select the **Cut Assignment** or **Copy Assignment** option. Open the Assignment Builder window of another assignment block and then click the drop-down icon in the last assignment line and select the **Paste Assignment** option. You can then change the order of this assignment by dragging and dropping it on the required position.



Adding an Array Function (Release 9.2.7)

You can add multiple array functions to a logic extension. When you add an array function to the logic extension, the Array Function panel is displayed on the right.

The following array functions are available in the Array panel: get row, append row, and update row. You can also choose to add a system generated While statement template to process the array data in a loop.

To add an array function:

1. On the design page, hover over the location where you want to add the array function and then click the **+** icon. Select **Array Function** from the Action menu. The system displays the Array Function panel on the right.

2. In the Array Function panel, edit the label and enter the description for the array function.
3. In the Array drop-down menu, the system displays the arrays that are defined in the Data Structure tab. Select the required array. The system displays the columns of the selected array. See *Defining Data Structure*

Note: This is a required field.

When you select an array from the Array drop-down menu, the system displays the Array Function drop-down menu and the Iterate Array button.

4. From the Array Function drop-down menu, select the required function.
 - o get row: Use this function to load a particular row of data into the members of the array defined in the Data Structure or the Variables tab. When you select the get row function, the system displays the Index drop-down menu to control which row of the array is returned. This Index drop-down contains all the numbers available in the Data Structure and Variables tab for selection or you can also enter an integer for the value (the index of an array starts at zero).
 - o append row: Use this function to append a new row at the end of an array.
 - o update row: Use this function to update a row. When you select the update row function, the system displays the Index drop-down menu to control which row of the array is updated. The Index drop-down contains all the numbers available in the Data Structure and Variables tab for selection or you can enter an integer for the value (the index of an array starts at zero).
5. Without selecting an option from the Array Function drop-down menu, you can click the **Iterate Array** button to create a counter and a While loop to iterate over the array. This button enables you to add a system generated While statement template to process the array data in a loop. This template consists of a While statement with a counter, a get row array function, and an assignment block that increments the array counter. You can change or define mappings for this assignment block using the Assignment Builder window.

Note: When you click the Iterate Array button, a counter array with the name `<array name>Counter` is added in the Variables tab.

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

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

Adding an IfElse condition

You can use If, Else If, Else, and While statements as conditional instructions in a logic extension. These statements evaluate conditions and dictate the flow of logic when the logic extension is processed. When you create an If statement, the system inserts an Else clause.

When you insert or select a conditional statement in the logic extension design page, the system displays an If Else or While panel on the right. In the If Else or While panel, you can click the Launch Criteria Builder icon to open the Criteria Builder window where you can define the required logic for the conditional statements.

To add an If Else condition:

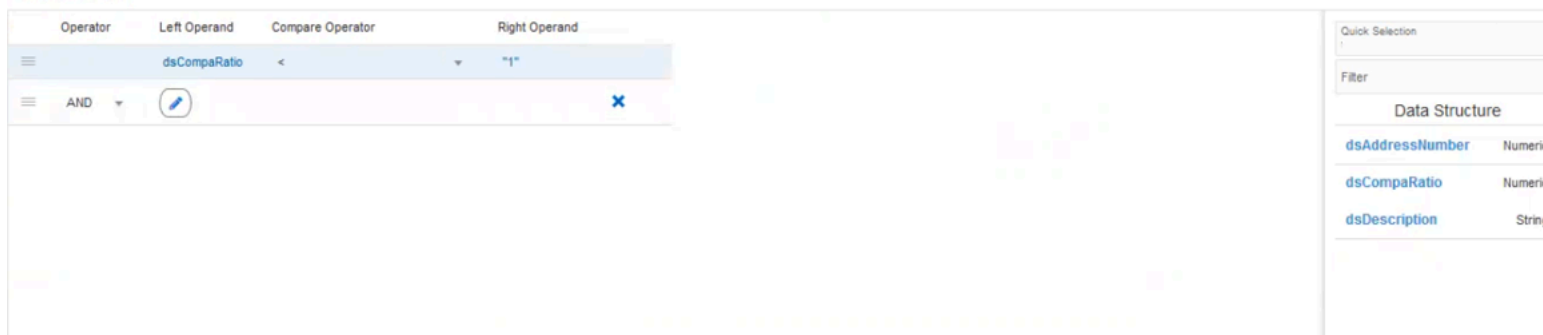
1. On the design panel, hover over the location where you want to add an If Else condition and then click the **+** icon. Select **If Else** from the Action menu. The system displays the IfElse panel on the right.
2. In the If Else panel, enter the description and appropriate labels for the **If Else**, **If**, and **Else** branch.

Note: To add an Else If condition, hover between the If and Else branch and click the **+** icon.

3. Click the **Launch Criteria Builder** icon. The system displays the Criteria Builder page containing two panels:
 - o Criteria Panel (left panel): Displays a criteria table with the Operator, Left Operand, Compare Operator, and Right Operand information.
 - o Object Selector Panel (right panel): Using this panel, you can filter and select the values for the Left Operand and the Right Operand fields. You can click the Close or Open icons to hide or view this panel.
4. In the Criteria Builder window, create your logic by selecting values in the Operator, Left Operand, Compare Operator, and Right Operand fields. Use the Quick Selection and Filter fields in the right panel to select and add values to your criteria.
5. Click **Close**.
Your mappings are displayed in the IfElse panel.
6. Click **Save**.

(Release 9.2.6.3) You can cut and copy the conditional branches from one position and paste them to another position in the logic extension. To do so, right-click the conditional branch and select the **Cut Branch**, **Copy Branch**, and **Paste Branch** options.

Criteria Builder



Note: Similarly, you can add a While loop condition to your logic extension.

Adding a Table I/O

Overview

Use the Table I/O option in the logic statements panel to create instructions that perform table input and output. The Table I/O feature enables you to access a table or a business view and perform the following tasks:

- Retrieve records.
- Update or delete records.
- Add records.

To add a Table I/O:

1. On the design page, hover over the location where you want to add Table IO and then click the **+** icon. Select **Table I/O** from the Action menu. The Table I/O panel is displayed on the right.
2. In the Table I/O panel, change the label and description of the Table I/O.
For Table I/O, by default, the Label field displays the selected operation (such as Fetch Single, Insert, Update, and so on) and the Description field displays the object name. You can override the automatically updated values in the Label and Description fields. But, if you change the values in the Operation and Object Name fields after overriding the values in the Label and Description fields, the system will not automatically replace

the new values in the Label and Description fields unless the overridden values are blank. If you change the Object Type from Table to View or vice versa, the Description field will be automatically populated with the value in the Object Name field even if it was previously overridden.

- From the Object Type field, select **Table** or **View**.

Note: If you select the View option, the system displays the Get View from Form icon. Click this icon and from the Get View from Form window, you can search and select the required application, form, and business view.

- Click the **Search** icon and then find and select the required object name.
- Select the required value from the **Operation** drop-down list. For more information, see *Available Operations*.
- Click the **Launch Column Mapping Builder** icon. The system displays the Column Mapping window containing two panels:
 - Column Mapping Panel (left panel): Displays the table with the Column, Filer, Return, and Mapped Object information.
 - Object Selector Panel (right panel): Filter and select the value for the Mapped Object field using this panel. You can click the Close or Open icons to hide or view this panel. See *Valid Mapping Operators*.
- Click the **Close** button.

Your mappings are displayed in the Table I/O panel.

- Click **Save**.

(Release 9.2.6.3) You can add the Table I/O parameters to the Data Structure or to the Variables tab from the Column Mapping window. Click the drop-down icon next to the Column field and select the **Add to Data Structure** or **Add to Variables** or **Add to Array (Release 9.2.7)** options as required.

Available Operations

This table describes the operations that you can perform using the Table IO feature.

Operation	Description
FetchSingle	Combines the Select and Fetch options in a basic operation. Indexed columns are used for the Select option, and non-indexed columns are used for the Fetch option. The operation opens a table for input/output but does not close it.
Insert	Inserts a new row.
Update and Retain (Release 9.2.7)	Updates data in columns that are mapped for input, and for which input is provided. If a column is mapped for input but input is not provided, the data is retained with its existing value. Use this operation when you want to update mapped columns and retain existing data from the remaining mapped columns. Unmapped columns are not modified.
Update and Clear (This option is displayed as Update in the releases prior to 9.2.7)	Updates data in columns that are mapped for input, and for which input is provided. If a column is mapped for input but input is not provided, the data in the column is cleared. Use this operation when you want to update mapped columns and clear old data from remaining mapped columns. Unmapped columns are not modified.
Delete	Deletes one or more rows in a table.
Open	Opens a table or a business view.
Close	Closes a table or a business view.
Select	Selects one or more rows for a subsequent FetchNext operation. Click the Iterate Records button to automatically add a While loop to iterate over the resulting record set.

SelectAll	Selects all rows for a subsequent FetchNext operation. Click the Iterate Records button to automatically add a While loop to iterate over the resulting record set.
FetchNext	Fetches the rows that you specify. You can fetch multiple records with multiple FetchNext operations or with a single FetchNext operation in a loop.

Valid Mapping Operators

The Column Mapping form displays available objects that you can map to selected table columns. For SELECT statements, the available objects are used to build a WHERE clause. For FETCH statements, the available objects are used to receive data fetched from the database.

Key columns have a **Key** icon next to them.

This table describes the operators that you can use for mapping specific table I/O operations:

Table IO Operation	Mapping Operators
FetchSingle	Index Fields: =, <, <=, >, >=, !=, Like Non-Index Fields: Copy Target
Insert	All Fields: Copy Source
Update and Clear	Index Fields: = Non-Index Fields: Copy Source
Update and Retain (Release 9.2.7)	Index Fields: = Non-Index Fields: Copy Source
Delete	All Fields: =
Open	NA
Close	NA
Select	All Fields: =, <, <=, >, >=, !=, Like
SelectAll	NA.

Adding an Aggregation Statement (Release 9.2.7)

Use the aggregation statement in your logic extensions to aggregate data that is read from EnterpriseOne tables. In addition to reading data from EnterpriseOne tables, you can perform functions such as sums, averages, counts, and minimum or maximum values. These aggregated results can then be used in subsequent operations within the logic extension or passed as output to the next step in an orchestration.

To add an aggregation statement:

1. On the design panel, hover over the location where you want to add an aggregation statement and then click the **+** icon.

2. Select **Aggregation** from the Action menu. The system displays an Aggregation statement along with the branch label in the design panel and an Aggregation panel on the right.

The screenshot shows the Orchestrator Studio interface. On the left is a vertical menu with various action icons and labels: Assignment, Array Function, If Else, While, Table I/O, Aggregation, Business Function, Report, Workflow, Error, and Return. The 'Aggregation' option is highlighted. To the right of the menu is a form with two input fields: 'Name' and 'Description'. Further right, there are labels for 'Product C' and 'Category' with a pencil icon. The main design panel on the right displays a flow diagram starting with a 'Start' node, leading to an 'Aggregation' node (represented by a purple box with a sigma symbol and a red warning icon), which then leads to an 'End' node. A 'For Each Row' loop is connected to the 'Aggregation' node.

3. In the Aggregation panel, change the label and description.
4. In the Branch Label field the **For Each Row** value is displayed by default. This is an editable field. If you change the value in this field, the value in the branch label in the design panel will also change automatically.
5. Select the Object Type as:
 - o **Table:** If you select this option, you can enter the table name in the Object Name field or search for and select the table name.
 - o **View:** If you select this option, you can enter the business view in the Object Name field or search for and select the business view. The system also enables you to search for the business view using the application name. When you select View, the **Get View from Form** icon is enabled. Click this icon and enter the application name in the Application field to search and select the required form and business view.

6. Click the **Launch Column Mapping Builder** icon next to Column Mapping.

The system displays the Assignment Builder window containing these two panels:

The screenshot shows the 'Column Mapping - Aggregation' window. On the left is a grid of columns with various icons for filtering and aggregation. On the right are several configuration panels:

- Filter Criteria:** Includes a toggle for 'Include Empty Query Values' (checked), a 'Match Type' section with 'Match All' selected, and a table with one row:

Description	Operator	Mapped Object
AddressNumber	≤	up to an8 (e.g. 1000)
- Aggregations:** Includes a toggle for 'Include Count' (checked) and a table with two rows:

Description	Mapped Object
NameAlpha (Minimum)	AggMinNameAlpha
NameAlpha (Count Distinct)	count distinct
- Having:** Includes a table with one row:

Description	Operator	Mapped Object
NameAlpha (Count Distinct)	is greater than	"3"
- Group By:** Includes a table with one row:

Description	Format	Mapped Object
AddressType1		GrpAddressType1
- Order By:** Includes a table with one row:

Description	Order
AddressType1	Ascending

- o Column Mapping - Aggregation Panel (left panel): Displays a grid with the Filter field where you can filter the columns. The following icons are displayed for each column in the grid: Filter, Aggregation, Having,

Group By, and Order By. You can also see the corresponding sections next to the grid. These sections are empty by default. The system populates these sections based on your selection for a column in the grid.

You can click the drop-down menu in the Description field in all the sections (except in the Order By section) and select Add to Data Structure, Add to Variables, or Add to Array option. Click the **X** icon in the rows to remove a clause.

- Filter Icon:

The Filter icon is enabled by default for all the columns. You can click the Filter icon to add a column to the left operand (Description field) in the Filter Criteria section. You can filter and add the same column multiple times and define multiple criteria.

Filter Criteria Section

The Filter Criteria section contains a table with Description (read-only), Operator, and Mapped Object fields.

- a. The Include Empty Query Values option is enabled by default. Disable this option if required.
- b. Select the Match All (selected by default) or Match Any option.
- c. Select Operator from the drop-down menu. This is a required field.

Note: This table lists the data types and its supported operators in the Filter Criteria section.

Data Type	Supported Operators/Math Comparisons
String	<p>is equal to, is not equal to, starts with, ends width, contains, is between, is in list, is less than, is less than or equal to, is greater than, and is greater than or equal to</p> <p>If the Include Empty Query Values option is enabled, the system supports these two operators along with the above operators: is blank and is not blank.</p>
UTime	=, !=, <, <=, >, >=, is in list (between operator is not supported for UTime)
All other data types	=, !=, <, <=, >, >=, is between, is in list

For **in between** operator, the system the displays the Minimum Value and Maximum Value fields for mapping. For **is in list** operator, the system enables you to add multiple mappings. For **is blank** and **is not blank** operators, the system will not display any field for mapping.

- d. Click the Edit icon in the Mapped Object field. The system displays the Object Selector panel on the right. Use this panel to map the values.

- Aggregation Icon

The Aggregation icon is enabled by default for all the columns. This icon is disabled the Group By icon is selected.

The system displays the following options when you click the Aggregation icon:

- o Non-number type column: Sum, Average, Sum Distinct, Average Distinct, Minimum, Maximum, and Count Distinct.
- o Number type column: Minimum, Maximum, and Count Distinct.

Note: You can select the aggregation options such as Sum, Average, Sum Distinct, and so on only once for a particular column. For example, if you have already selected Sum for a column, the system will not display the Sum option again when you click the Aggregation icon for that column. You can select from the remaining options.

- Aggregations Section

The options selected for the columns using the Aggregation icon are displayed in this section. For example, if you select the Minimum option for the LineNumber column, the system displays `LineNumber (Minimum)` in the Description field in the Aggregations section. This section contains Description and Mapped Object fields.

- a. The Include Count option is disabled by default. You can choose to enable this option and then click Edit to create mapping it to a number type data structure or a variable using the object selector panel (right panel). The system displays a validation error if you have enabled this option without mapping. You can click the drop-down menu next to the Include Count field and select Add to Data Structure, Add to Variables, or Add to Array option.
- b. Click the Edit icon in the Mapped Object field for each row and map the objects by selecting the values from the right panel.

- Having Icon

The Having icon is disabled by default for all the columns. It is enabled for a column that has an aggregation value. The context menu options displayed when you click the Having icon depends on the values you have selected for your aggregation clause. For example, if you have selected Minimum and Maximum as your aggregation clauses, the system displays the Minimum and Maximum options when you click the Having icon.

Having Section

This section is similar to the Filter Criteria section, and it contains Description (read-only), Operator, and Mapped Object fields. You can click the Edit icon in the Mapped Object field and map the values using the right panel.

- Group By Icon

The Group By icon is enabled by default for all the columns, and it is disabled if a column is already included in the Aggregation section or Group By section.

Group By Section

The Group By section contains a table with Description (read-only), Format (required field for date type column), and Mapped Object fields. The available formats for date type format are: yyyy-MM-

dd, Year, Year-month, and Calendar Quarter. Use the right panel and select the mappings for the Mapped Object field.

- **Order By Icon**

The Order By icon is disabled by default for all the columns. This icon is enabled if:

- an aggregation combination is included in the Aggregation section but not in the Order By section.
- or a single column is already included in the Group By section and not in the Order By section.

Order By Section

The Order By section contains the Description (read-only), and Order (required) fields. From the Order drop-down menu, select the Ascending or Descending option.

- Object Selector Panel (right panel): Filter and select the mappings using this panel. You can click the Close or Open icons to hide or view this panel.

7. Click **Close**. All the mappings are displayed in the Aggregation panel.
8. Click **Save**.

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

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

Note: [Click here to view an OBE of this feature.](#)

Adding a Business Function (Release 9.2.6.3)

You can use the Business Function option in the design tab to create logic extensions that call EnterpriseOne business functions. This option enables you to call a business function from the expansive library of business functions that are included with the EnterpriseOne product, as well as custom business functions.

When you insert a business function to your logic extension, a Business Function panel displayed on the right. When you click the Edit icon, the system displays the Business Function Parameter Mapping window, where you can use the filter field to filter the parameters by name, and then map the parameters.

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

To add a business function to your logic extension:

1. On the design page, hover over the location where you want to add the business function and then click the **+** icon. Select **Business Function** from the Action menu. The system displays the Business Function panel on the right.
2. In the Business Function panel, edit the label and enter the description for the business function.
3. Enter a value in the Function Object Name field. You can also search and select the required object.
4. Select a value from the Function Name drop-down menu.
5. Click the **Edit** icon. The Business Function Parameter Mapping window is displayed.

The Business Function Parameter window contains two panels:

- Business Function Mapping Panel (left panel): Displays a table with the ID, Name, Description, Input, Output, Required, IO, and Mapping fields.

- o Object Selector Panel (right panel): Filter and select the value for the Mapped Object field using this panel. You can click the Close and Open icons to hide or view this panel.
6. In the Business Function Mapping window, use the Filter field to filter the objects.
 7. Click the **Close** button.

Your mappings are displayed in the Business Function panel.

8. Click **Save**.

You can add the business function parameters to the Data Structure and to the Variables tab from the Business Function Parameter Mapping window. Click the drop-down icon next to the Name field and select the **Add to Data Structure** or **Add to Variables** or **Add to Array (Release 9.2.7)** options as required.

Business Function Parameter Mapping - B43C0010 - Process F43C05 Data - D43C0010A

ID	Name	Description	Input	Output	Required	I/O	Mapping
5	cActionCode	Action Code (ACTN)	<input checked="" type="checkbox"/>	<input type="checkbox"/>		←	cActionCode
10	szAction_Date_Code	Action Date Code (ACTDTCD)	<input type="checkbox"/>	<input type="checkbox"/>			
11	szAction_Date_Description	Action Date Description (ACTDTDSC)	<input type="checkbox"/>	<input type="checkbox"/>			
17	szActionCodeComment	Action Date Comment (DTCOMM)	<input type="checkbox"/>	<input type="checkbox"/>			
23	szTransactionOriginator	Transaction Originator (TORG)	<input type="checkbox"/>	<input type="checkbox"/>			

Adding a Logic Extension (Release 9.2.7.3)

Starting with Tools Release 9.2.7.3, you can create logic extensions that call other logic extensions.

You can create a logic extension as a reusable component and add it in other logic extensions as a child component. This enables reusability of a logic extension.

To add a logic extension:

1. On the design page, hover over the location where you want to add the logic extension and then click the **+** icon. Select **Logic Extension** from the Action menu.
2. In the Logic Extension panel, edit the label and enter a description.
3. In the Logic Extension Name field, click the **Search** icon and then search for and select the required logic extension. A list of parameters is displayed in the Parameter Mapping section. The system displays the red "i" icon next to the required parameters.

Note: The name of the selected logic extension is read-only. The system displays the name of the selected logic extension in the Description field if the field is left empty. You can override the description.

4. Click the "?" icon or click the Edit icon. The system displays the Logic Extension Parameter Mapping window.

Note: The system displays the red "i" icon next to the required parameters.

The Logic Extension Parameter Mapping window contains two panels:

- o Logic Extension Parameter Panel (left panel): Displays a table with the ID, Name, Required, IO (not for the array type parameter), and Mapping fields.
You can click the drop-down list icon next to the Name field and select the **Add to Data Structure** or **Add to Variables** or **Add to Array** option as required. You can select the **Clear Mapping** (Eraser) option from the drop-down list to delete the mapping in the row. You can also click the Eraser icon at the end of the row to delete the mappings.
- o Object Selector Panel (right panel): Filter and select the value for the Mapping field using this panel. You can click the Close and Open icons to hide or view this panel.

5. In the Logic Extension Parameter Mapping window, map the parameters. For the Input Only parameters, you can map data structures, variables, literals, or system literals. For the Output Only parameters, you can map data structures or variables.

For an array object row, the Name field is displayed in grey. Data Dictionary and IO type fields are empty. The Mapping field displays the mapping depending on the children's array column mapping.

For the array column row, an array icon is displayed to indicate this is an array column.

Note: All the columns of an array parameter can only be mapped to the columns of another array. You cannot map an array parameter to the columns of multiple arrays simultaneously.

6. Click the **Close** button.
Your mappings are displayed in the Logic Extension panel.
7. Click **Save**.

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

Note: [Click here to view an OBE of this feature.](#)

Adding a Report (Release 9.2.6.3)

You can add a report interconnect to your logic extension. When you add a report statement to your logic extension, the system displays the Report panel on the right.

You can click the Edit icon in the Report panel to access the Report Interconnects Parameter Mapping window, where you can map the parameters.

To add a report interconnect:

1. On the design page, hover over the location where you want to add the report and then click the + icon. Select **Report** from the Action menu. The system displays the Report panel on the right.
2. In the Report panel, edit the label and enter the description for the report.
3. Enter a value in the Report Name field. You can also search for and select the required report.
4. Select a value from the Report Version drop-down menu. The report version can be dynamic and you can also select a data structure or a variable.
5. Click the **Edit** icon. The system displays the Reports Interconnects Parameter Mapping window containing two panels:

- Report Parameter Mapping Panel (left panel): This panel displays a table with the Name, Description, Input and Mapping fields. Optionally, you can use the Outputs window, and map the Job Number and Execution Server using the right panel.
- Object Selector Panel (right panel): Filter and select the mapping values using this panel. You can click the Close or Open icons to hide or view this panel.

6. Click the **Close** button. Your mappings are displayed in the Report panel.

7. Click **Save**.

Click the drop-down list icon next to the Name field and select the **Add to Data Structure** or **Add to Variables** or **Add to Array (Release 9.2.7)** option as required.

You can use the right panel to map the parameters.

Report Interconnects Parameter Mapping - R54HS400 - Detailed Incident Report

ID	Name	Description	Input	Mapping
2	mnIncidentNumber	Incident Number (HSINO)	<input checked="" type="checkbox"/>	
3	szProgramId	Program ID (PID)	<input type="checkbox"/>	

Adding a Workflow (Release 9.2.6.4)

Starting with Tools Release 9.2.6.4, you can add workflows to your logic extensions. When you add a workflow to your logic extension, the Workflow panel is displayed at the right hand side of the design page that enables you to search for and add a workflow in the Workflow field.

After you add the workflow in the Workflow field, the Key Data Structure and the Additional Data Structure sections are populated. You can click the Edit icon next to Key Data Structure and Additional Data Structure and map the parameters. Next to each required parameter the system displays the Incomplete Mapping (!) icon in red. You can click the Edit icon to open the Workflow Parameter Mapping window and complete the mappings for a required parameter. The Input fields are disabled for the required parameters.

For more information on workflows, see *Designing a JD Edwards Workflow Process in Orchestrator Studio Workflows (Release 9.2.6)*.

To add a Workflow:

1. On the design page, hover over the location where you want to add the workflow and then click the **+** icon. Select **Workflow** from the Action menu. The system displays the Workflow panel on the right.
2. In the Workflow panel, edit the label and enter the description for the workflow.

3. In the Workflow field, click the Search icon to search for and select the required workflow from the Workflow List window. After you select the workflow, the Key Data Structure and Additional Data Structure sections are displayed to create the mappings. Next to each required parameter the system displays the Incomplete Mapping (!) icon in red.
4. Click the **Edit** icon next to Parameter Mapping and Additional Data Structure to open the Workflow Parameter Mapping window and complete the mappings as required.
5. Click the **Close** button.
6. Click **Save**.

You can add the workflow parameters to the Data Structure tab and to the Variables tab from the Workflow Parameter Mapping window. Click the drop-down list next to the Name field and select the **Add to Data Structure** or **Add to Variables** or **Add to Array (Release 9.2.7)** option as required.

Adding a Form Control Function (Release 9.2.7.3)

Starting with Tools Release 9.2.7.3, you can design a logic extension with the ability to change the simple user interface on the form, for example, to show, hide, enable, disable, or change the colour of a control. You can determine the value a user has entered into a field, and based on that value conditionally enable, disable, or highlight a different control.

To know more about how to associate logic extensions with events in Form Extension, see *Associating Orchestration with Events* in the *JD Edwards EnterpriseOne Tools Using and Approving User Defined Objects Guide*.

To add a Form Control Function:

1. On the design page, hover over the location where you want to add the form control function and then click the **+** icon. Select **Form Control Function** from the Action menu. The system displays the Form Control Function panel on the right.
2. In the Form Control Function panel, edit the label. The system displays the function name as the description of the form control. You can override the description.
3. In the Function Name drop-down list, select the function name. The system supports the following functions:

Note: You can hover over the Information icon next to the Name drop-down list to see the following information about the controls.

- o **Enable Control:** Only applies to push buttons, check boxes, combo boxes, radio buttons, edit fields, and hypercontrols such as menu controls and exits.
- o **Disable Control:** Only applies to push buttons, check boxes, combo boxes, radio buttons, edit fields, and hypercontrols such as menu controls and exits.
- o **Show Control:** Only applies to push buttons, check boxes, combo boxes, radio buttons, edit fields, static text, and images.
- o **Hide Control:** Only applies to push buttons, check boxes, combo boxes, radio buttons, edit fields, static text, and images.
- o **Set Control Text:** Only applies to push buttons, check boxes, combo boxes, and static text.

The system enables the **Text** field when you select this option. Either enter a value in the Text field or select the value from the drop-down list.

The string type data structures and variables (excluding array object, array length, and control object) are displayed when you click the Text drop-down list.

Note: Text field is an optional field.

- o **Set Edit Control Color:** Only applies to edit fields.
The system enables the Color drop-down list when you select this option. Select the color from the drop-down list.

- From the Control drop-down list, select the control defined in the Data Structure tab.

Note: The Form Control function will execute only when this logic extension is called directly from a from extension. If this logic extension is included in an orchestration, the Form Control function will have no effect.

- Click **Save**.

Note: The inputs and mappings for the Form Control Function are defined in Form Extension of the EnterpriseOne form. Therefore, the form controls are not displayed in the Test tab of the Logic Extension page. Similarly, the diagnostic mode does not display any information about the form controls. In the diagnostic mode, you can only see the execution path of the form control function.

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

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

Note: [Click here to view an OBE of this feature.](#)

Adding an Error Statement

You can set an error statement at the required point in the logic extension. When you add an error statement in the logic extension, the Error panel is displayed on the right. For the Error field, you can type text that will display in the error when the Error step is encountered during execution. Alternately, you can click the **Search** icon to find a Data Dictionary Message for the error. An error must be associated with one of the Data Structure elements of the Logic Extension. The associated data structure element ID is displayed along with the error in the response. If a data structure element is not specified then the first data structure element is automatically selected.

To add an error statement:

- To add an error statement, on the design page, hover over the required location and click the **+** icon.
- Select **Error** from the Action menu. The system displays the Error panel on the right.
- In the Error panel, edit label and enter the description for the error statement.
- In the Error field, enter or search for an error message.
- In the Structure Item field, select a value using the drop-down menu.
- Click **Save**.

Adding a Return Statement

Use the return statement to stop execution. When a return statement is encountered during execution, the logic extension stops processing and returns the output data in the current state. This can be used to exit the logic extension when a certain condition is met or as a temporary statement while debugging to see the state of the output data prior to a step.

To add a Return statement:

- On the design panel, hover over the location where you want to add the return statement and then click the **+** icon.
- Select **Return** from the Action menu.
- In the Return side panel, edit the label and enter the description for the return statement.
- Click **Save**.

Testing the Logic Extension

(Release 9.2.6.3) If you want to skip a particular logic while testing your logic extension, you can right-click the logic, and then select the **Disable** option from the context menu. The logic statement that is disabled is grayed out and a disable icon is displayed at the left corner. The system skips the disabled logic statement while running the logic extension. When a parent statement is marked as disabled, all the children statements will be disabled (grayed out). You can right-click the disabled logic statement and select **Enable** to enable the statement again.

See *Understanding the Test tab*.

To test a logic extension:

1. Click the **Test** tab.
2. Enter the values in the Input field.
3. Click the **Test** button to test your logic extension. Verify the response in the Response panel at the right hand side.

(Release 9.2.6.3) You can click the **Test in Diagnostic Mode** button to troubleshoot your logic extension.

Starting with Tools Release 9.2.7, the Abort function is supported for the following use cases:

The system displays a pop-up error message with information about the aborted statement when you click the Test button. You can see that the highlighted path ends at the aborted statement in the diagnostic mode. . You can inspect more using the diagnostic mode.

- The get row array function is called with the \geq index value even if the array is initialized or any index for an uninitialized array. (for example, "array get by index out of bound abort")
- The update row array function is called with the \geq index value or the array is uninitialized. (for example, "The update row array function is called with the \geq index value")
- Database issues or if the column security prevents the access of BSSV columns.
- Invalid input for aggregation.

Working with Logic Extension Assertions (Release 9.2.7.4)

You can validate a logic extension by saving a set of inputs and a set of outputs using the built-in framework provided by the JD Edwards Orchestrator Studio. You can save multiple test cases and run them individually or as a set. This feature enables you to test and revalidate the logic whenever something changes, for example, after applying a software update.

To create an assertion (test) case:

1. Create your logic extension. See *Creating a Logic Extension*.
2. Click the **Test** tab.

The system displays the Assertion Case Selection drop-down list with the default value `<clear>` .

Note: The grid displays the Name, Input, and Output columns.

3. Enter the input values in the Input field.
4. Click the **Run** button.

The system displays the output value in the Output field, details in the Response and Expected Error sections, and enables the Save As button.

5. To save the input and output values as a test case, click the **Save As** button.

In the Save As pop-up window, retain the auto-generated name or change the name in the Name field, and then click **OK**.

The system displays the newly created test case name in the Assertion Case Selection drop-down list.

Note: The grid in the Test tab now displays the Name, Input, Expected Output, and Actual Output columns. You can also see the read-only Expected Error section below the grid. Click the **Menu** icon and select **Delete** or **Rename** options if required for a test case.

The screenshot shows the Orchestrator Studio interface for configuring a logic extension named "Generate Discount Message". The breadcrumb trail is "Home > Logic Extensions > Generate Discount Message". The main area shows the logic extension details, including its name, description, and various input/output fields. A dropdown menu is open, showing options: "Delete", "Rename", and "Run all tests in this Object". The dropdown menu is highlighted with a red box.

Name	Input	Expected Output	Actual Output
Item Price	100		
Quantity	10		
Discount Percentage	0.2		
Message		With discount you save:200.0	With discount you

Expected Error
[]

- To compare the test case output with the actual output, click the **Run** button again, and verify the values in the Expected Output and the Actual Output fields. You can also change the values in the Input field to compare the outputs.

If there is a single value or an array value difference between the actual output and the expected output, the system displays the value in the Actual Output column in red. Additionally, the system displays the status of the result as Passed or Failed. You can review the details in the Response and the Expected Error sections.

If there is any difference between the actual error and the expected error, the system displays a red border around the Expected Error section.

Similarly, you can create and test multiple test cases. To create a new test case with different input and output, reset the value in the Test Case Selection drop-down list as <clear>, enter input values in the grid, click **Test**, click **Save As**, retain the auto-generated name, or enter a new name, and then click **OK**.

You can click the Assertion Case Selection drop-down list to view all the test cases. To modify an existing test case, select the test case from the Test Case Selection drop-down list, enter the new input values in the grid, click **Test**, and then click **Save**.

Home > Logic Extensions > Generate Discount Message

Name: **Generate Discount Message** Product Code: 55 - Reserved for Clients

Description: Generate a message for how much is saved with a discount. Inputs are Item Price, Quantity, and Discount Percentage. This logic extension calls... Category: []

Test Case Selection: Case 1 + Save As Save Passed

Test Test in Diagnostic Mode

Name	Input	Expected Output	Actual Output
Item Price	100		
Quantity	10		
Discount Percentage	0.2		
Message		With discount you save:200.0	With discount y

Expected Error: []

```

Response
{
  "submitted": true,
  "output": {
    "4": "With discount you save:200.0"
  },
  "error": null,
  "errorList": [],
  "diagnostics": null
}
    
```

Running Assertion Cases in a Batch

To run assertion cases in a batch:

1. After you create all the required the assertion cases, click the **Menu** icon.
2. Click **Run all assertions in this object**. The system displays the Test Report window and the title for this window displays the Logic Extension name. In the Test Report window, verify the progress and status of the assertion cases.
3. To download the assertion report, click **Create Report**. The system downloads a csv file with the assertion case details.

To save the assertion cases as User Defined Objects (UDO), click **Save** at the right corner of the Logic Extension window.

Home > Logic Extensions > Generate Discount Message

Name: **Generate Discount Message** Product Code: 55 - Reserved for Clients

Description: Generate a message for how much is saved with a discount. Inputs are Item Price, Quantity, and Discount Percentage. This logic extension calls... Category: []

Test Case Selection: Case 1 + Save As Save Passed

Test Test in Diagnostic Mode

Name	Input	Expected Output	Actual Output
Item Price	100		
Quantity	10		
Discount Percentage	0.2		
Message		With discount you save:200.0	With discount y...

Expected Error: []

```

Response
{
  "submitted": true,
  "output": {
    "4": "With discount you save:200.0"
  },
  "error": null,
  "errorList": [],
  "diagnostics": null
}
    
```

Note: [Click here to view an OBE of this feature.](#)



Testing the Logic Extension Using the Diagnostic Mode (Release 9.2.6.3)

Using the diagnostic mode, you can troubleshoot a logic extension by verifying the values of the variables, data structure items, and all system variables at each step.

To test your logic extension using the diagnostic mode, access the logic extension and in the Test tab, click the **Test in Diagnostic Mode** button. You can see that the executed path of your logic extension is highlighted in green. A number is displayed in a green field on the branch label of the While statement, Aggregation statement (Release 9.2.7) and every branch label of the If Else statement. The value displayed in each of these fields represents the number of times a branch is executed. You can hover over the branch label of If and While statements to view the detailed criteria without exiting the diagnostic mode.

When you hover over a step in the design panel, the corresponding steps are highlighted in the right panel. You can click a particular step in the design panel to filter and view only the corresponding steps in the right panel. To view all the steps again in the right panel, click the Show All button.

In the right panel, you can expand each step and verify the details (logs) at the execution level. The changed values are displayed in red and the unchanged values are displayed in black. Use the Previous and Next buttons to navigate across the execution path.

The system enables you to set the checkpoint for a step and then inspect the data at the checkpoint . If the value has changed during the execution of a statement, the information  icon is displayed. Hover over the information icon to view the changes.

When there is no filter (statement or string filter), you can use the Previous and Next buttons to move the checkpoint backward or forward along the execution path and the system highlights the statement executed in the checkpoint entry. The step is expanded enabling you to inspect the details.

Starting with Tools Release 9.2.7.4, the system displays the Statement Detail view in diagnostic mode for all types of logic extension statements except for the While and Return statements. For the While and Return statements, the system displays only the All Data view.

The Statement Detail view is displayed by default and you can toggle between the All Data and the Statement Detail views. In the All Data view, information about the data structure, variables, and system variables is listed. In the Statement Detail view, only the data related to the context of the particular statement is displayed.

For example, in the Statement Detail view of an Assignment statement, the system displays the name of the assignment, input and output parameter mappings, and related values. The related values are displayed in both the before and after states if the value has changed.

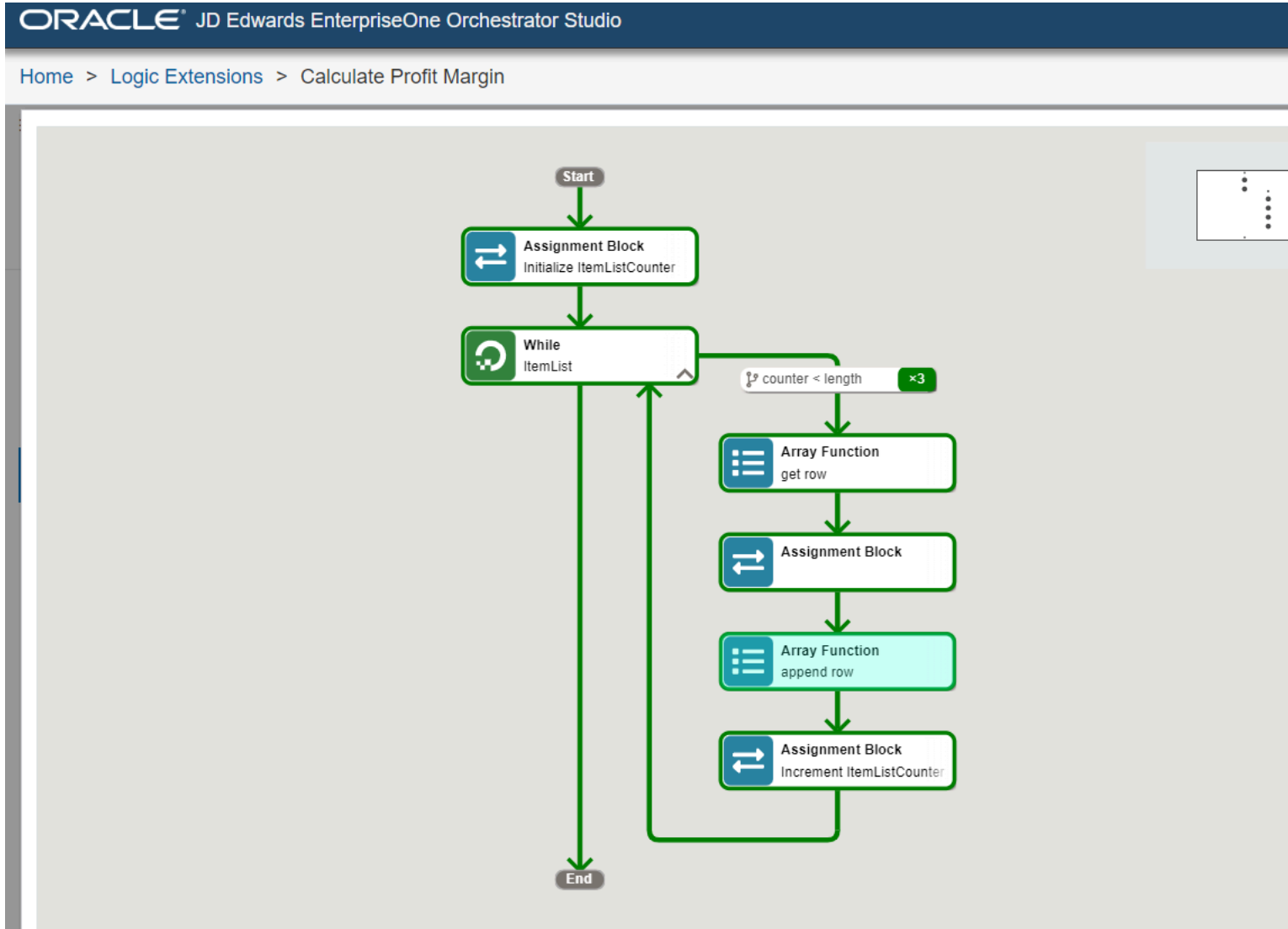
The screenshot displays the Oracle JD Edwards EnterpriseOne Orchestrator Studio interface. The main workspace shows a workflow diagram with the following steps: Start, Assignment Block, Array Function (get row), Assignment Block, and End. The right-hand panel is titled 'Assignment Block' and shows the 'Statement Detail' view. It lists various variables and their values, with some values highlighted in red to indicate changes. The variables listed include num4, num1, v num2, string2, string6, num1, string2, and string2.

(Release 9.2.7) When you hover over an Aggregation statement in the design panel, the system highlights all the iterations of aggregation in the right panel. You can click the Aggregation statement in the design panel to filter and display only the corresponding aggregation iterations in the right panel. Expand the aggregation statements in the right panel to view the execution details.

(Release 9.2.7) When you hover over an Array function in the design panel, the system highlights all the iterations of the array in the right panel. You can click the Array function in the design panel to filter and display only the corresponding

array iterations in the right panel. Expand the Array functions in the right panel to view the execution details. The system displays the changes in red color.

For example, in the following screenshot, you can see details of the append row array function. The number 2 is highlighted in red to indicate that the second row is newly appended. When you click the Array (Show Detail) icon, a pop-up window with the array table is displayed. The newly appended array is displayed in red color.



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

5 Creating Orchestrations with Orchestrator Studio 9.2.4

Creating Orchestrations

This section contains the following topics:

- *Understanding Orchestrations*
- *Creating an Orchestration*
- *Adding Steps to an Orchestration*
- *Adding Inputs to an Orchestration*
- *Setting Up Job Queue and Scope for an Orchestration (Release 9.2.4.4)*
- *Mapping Orchestration Inputs*
- *Retrieving and Passing Data Sets in an Orchestration*
- *Working with Orchestration Output*

Understanding Orchestrations

Creating an orchestration in the Orchestrator Studio involves:

- Naming the orchestration and specifying the input format for the orchestration.
- Adding inputs to the orchestration.

The inputs define the data that is passed to the orchestration from a device, third-party application, custom program, or Cloud service. For example, an orchestration that is designed to receive input from a sensor, will receive an input that may include an ID that identifies the sensor, and data 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 step can be a service request, rule, cross reference, or white list. At a minimum, an orchestration requires 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 that are defined in each orchestration step, such as inputs in a rule, cross reference, white list, service request component, and system variables.

Note: Remember that when you are ready to "request to publish" an orchestration, you need to make sure that you also request to publish all the components that are associated with the orchestration. The request to publish the components ensures that an administrator saves all the 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 an error.

Creating an Orchestration

To create an orchestration:

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

All orchestrations that are created in Orchestrator Studio 9.2.4 or later are saved as version 3 orchestrations. You cannot change this value. If you want to update a version 3 orchestration that was created in a previous version of the Orchestrator Studio, see [Updating to Version 3 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.

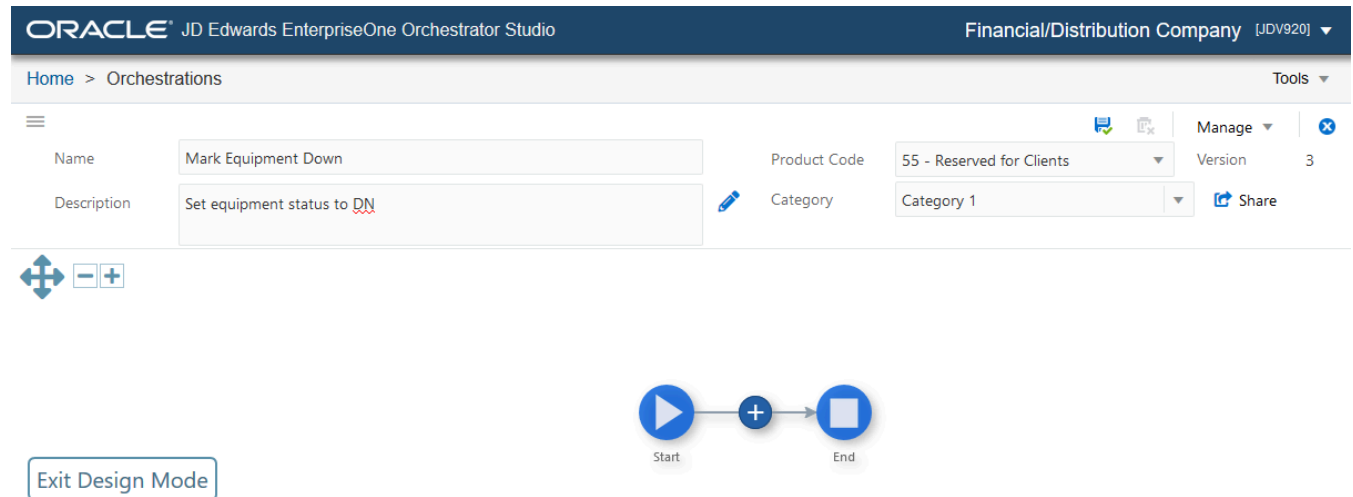
This enables an administrator to manage UDO security for orchestration components by product code.

5. In the **Description** field, enter a short description with a maximum of 200 characters. This description appears along with the orchestration name in the component list.

- Click the **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.

At this point, the design mode is enabled for the orchestration and the canvas displays the Start and End nodes of the orchestration as shown in the following figure:



- Enter a Category for the orchestrations or the orchestration components. The Category drop-down list displays the orchestrator categories if any exist. You can choose from the available categories or enter a new value in the Category field.
- Click **Save** before defining inputs or adding 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 steps of the orchestration. That is, both the original orchestration and the new orchestration use the same components that are included in 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.

Adding Steps to an Orchestration

Each component (data request, form request, rule, cross reference, white list, and so on) that you add to an orchestration is an orchestration step.

Adding the Initial Step to an Orchestration

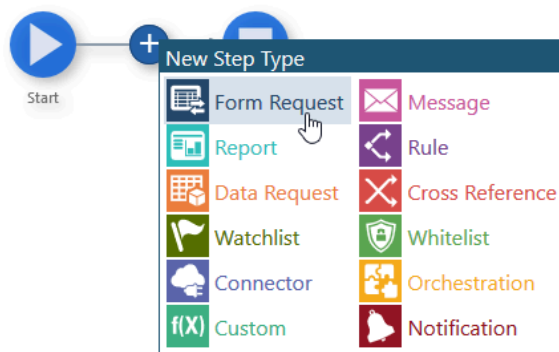
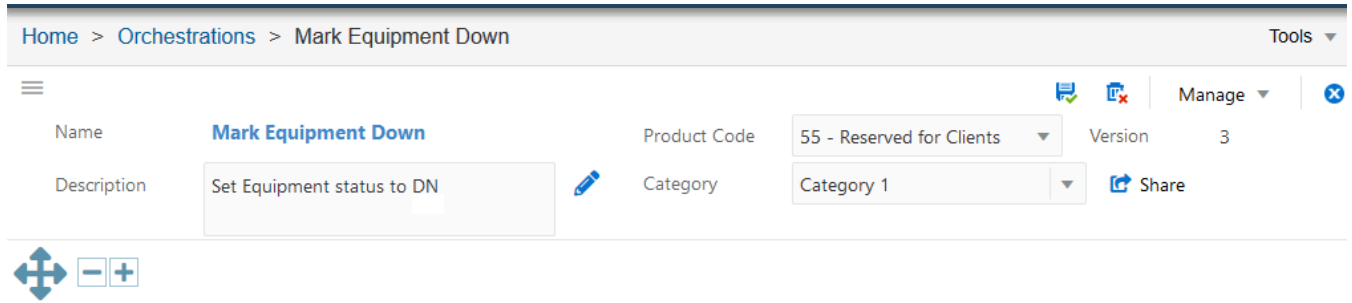
1. In an Orchestration, click the **Design Mode** button at the bottom-left corner of the canvas area.

A plus sign (+) appears in the orchestration path between the Start and End nodes.

2. Click the **Add Step** button (+).

Alternatively, double-click the path in between the Start and End nodes in the orchestration.

3. In the **New Step Type** drop-down list, select one of the component types to add to the orchestration. The following figure shows the New Step Type drop-down list.

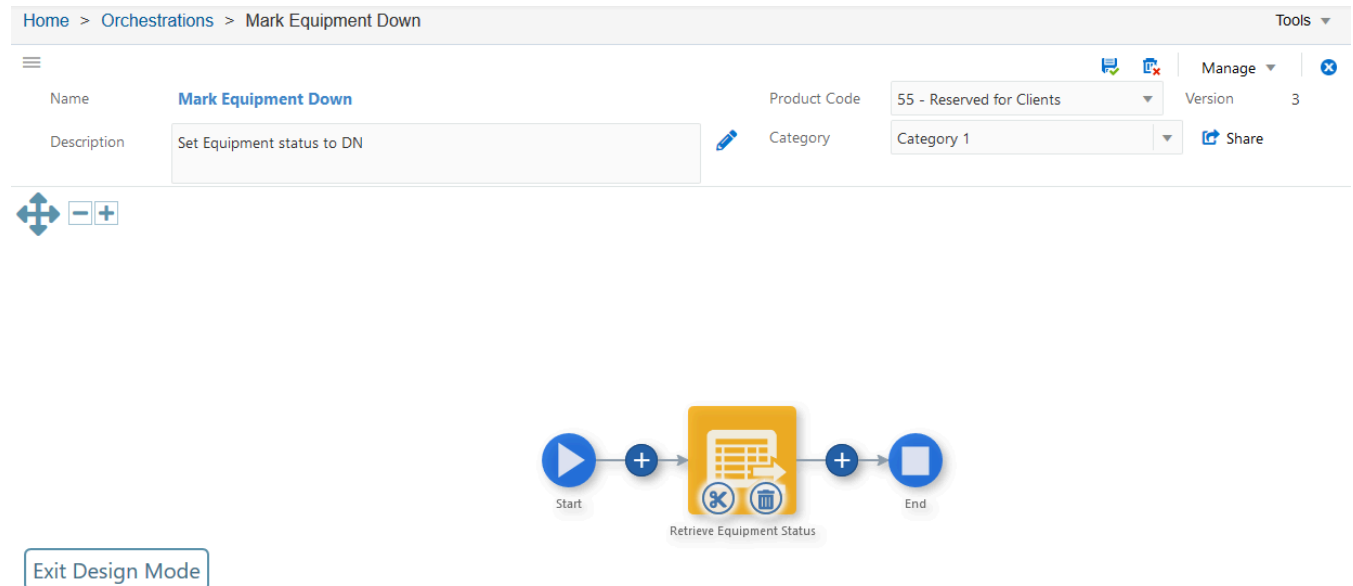


Exit Design Mode

4. The Orchestrator Studio displays a list of objects for the corresponding component type.
5. Define the component for the step:
 - a. To use an existing component for the new step, search and select a component from component list.
 - b. To create a new component for the step, click the New button to access the design page for creating the new component.

After creating the component, when you return to the orchestration, the component is added a new step to the orchestration.

The Orchestrator Studio adds the component as a step to the orchestration as shown in the following figure:



Adding Additional Steps to an Orchestration

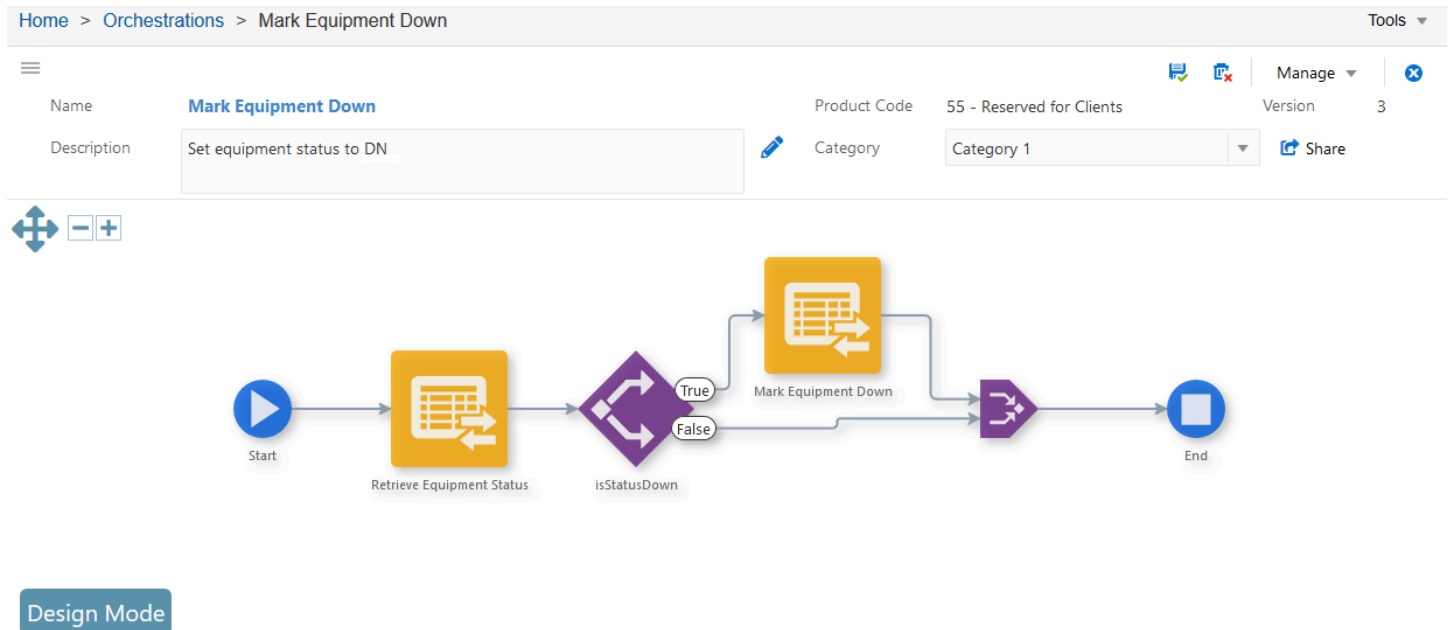
1. In the **Design Mode** of the orchestration, select the location in the orchestration path where you want to add step.
2. Click the **Add Step** button.

Alternatively, double-click the path at the location where you want to insert a step in the orchestration and select the component type from the New Step Type drop-down list. For example, you can add a white list to an existing orchestration, as a step before the other steps or in any other location in the orchestration.

3. In the **New Step Type** drop-down list, select the component type that you want to add.
4. In the component list that is displayed, search and select the component that you want to add.

The Orchestrator Studio adds the new component as an additional step to an orchestration.

The following figure shows an orchestration that contains a rule and form service requests steps:



Each box in the orchestration canvas represents a step in the orchestration. For information about the other features of the canvas, see *Working with the Graphical Representation of an Orchestration*.

- **Name.** Displays the name of the component.
- **Type.** Each component type is displayed in a unique icon: Rule, Form Request, Cross Reference, White List, and so on.
- **Action** (True or False). Define which subsequent component is invoked based on the criteria in the preceding rule. See *Defining the Actions Between a Rule and Dependent Components* for details.

As the next task, add and map the orchestration inputs to inputs in the orchestration steps. See, *Adding Inputs to an Orchestration* and *Mapping Orchestration Inputs*.

Adding a Local Orchestration or Notification as a Step to an Orchestration (Release 9.2.4.3)

Starting with Tools release 9.2.4.3, you can add a local orchestration or notification directly as a step of an orchestration without creating a connector.

To add a local orchestration:

1. In the **Design Mode** of the orchestration, select the location in the orchestration path where you want to add the orchestration.
2. Click the **Add Step** button.

Alternatively, double-click the path at the location where you want to insert a step in the orchestration and select the component type from the New Step Type drop-down list.

3. From the **New Step Type** drop-down list, select the **Orchestration** component.
4. From the component list that is displayed, select the orchestration that is available in the current environment.

The Orchestrator Studio adds the selected orchestration as a step to an orchestration.

As the next task, add and map the orchestration inputs to the inputs in the local orchestration.

5. In the orchestration flow, click the local orchestration from which you want to add and map the inputs, and then click the **Transformations** icon.
6. In the Transformations dialog box, click the **Add Inputs to Orchestration** button to generate inputs from the local orchestration. Use the **Auto Map** button to automatically map any matching inputs in the orchestration and the local orchestration.

The following figure shows an example of the Transformations dialog box for a local orchestration that is added as a step to a parent orchestration:

The screenshot shows the 'Transformations' dialog box for the 'Add Employee' orchestration. The dialog includes a table with the following data:

Input	Available Values	Default Value
TaxID	csvArray.TaxId	
Name	csvArray.Name	
Salary	csvArray.Salary	
ZipCode	csvArray.ZipCode	
Gender	csvArray.Gender	

The dialog also features a 'View Example Input' pop-up window showing the following JSON structure:

```
{
  "TaxID": 0,
  "Name": "string",
  "Salary": 0,
  "ZipCode": "string",
  "Gender": "string"
}
```

7. Enable the **Fire and Forget** toggle if you want the orchestration to execute without waiting for a response. If the Fire and Forget toggle is enabled, the called orchestration will run asynchronously.

(Release 9.2.4.4) The **Number of Threads** option is displayed when you enable the Fire and Forget toggle. This option enables you to select the number of threads.

For example, if an array of 1,000 records is sent as the iterator for a fire and forget orchestration and the value in the Number of Threads option is 10, only 10 orchestrations will be executed at one time. The remaining orchestrations will be in a waiting state until one of the 10 threads is available again. Similarly, the remaining set of orchestrations are processed until all of them are completed.

8. Enable the **Use Object for Input** toggle if you want to pass and map an object as an input to an orchestration.
9. Click the **View Example Input Object** button to view the sample input for the selected orchestration when you use an object as an input.

In the Orchestration Outputs dialog box, you can select the entire output from the local orchestration or notification step. You cannot individually select the return fields that you want to add to your orchestration output.

Individual outputs returned from a local orchestration can also be mapped as variables to subsequent steps. You cannot pass an array.

When you run an orchestration that contains a local orchestration, the Output section on the Run Orchestrations page displays the name of local orchestration. This is not the case when an orchestration is invoked using a connector. The Output section enables you to identify orchestrations that are used directly as steps and not as connectors.

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. In the **Design Mode** of the orchestration, select the component box that you want to remove.

The action control icons are displayed for the component.

2. Click the **Remove Step** icon to remove the component from the orchestration.

If you make a mistake when updating an orchestration, select the Restore option from the Manage drop-down menu to restore the orchestration to its last saved version. This action erases any changes made since the last save.

Moving a Component in the Orchestration

You can use the Cut Element and Paste Element options to move a component within an orchestration. You can move a component within an orchestration, but not across different orchestrations.

To move component in an orchestration:

1. Open an orchestration in the design mode.
2. Select the component in the orchestration that you want to move.

The action control icons for the component are displayed.

3. To move or cut a component, click the **Cut Element** icon.

When you use the Cut Element option, the selected component is copied to the clipboard. At this point, the Paste Element icons appear in the orchestration path.

4. Click the **Paste Element** icon at the location in the orchestration where you want to place the component.
5. Save the orchestration.

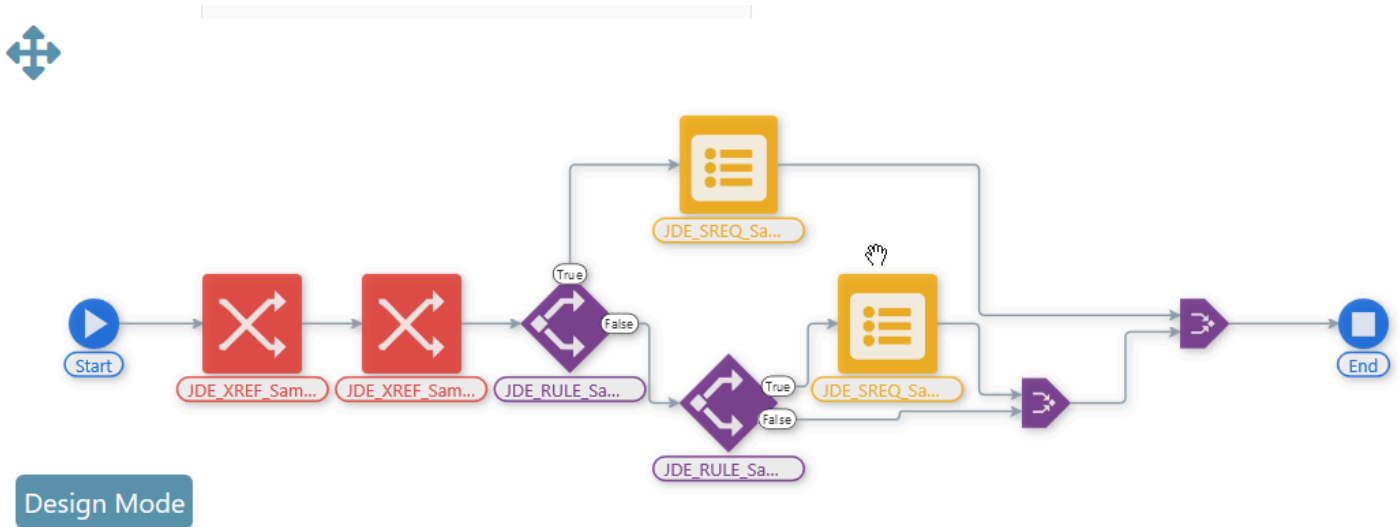
Defining the Actions Between a Rule and Dependent Components

The Orchestration design page enables you to define the actions between a rule component and other components in an orchestration. After you create a rule with conditions and add the rule 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 the orchestration needs to invoke a service request when the condition of a rule is met, then in the path for True, you have to add the service request.

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

The following figure shows an example of an orchestration with two rules and two service requests, with the following defined actions:

- When the condition in the first rule step is met (True), the action is set to instruct the orchestration to invoke a form request step.
- When the condition in the first rule step is NOT met (False), the action is set to instruct the orchestration to invoke a second rule.
- When the condition in the second rule is met (True), the action is set to instruct the orchestration to invoke a second form request.



To set the steps for True or False conditions of a Rule:

1. In the Orchestration design mode, add a Rule component to an orchestration.
2. In the orchestration path for **True**, click the **Add Step** button (+) to add a step for the true condition.
3. Similarly, in the orchestration path for **False**, click the **Add Step** button (+) to add a step for the false condition according to your requirement.

4. Click the **Save** button.

After completing the orchestration, you can use the Run Orchestrations page to test the orchestration in a test environment. You can access the Run Orchestrations page from the Tools drop-down menu in the upper-right corner of the Orchestrator Studio. See Chapter "*Testing Orchestrations*" for more information.

Defining Error Handling for Orchestration Steps

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

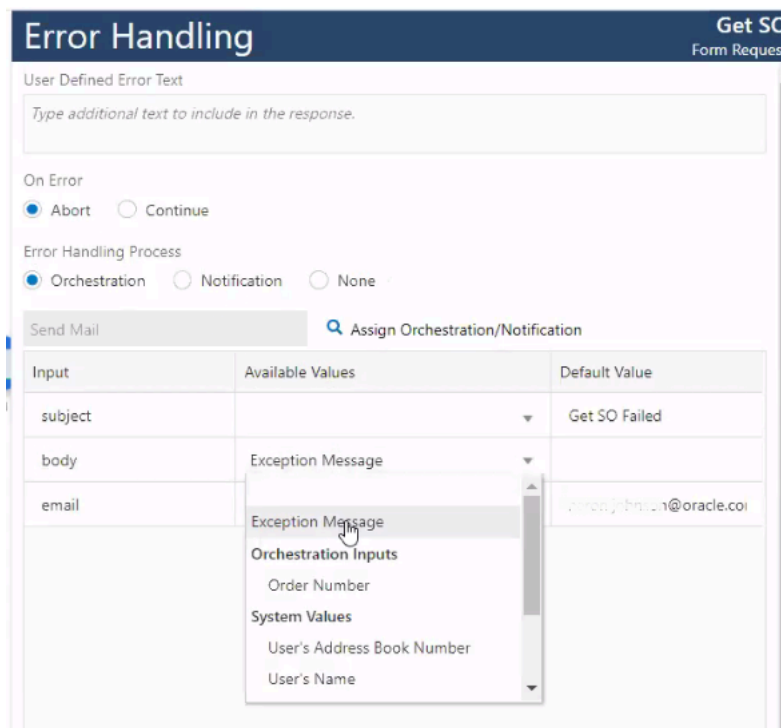
- Continues processing the orchestration after the error occurs, effectively ignoring the error.
- Cancels the orchestration.
- Cancels the orchestration and invokes an alternate orchestration or notification. For example, the orchestration might call a notification to alert subscribers if an orchestration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

You have the option to either include additional text in the error, or to 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 *Orchestrator Health and Exception Monitoring*.

To define error handling for an orchestration step:

1. In the orchestration design page, click the component step.
The action control icons are displayed for the component.



2. Click the **Error Handling** icon.

The Error Handling dialog box displays the type and name of the step, as shown in the following figure:

3. 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 braces.

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

4. For **On Error**, select one of the following options:
 - o Abort
 - o Continue
5. (Optional) If you choose to **Abort** on error, you can configure the orchestration step to invoke another orchestration or notification:
 - a. Click the **Assign Orchestration/Notification** button and select an orchestration or notification to invoke after an error.

Starting with Tools Release 9.2.6.3, when mapping values for orchestrations or notifications you can select **<Space>** from the Default Value drop-down menu.

- b. Map the inputs as appropriate.

(Release 9.2.4.4) You can pass the exception message to the called orchestration or notification. For example, you can choose to receive a notification email that includes the exception message from the failed step. You can pass the exception message to one of the orchestration or notification inputs by choosing **Exception Message** from the drop-down menu in the Available Values section.

(Release 9.2.8.2) For a REST or Open API connector step, you can pass the response status for the step to the called orchestration or notification. You can pass the response status to one of the orchestration or notification inputs by choosing **Response Status** from the drop-down menu in the Available Values section.

Adding the While Step (Release 9.2.8.3)

You can add a While step to repeat steps while a particular condition is true in your orchestration. The While step uses the existing rules in the Orchestrator Studio to add conditions.

To add a While step in your orchestration:

1. Access your orchestration and click **Design Mode**.
2. Identify the location in your orchestration diagram where you want to add the while step and click the **Add Step** button (+).
3. In the New Step window, select **While**. The system displays a window along with the list of existing rules.
4. Select the rule as required from the list. You can also click the **New** button and define a new rule.

See *Creating Rules*. This new rule is used in the while step when you save it. The system displays the name of the selected rule as the name of the while step.

5. Click the While step, and use these icons:
 - o **Choose Step**: Click this icon to change the rule that you added to your while step.
 - o **Edit**: Click this icon to make changes to the added rule.
 - o **Transformations**: Click this icon to map orchestration inputs to inputs. See *Mapping Orchestration Inputs*.

In the **Wait Time (Seconds)** field, specify a value. This is the wait time between any two iterations. You can enter a number between 0 and 60.

In the **Maximum Iterations** field, specify the limit for the number of times you want the while loop to iterate in order to prevent the while loop from iterating endlessly. If the while loop exceeds the iteration limit, the while step will end in error. The default value in the Maximum Iterations field is the value entered in the **Maximum While Loop Iterations** field for the AIS Server in Server Manager. You can only decrease this value.

To increase the maximum value, change the value in the **Maximum While Loop Iterations** field for the AIS Server in Server Manager.

- **Error Handling:** Click this icon to set up error handling. See *Defining Error Handling for Orchestration Steps*.

6. Click **Save**.

Adding Inputs to an Orchestration

Orchestration inputs identify the data an orchestration consumes from a calling custom program, a third-party application, a mobile application, an equipment meter, an IoT device, or a Cloud service. For example, you could have an orchestration that consumes data from an equipment manufacturer, with an "EquipmentID" input to pass the meter reading of the equipment to the orchestration.

For each input, you specify an input type, which can be a string, a numeric value, or various date formats. You can add four types of inputs to an orchestration:

- **Orchestration Inputs.**

You can manually enter orchestration inputs, including arrays and objects, 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 in the Transformation dialog box to automatically add inputs that are defined in a component as inputs to the orchestration. You can remove any of the inputs as needed, and then map the orchestration inputs to the appropriate inputs in the orchestration steps.

- **Variables.**

You can enter variables that are available as inputs to other steps or that can be used to return data. To change the variable value in a step, you must define the variable value 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 the subsequent steps in the orchestration.

- **System Values.**

New orchestrations include the following default inputs: *User's Address Book Number*, *User's Name*, *User ID*, *User ID Long*, *System Date*, *System Time*, *Orchestration Input*, and *Environment* (Release 9.2.6). 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.

- **Value from Steps.**

When you add a form request, data request, or cross reference that is configured to return values from EnterpriseOne, the variables that are defined to represent the return values appear as additional orchestration inputs. Therefore you can map the 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 input includes a variable that represents the JSON output of the service request, and gives you the option to map the JSON to a subsequent orchestration step.

Each component that you add to an orchestration—a service request, rule, white list, or cross reference—also includes one or more inputs which evaluate data from orchestration 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. On the Orchestrations design page, click the **Start** node.
The action control icons are displayed above the Start node.
2. Click the **Inputs and Values** icon.
3. In the Inputs and Values dialog-box, click the Input Format drop-down menu and select the appropriate format:
 - o JDE Standard
 - o Generic

See *Supported Input Message Formats* for more information about which input format to use.

4. Click the **View Example Input** button to view a sample of the input that is passed to the orchestration. The sample input is displayed in the JSON format. (Release 9.2.4.3)
5. In the **Orchestration Inputs** tab of the Inputs and Values dialog box, enter the name of the input in the Input column.

Starting with Tools Release 9.2.8.2, you can reorder orchestration inputs by dragging and dropping the rows. This feature improves readability and enables orchestrations to easily integrate with user interfaces or third-party systems that might be sensitive to the order of inputs.

6. In the **Value Type** column, select the input value type. Valid values are:
 - o Array

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

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. And then entered the inputs "Update Row" and "Quantity" for the array in the subsequent rows.

Inputs and Values					
Input Format: Generic		View Example Input		Job Queue: Default	Job Queue Scope: System
Orchestration Inputs		Variables	System Values	Values From Steps	
	Input	Value Type	Default Value	Required	
▲	GridData	Array		<input type="checkbox"/>	✕
	GridData	Update Row		<input type="checkbox"/>	✕
	GridData	Quantity		<input type="checkbox"/>	✕

- o String
- o Numeric
- o Boolean
- o Object

This input type allows you to pass a JSON object into an orchestration. You can use this input type to pass nested arrays of data by defining arrays of objects.

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

- o `dd/MM/yyyy`
- o `dd/MM/yy`
- o `yyyy/MM/dd`
- o `MM/dd/yyyy`
- o `MM/dd/yy`
- o `yy/MM/dd`

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

- o `Milliseconds`
- o `yyyy-MM-dd'T'HH:mm:ss.SSSZ`

7. Optionally, use the **Default Value** column to enter a default value that you want to pass through the orchestration input.

(Release 9.2.6.2) To specify the default values for orchestration inputs that are in an array:

- a. Click the **Default Array** icon in the Default Value column. A pop-up window containing the number of columns in your array is displayed.
- b. Make changes to the array as required. You can enable the **Raw** option if you want to use the JSON format.

Note: You can click the **Array Inputs** icon in the Run Orchestrations window to add or modify the array inputs. See [Testing an Orchestration](#).

Starting with Tools Release 9.2.6.3, when mapping values, you can select **<Space>** from the Default Value drop-down menu.

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

To generate orchestration inputs from an orchestration component:

1. In the orchestration design page, click the component from which you want to generate the inputs. The action control icons are displayed for the component.
2. Click the **Transformations** icon.
3. In the Transformations dialog box, click the **Add Inputs to Orchestration** button.
4. Click the **Start** node and then select **Inputs and Values** icon from the action control icons. In the Inputs and Values dialog box, the inputs that are defined in the component appear as inputs to the orchestration.
5. Use the **X** button at the end of each row to remove any unnecessary inputs.
6. Modify the **Value Type** and **Default Value** columns as needed.

7. Click on the canvas to close the Inputs and Values dialog box and then click Save.

See *Mapping Orchestration Inputs* to map these inputs to the appropriate component inputs.

Defining Files as Orchestration Input (Release 9.2.6)

You can provide a file or files directly as orchestration input and you can choose to process the files individually or as an array. You can use the file content in many ways after you add the file content as an input to the orchestration. You can write a custom script to read and parse the contents of the file for use in subsequent orchestration steps, or pass the content as input to an external system in an outbound REST connector, or upload the file to EnterpriseOne as a media object attachment.

The **Files(O)** button is enabled in the Run Orchestrations window for orchestrations that have a defined file input, and you can upload the files.

To define a file as an input to an orchestration:

1. On the Orchestrations design page, click the **Start** node.

The action control icons are displayed above the Start node.

2. Click the **Inputs and Values** icon.

3. Click the **File Inputs** tab.

4. Select from these options:

- a. **No File Inputs:** This option is selected by default.

- b. **Process Files Individually:** Select this option to process the files individually.

- i. Enable the **Overwrite Existing Files** option to use the original file name for all the files throughout the processing. Disable this option to assign a unique file name to each file.
- ii. Enable the **Keep Temp Files** option to save the files in the root temporary directory for an indefinite period of time. If you disable this option, the files will be saved in the user's session temporary directory and all the files will be deleted when the session ends.
- iii. To define an individual file name pattern, enter a name in the File Name Pattern field and select the required content type from the Content-Type drop-down menu. The system automatically displays the values in the Variable Name field as `fileName_0`, `fileName_1`, and so on. You can override the system generated values in the Variable Name field.

For example, if you enter the file name patterns as `order*` and `detail*`, and select the content type as **application/pdf** and **image/jpeg**, respectively, the orchestration will accept only two files, one is a file with a name starting with `order` that has a content type of `application/pdf`, and the other is a file with a name starting with `detail` that has a content type of `image/jpeg`. The system will

display an error message if you try to add any file with a name that is not starting with `order` or `detail`, or if you add a file other than the specified content type.

You can enable the **Required** option to mark a file as required and click the **Delete** icon to delete a file name pattern.

Inputs and Values

Input Format: Generic View Example Input Job Queue: Default

Orchestration Inputs **File Inputs** Variables System Values Environment Pro

No File Inputs
 Process Files Individually
 Process Files As An Array

Overwrite Existing Files ⓘ
 Keep Temp Files ⓘ

File Name Pattern	Content-Type	Variable Name
order*	application/pdf	orderFile
detail*	image/jpeg	detailFile

Note: The system displays a blue dot on the File Inputs tab to indicate that the files are defined for orchestration input.

- c. **Process Files As An Array:** Select this option to process an iterative array of files.
 - i. Enable the **Overwrite Existing Files** option to use the original file name for all the files throughout the processing. Disable this option to assign a unique file name to each file.
 - ii. Enable the **Keep Temp Files** option to save the files in the root temporary directory for an indefinite period of time. If you disable this option, the files will be saved in the user's session temporary directory and all the files will be deleted when the session ends.
 - iii. In the File Array Name field, enter an array name.
 - iv. Select a value from the Content-Type drop-down menu. If you leave this field blank, you can process the files with any content type.
 - v. Enable the **File Array Required** option to mark the file array as required.
5. You can now access the Run Orchestrations window, and click the **Files(0)** button to drag and drop, or browse and upload files. The system updates the number next to **Files** in the button title depending on the number of files you upload. For example, if you upload two files, the button name is displayed as **Files(2)**. Click the **Run** button to process the input files and verify the results in the Output tab.

Note: In the Run Orchestration window, the Content-Type field displays the value as **multipart/form-data** by default, when file inputs are defined in the orchestration.

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

Note: [Click here to view an OBE of this feature.](#)

Using Environment Properties in Orchestrations (Release 9.2.6.4)

Starting with Tools Release 9.2.6.4, you can use orchestrations to access environment properties configured in the F951000 table using the P951000 (Business Service Property) application.

Environment-level properties enable orchestrations to behave differently based on the environment in which they are processed. The F951000 table is stored in the Control Tables data source and the records are stored separately for each environment and therefore may have different values.

Configuring Environment Properties

To configure environment properties:

1. Log in to the JD Edwards EnterpriseOne application.
2. Use the Fast Path to access the P951000 application.
3. Click the **Add** button. The Add BSSV Property window is displayed.
4. In the Key field, enter a name that uniquely identifies the business service property. This name cannot be changed. The length of the property key can be up to 255 characters.
5. In the Values field, enter a value that defines a specific criterion.
6. In the Description field, enter a phrase or a sentence that identifies the purpose of the business service property.
7. In the Level field, select the AIS option. This selection enables you to access the property from the Orchestrator Studio.
8. Click **Save**.

For more information, see “[Adding a Business Service Property Record](#)” in the *Business Services Development Guide*.

Alternatively, you can access the environment properties using the Tools drop-down menu in the Orchestrator Studio.

When you click the Environment Properties link in the Tools drop-down menu, you will be automatically signed in to the EnterpriseOne application and directed to the P951000 application.

Using Environment Properties in Orchestrations

The records that have the AIS option defined for the Level field in the P951000 application are displayed in the Inputs and Values window of an orchestration. You can click the Start icon and select Inputs and Values to access the Inputs and Values window. The available properties are displayed in the Environment Properties tab.

Inputs and Values

Input Format: Generic View Example Input Job Queue: Default Job Queue Scope: System

Orchestration Inputs	File Inputs	Variables	System Values	Environment Properties	Values From Steps
Property	Description		Value	Value Type	
ISPROD	Is Production Instance		FALSE	String	
PROCESS_EX_RATE	Execute Exchange Rate Processing		TRUE	String	

Note: The properties that do not have the level defined as AIS in the P951000 application are not displayed in the Environment Properties tab.

The Environment Properties option is also available in the Transformations window of all the steps of the orchestration. In the following screenshot, the property value is passed to a rule.

Transformations

TestProperty Rule
+ Add Inputs to Orchestration

Auto Map Iterate Over ▼

Input	Available Values	Default Value
PropertyValue	<div style="border: 1px solid #ccc; padding: 5px; margin-top: 5px;"> ISPROD - Is Production Instance System Time Orchestration Input Environment Environment Properties ISDEV ISPROD - Is Production Instance PROCESS_EX_RATE - Execute Exchange Rate Processing </div>	

Note: Environment variables are cached for orchestration runtime. Therefore, if any new values are added, removed, or changed, you must clear the AIS Orchestration caches to apply the new changes. This step is also applicable if you want to debug orchestrations. If you see differences in the properties between the orchestration input dialog and the orchestration debugger, use the Refresh Cache button in the Run Orchestration page to synchronize the cache with the values in the database.

Setting Up Job Queue and Scope for an Orchestration (Release 9.2.4.4)

You can specify how you want to process an orchestration: in a single-threaded queue, in a fire-and-forget multithreaded queue, or in the default queue. Using this feature, you can have more control over the runtime processing of orchestrations.

The orchestrations are executed in one of these queues:

Queue Name	Job Queue	Job Queue Scope	Description
JDE_DEFAULT	Default	System	This is the default queue for all the requests entering the system. The orchestration is run synchronously, and therefore the caller receives a response after the orchestration is processed. This queue can run the number of jobs that is specified in MaxConcurrentJobs in the rest.ini file. See <i>Maximum Concurrent Job Limit</i> .
JDE_ASYNC	Fire and Forget	System	This queue is used for asynchronous orchestration requests. For orchestrations sent to this queue, a caller receives an immediate response with a 202-Accepted status and the orchestration then continues to be processed on the server until completion. This queue is also restricted by the MaxConcurrentJobs limit. See <i>Maximum Concurrent Job Limit</i> .
JDE_SINGLE	Single Threaded	System	In this queue, one job is executed at a time based on the First In First Out (FIFO) method. Use this queue if you want the request to be processed in the order that it is received on the server.
JDE_USER_SINGLE	Single Threaded	User	This queue is started only if a user requests to run an orchestration on this queue. It allows requests from each user to be processed on a FIFO basis without any dependency on the completion of the orchestrations being processed on the system in a single threaded queue.
JDE_USER_ASYNC	Fire and Forget	User	This queue is started only if a user requests to run an orchestration on this queue. This queue allows user-specific asynchronous execution, and therefore the processing of these requests is not dependent on the completion of the orchestrations being processed on the system in a fire and forget queue.

Overriding the AIS Job Queue and Scope for Individual Orchestrator Calls

The system uses the AIS Job Queue and Scope settings that you define in the Orchestration design page as the default settings for the orchestration. You can override these settings for an individual orchestration call by using the `jde__jobQueue` parameter to pass a different queue in the body of the REST API call.

For example, to specify that an orchestration instance runs in the user's queue and in single-threaded mode, you can pass the following parameter in the body (JSON) of the request: `"jde__jobQueue" : "JDE__USER_SINGLE"`.

The following five values are valid overrides for queue and scope:

Note: The valid override values include a double underscore after `JDE`.

`"JDE__SINGLE"`, `"JDE__USER_SINGLE"`, `"JDE__USER_ASYNCH"`, `"JDE__ASYNCH"`, `"JDE__DEFAULT"`.

Maximum Concurrent Job Limit

The maximum values that apply to each of the queues are specified in `MaxConcurrentJobs` in the `rest.ini` file. This value can be changed in the Server Manager. The default value for `MaxConcurrentJobs` is 100. When a queue hits its maximum number of jobs, it does not reject the job, but it keeps the job in a waiting stage until a thread becomes available. A warning is recorded in the log once a minute for such instances.

Setting Up Job Queue and Scope

To set up the job queue and the scope for an orchestration:

1. On the Orchestrations design page, click the **Start** node.

The action control icons are displayed above the Start node.

2. Click the **Inputs and Values** icon.

Starting with Tools Release 9.2.7.4, the Job Queue and the Job Queue Scope drop-down lists are moved to the Run Options window. If you are using Tools Release 9.2.7.4 and later, click the **Run Options** icon.

3. From the Job Queue drop-down list, select the values as required:
 - Single Threaded. Use this option if you want to wait and receive a response after the orchestration is processed. This option enables you to run the orchestration in the order it is received by the server.
 - Fire and Forget. This option enables the queue to run asynchronously. The caller will only receive an acknowledgment that the job was received. The orchestration will continue executing on the server until completion.
 - Default. Use this option if you want to wait and receive a response after the orchestration is processed. At a time, this queue runs the number of jobs that is specified in `MaxConcurrentJobs` in the `rest.ini` file. The orchestration will undergo a waiting period before it is processed only if the queue is full.

Note: If you select the Default option, the Job Queue Scope is automatically populated as System.

4. From the Job Queue Scope drop-down list, select the values as required:
 - System
 - User. Selecting a user-specific queue allows a new queue to be generated for each user that requests the orchestration. The user-specific queues will have precedence over the system queues.
5. Click **Save** to save your changes. The saved queue will be used every time the orchestration runs unless the queue is overridden.

Mapping Orchestration Inputs

On the Orchestration design page, access the Transformations dialog box to map orchestration inputs to the inputs that are defined in an orchestration step, such as inputs in a rule, cross reference, white list, or any other component. The Transformations dialog box includes an Auto Map button for automatically mapping any matching inputs in the orchestration and the 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 dialog box to map the inputs. This facilitates the reuse of components, and eliminates the need to create a new component 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. In this example, the orchestration inputs have the same name as the inputs in the service request. These inputs were automatically mapped using the Auto Map button.

Input	Available Values	Default Value
EquipmentNumber	EquipmentNumber ▼	
NewFuelMeterReading	NewFuelMeterReading ▼	
NewHourMeterReading	NewHourMeterReading ▼	

Initially, when you have a small number of orchestrations in your system, it is recommended to keep all the input names consistent among the orchestration and all the components (orchestration steps) used by the orchestration. But as your library of orchestrations and orchestration components increases, instead of creating new components with input names that match the names of 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, click the component to which you want to map orchestration inputs. The action control icons are displayed for the component.
2. Click the **Transformations** icon.
3. In the Transformations dialog box, click the **Auto Map** button to map matching inputs. The Studio automatically maps inputs with the same name, and the matching orchestration inputs appear in the Available Values column next to the matching input in the Input column.
4. To map an orchestration input to an orchestration step input that does not have the same name:
 - a. In the grid, click the drop-down menu in the **Available Values** column.
 - b. Select the orchestration input to map to the orchestration step input.
5. You can map a literal value to the orchestration step input by entering a value in the **Default Value** column.

Starting with Tools Release 9.2.6.3, when mapping values, you can select **<Space>** from the Default Value drop-down menu.

6. If you are mapping values from an array, you can configure the orchestration to iterate over the orchestration step so that the step is called once for each row in the array:
 - a. After defining the array in the Inputs and Values dialog box, click the **Iterate** Over drop-down list in the Transformations dialog box and select the name of the array input.
 - b. Map the orchestration inputs for the array to the inputs that are defined in the orchestration step.

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 that is 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.

The following figure shows an example of a data request that is configured with a data set named "Employees" and will retrieve the address book number and tax ID of all employee records in EnterpriseOne.

Home > Data Requests > Retrieve Employee Tax Info Tools ▾

Name: **Retrieve Employee Tax Info** Product Code: 55 - Reserved for Clients Version: 3

Description: Retrieve employee tax information Category: Category 1 Share

Table/View Name: F0101 Address Book Master Load Get View from Form Aggregation Data Set Variable Name: Employees Options Preview

Filter: x

Description	Data Dictionary						
Address Number	AN8	⌵	⏪	Σ	👤	👤	☰
Long Address	ALKY	⌵	⏪	Σ	👤	👤	☰
Tax ID	TAX	⌵	⏪	Σ	👤	👤	☰
Alpha Name	ALPH	⌵	⏪	Σ	👤	👤	☰
Description Compressed	DC	⌵	⏪	Σ	👤	👤	☰
Business Unit	MCU	⌵	⏪	Σ	👤	👤	☰
Industry Class	SIC	⌵	⏪	Σ	👤	👤	☰
L	LNGP	⌵	⏪	Σ	👤	👤	☰

Filter Criteria

Match Type: Match All Match Any


Description	Operator	Values
✕ Sch Typ	is equal to	E

Return Fields and Variable Names

Description	Variable
✕ Address Number	ABNum
✕ Tax ID	TaxID

Order By

Configuring a Data Request to Retrieve a Data Set

1. On the **Data Requests** design page, enter a name for the data set in the **Data Set Variable Name** field.
2. Define the 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** section, enter a variable name for each return field.

Starting with Tools Release 9.2.8.2, you can reorder all the selected columns in each section of the data request, including the Returns and Aggregations, by dragging and dropping the rows. The system preserves the order that you

set when you **Save** the changes and reflects this order in Orchestration Outputs when the Data Request is added to an orchestration. This feature improves readability and enables orchestrations to easily integrate with user interfaces or third-party systems that might be sensitive to the order of outputs.

Note: Order of name-value pairs in JSON is not important or guaranteed but in most cases, the orchestration output reflects the order set at design time. If you use Output Array to File to create a CSV file, the system reflects the order of grid data specified at design time.

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

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 Requests** design page, locate the row *<Form Name>* - Grid node in the Available Actions grid.
2. In the *<Form Name>* - Grid row, enter a name for the data set in the Variable 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 row that you want to include in the data set, slide the toggle in the **Return** column.
4. For each row that you want to include in the data set, enter a variable name so that the returned column can be mapped to a subsequent orchestration step.

Starting with Tools Release 9.2.8.2, you can reorder the outputs of a form request by clicking the **Edit Returns** icon on the Return Form Data action and by dragging and dropping the rows. The system preserves the order that you set when you **Save** the changes and reflects this order in Orchestration Outputs when the Form Request is added to an orchestration. This feature improves readability and enables orchestrations to easily integrate with user interfaces or third-party systems that might be sensitive to the order of outputs.

Note: Order of name-value pairs in JSON is not important or guaranteed but in most cases, the orchestration output reflects the order set at design time. If you use Output Array to File to create a CSV, XML, or JSON file, the system reflects the order of grid data specified at design time.

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

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 **Transformations** dialog box of the orchestration step, in the **Iterate** Over drop-down list, select the data set name that is 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**.

Working with Orchestration Output

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. Orchestration output can also include the response from a connector, custom service request, or the result of a rule (true or false).

On the Orchestration Output dialog box, you can use a script to manipulate the contents of the response to refine the orchestration output as required by the parameters in the consuming device or program.

To work with orchestration output:

1. In the orchestration design page, click the **End** node.

The action control icon is displayed for the End node.

2. Click the **Outputs and Assertions** icon.

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

Name	Field	Output	Value Type	Select
Orchestration Inputs				
	Customer		Numeric	<input type="checkbox"/>
	Branch		String	<input type="checkbox"/>
	SO_Lines		Array	<input type="checkbox"/>
f(x) Get Assert P42101 Test Valuse				
	assert_invoiceTotal		String	<input type="checkbox"/>
	assert_tax		String	<input type="checkbox"/>
	assert_orderTotal		String	<input type="checkbox"/>
Assert P42101				
Enter New Order				
Order Summary				
	Order Total	Order Total	Numeric	<input checked="" type="checkbox"/>
	Tax	Tax	Numeric	<input checked="" type="checkbox"/>
	INVOICE TOTAL	Invoice Total	Numeric	<input checked="" type="checkbox"/>

(Release 9.2.4.4)

If you have inputs defined for an orchestration, you can see the Orchestration Inputs section in the Orchestration Output window. This functionality enables you to include any input value in the final output of an orchestration. You can rename the field in the output using the Output column and you can change the type of the output field by selecting the required type from the drop-down menu in the Value Type column.

If you have variables defined for an orchestration, you can see the Orchestration Variables section in the Orchestration Output window. Here, you can include the final value (the value of the variable after all the steps) for any variable that is defined in the orchestration.

Note: Simple conversions are supported.

3. (Release 9.2.4.3) Click the **View Example Output** button in the **Define Outputs** tab to view the sample output in the JSON format for the orchestration.
4. (Release 9.2.7.2) Enable the **Null Date as String** option to return null dates as strings ("null") in the response.

This option is turned on by default for existing orchestrations and will be turned off by default for new orchestrations.

5. If you want to view the JSON code of all the return fields in an orchestration, slide the **Select All** toggle to the right.

You can individually select the return fields for which you want to preview the JSON code. You can expand each step to see the return fields. The Output column displays the name that is defined for a return field.

- a. In the **Select** column, slide the toggle of the any return fields that you want to add to your orchestration output and then exit the Orchestration Outputs page.

The fields will be returned in JSON format, which you can test on the Run Orchestrations page.

- b. You can change the name for the return fields in the **Output** column.
6. Select Groovy, JRuby, or Jython, and you can edit the provided script template in the **Manipulate Outputs** tab to manipulate the contents of the response to refine the orchestration output.

For Groovy, see *Groovy Template for Manipulating Output from an Orchestration Response* for more information.

7. (Release 9.2.7.2) In the **Example Output for OpenAPI(JSON)** section of the Manipulate Outputs tab, you can enter an example response. This example response will be included in the OpenAPI document in place of the system generated response.

You can also copy an example output in this section when using the Manipulate Outputs script to alter the output to show the modified orchestration output in the OpenAPI definitions.

8. To save the orchestration outputs, return to the Orchestrations design page and click **Save**.

Mapping Orchestration Outputs

You can individually select the return fields to include them in the output and you can expand each step to see the available return fields.

In the **Define Outputs** tab:

- In the Select column, slide the toggle to the right for the fields that you want to add to your orchestration output.
- The Output column displays the name defined for a return field. The name must be unique and it can be overridden.
- The Value Type column provides a list of types that you can use to convert the return field.

Enable the **Select All** button to select all outputs available in the orchestration. When you enable the Select All button, ensure that all the names at the same level are unique. You can disable the **Select All** button to remove all the output mappings.

Click the **View Example Output** button in the Define Outputs tab to view the sample output in the JSON format for the orchestration. (Release 9.2.4.3)

You can edit the provided script template in the **Manipulate Outputs** tab to refine the orchestration output.

For Groovy, see *Groovy Template for Manipulating Output from an Orchestration Response* for more information.

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

For the fields that are returned from a Data Request or a Form Request, the system defaults the input in the Value Type field either as a string, a numeric value, or an ISO date (YYYY-MM-DD), from the Data Dictionary data type. You can select a different value type if required. If you set Value Type of a date field as string, the value is displayed in the user's date format.

If the value is a valid JSON string, you can select the type as **Object** from the Value Type drop-down menu. This value is then added to the orchestration output as a JSON object. For this to work, the value must be a valid JSON object or JSON array.

You will see the **Return All Grid Rows** option for the arrays available in the Define Outputs tab. If you enable this option, all the rows in the array are included in the output. The Output column name defaults to the step name and becomes the array name in the output. If you disable the **Return All Grid Rows** option, only the data from the first row of the array is included in the output at the top level.

(Release 9.2.5.4) You can also enable the **Rows Only** option for the arrays to further simplify the output. If you enable this option, only the rows in the array are included in the output.

If a step iterates over an array, a step level name is defaulted in the Output column for the array that contains the output from each execution of that step. The array name can be overridden. For example, if you have a step called GetData that iterates over an array, you will see GetData in the Output column of the step. If not overridden, GetData_Repeating becomes the name of the array containing the output from each execution of that step.

Note: (Tools Release 9.2.7.4) You can save the output array to a file for any step in an orchestration that produces an array. If you are using Tools Release 9.2.7.4 and later, the icon name is displayed as **Output Array to File** instead of **Output Grid to File**.

This table lists the information displayed for the step types in the Define Outputs tab:

Step Type	Information in the Define Outputs tab
Form Request	<p>The data returned from all the forms is shown in the Define Outputs tab. If the returned data is grid data, then it is represented as an array.</p> <p>The system defaults the input in the Value Type field from the Data Dictionary of the return field.</p> <p>(Release 9.2.5) You can enable the Output Grid to File option to save the output grid to a file. See <i>Using Disk Caching</i>.</p>
Data Request	<p>A single array containing the rows returned from the database is displayed in the Define Outputs tab.</p> <p>The system defaults the input in the Value Type field from the Data Dictionary of the return field.</p> <p>(Release 9.2.5) You can enable the Output Grid to File option to save the output grid to a file. See <i>Using Disk Caching</i>.</p>
Report	<p>A predefined list of data is displayed as output when you run the report. If the Fire and Forget option is enabled for report, the output contains the available data when the report is run asynchronously.</p>
White list	<p>A predefined list of data output is displayed in the Define Output tab.</p>
Custom	<p>A custom step may return a single or multiple outputs and may contain one array.</p> <p>(Release 9.2.7.3) You can enable the Output Grid to File option to save the output array to a file.</p>

Rule	The Boolean result of the rule is available for mapping in the Define Outputs tab.
Cross Reference	The Output Keys from the Cross Reference are displayed for mapping in the Define Outputs tab. The output corresponds to the values or pipe () delimited values in the EOne Value column in P952000.
Orchestration	<ul style="list-style-type: none"> The Named Object option is displayed at the orchestration level. If you enable this option, the output from the called orchestration is grouped as an object with the provided name. You can modify the Output and Value Type fields of the top-level output from the called orchestration. The grid-level output cannot be modified and can only be included or excluded using the Select option. If the orchestration is called with the Fire and Forget option enabled in the Transformations window, only a step level output is available indicating that the orchestration was submitted in the Define Outputs tab.
Notification	You can enable the Select option in the Define Outputs tab to include the result of the notification in the output.
Connectors	<ul style="list-style-type: none"> REST - The output defined in the REST Connector is displayed in the Define Outputs tab. FTP: <ul style="list-style-type: none"> Send/Receive File - You can enable the Select option in the Define Outputs tab to include the result of the FTP operation. CSV First Row Only - The CSV columns are displayed in the Define Outputs tab. CSV Array - An array containing all the CSV columns is displayed in the Define Outputs tab. Database - A Database Connector step may return a single or multiple outputs and may contain one array. Open API- The outputs defined in the Open API connector are displayed in the Define Outputs tab. If you are calling an orchestration using the connector (available in Orchestration Studio version 8 and prior), you can enable the Select option in the Define Outputs tab to include each output from the called orchestration.

Using Disk Caching (Release 9.2.5)

Disk caching ensures performance optimization for the Orchestrator and enables it to retrieve large data sets from the EnterpriseOne tables and pass the results in the orchestration output.

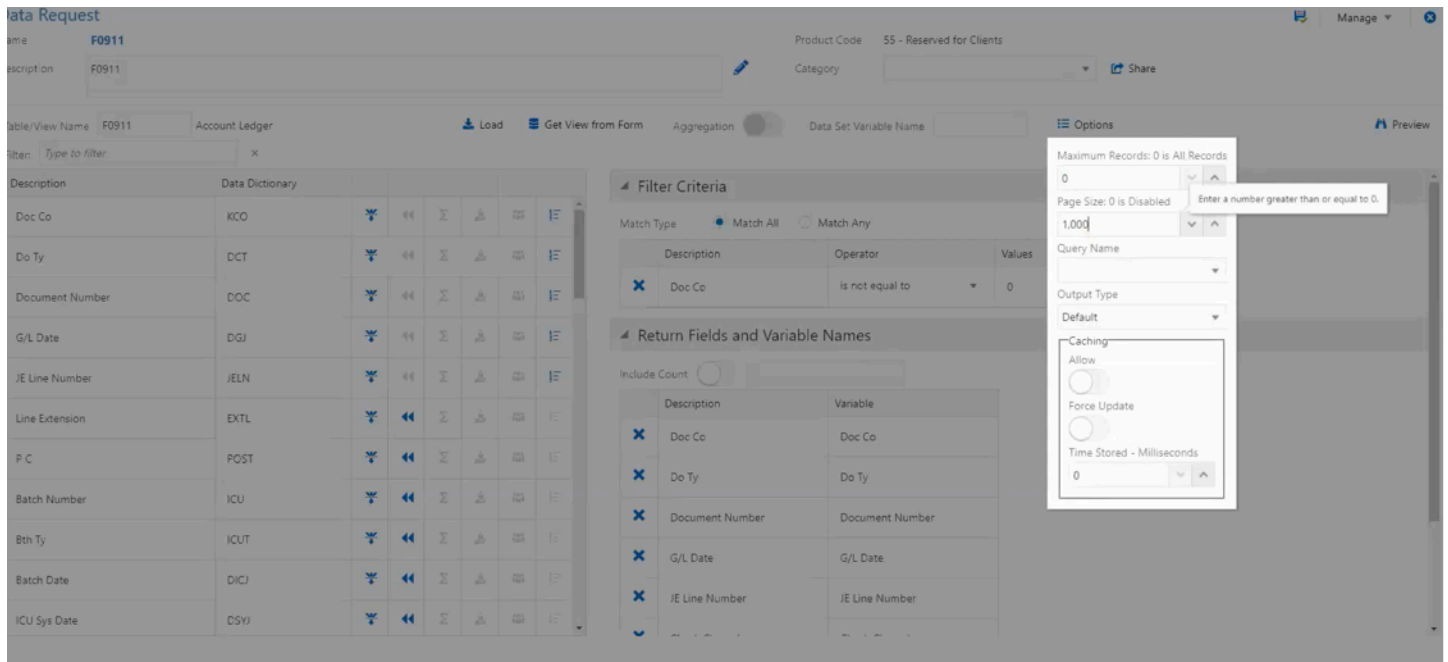
You can save the large output files from the form requests and the data requests on a disk. The output is saved on a disk only when the page size is not set to zero in the Form Request and Data Request windows.

The output files on the disk are then modified to apply the defined output mappings and the final result is displayed as output and then written to a file.

You can also define the output to include or exclude the value in the body of the response.

For example, for a data request to the Account Ledger table (F0911) which has 24000 records, you can configure the output to display all the 24000 records, 1000 records at a time.

On the Data Request window, select the **Options** icon, and enter the page size. The page size is set to 1000 by default.



In the orchestration in which you have added this data request as a step, in the Define Output tab of the Output and Assertions window, enable the **Select** option to include the required values from the data request. Click the **Output Grid to File** icon in the Select column for the data request and enable the **Output Grid to File** option and enter the file name in the **Output File Name** as required. Select a value from the **Output Type** drop-down menu. The options available are CSV, JSON, and XML.

(Tools Release 9.2.7.4) You can save the output array to a file for any step in an orchestration that produces an array. If you are using Tools Release 9.2.7.4 and later, the icon name is displayed as **Output Array to File** instead of **Output Grid to File**.

(Release 9.2.8) When you select the Output Type as CSV, the system shows the **Delimiter** field, where you can specify any separator character other than a comma. The default value is displayed as comma (“,”) in this field.

You can enable these options if required:

- **Keep Temp Files:** Select this option if you want to keep the CSV, JSON, or XML file in the temp directory. If you are using FTP to copy the file as a step in the orchestration, you can choose not to keep the files in the temporary file.
- **Overwrite Existing File:** Select this option if you want to over write an existing file with the same file name. If you do not choose this option, the files will be created with a unique identifier.
- **Remove Grid From Response** (Displayed as **Remove Array From Response** in Tools Release 9.2.7.4 and later): Choose this option if you want to exclude the records from the orchestration response to reduce the size of the payload received over HTTP.

Note: To enable disk caching, you must select the **Enable Disk Caching (Only With Page Size)** option in the cache section of the AIS Server page in Server Manager. This is the setting for orchestrations related to the disk caching. When this option is selected, the data set is returned from data request and form request in an orchestration and is written to a disk to prevent server memory overload. This setting will enable the streaming of responses from the HTML Server to a disk when the page size is indicated (it should not be zero) in the Form Request window or the Data Request window. If this option is not selected, the records are saved on the AIS Server and are not cached or stored on a disk and this may cause memory overload.

Defining Orchestration Response Messages (Tools Release 9.2.5.4)

Starting with Tools Release 9.2.5.4, you can create a custom message based on success, warning and/or error status to include in the orchestration response. For warning and error messages, you can create a custom message or allow the system to generate a message based on the existing response content. You can also choose to add information about the orchestration execution to the response.

For example, when a mobile application is used to create sales orders in EnterpriseOne, the orchestration designer can create a simple success message that says, "Sales Order 123123 is successfully created." The mobile application will easily access this message from the orchestration response and display the message to the user. Similar messages can be written for warnings and error conditions.

The following new output elements are now available in the Output tab in the Run Orchestrations window:

Note: The characters after `jde` are two underscore characters.

- `jde__simpleMessage`
- `jde__status`
- `jde__startTimestamp` and `jde__endTimestamp`
- `jde__serverExecutionSeconds`

You can define the messages in the Define Messages tab in the Outputs and Assertions window.

The success, error, and warning messages are returned using the `jde__simpleMessage` output element. When you run the orchestration, a success message is returned if the orchestration runs successfully. An error message is returned when the orchestration ends in an exception (500 status). A warning message is returned when the orchestration completes successfully but warnings or errors are returned from a form request or an orchestration step that failed, but without causing the orchestration to be terminated due to the **Continue on Error** configuration.

The messages are displayed as a string value that includes newline (`\n`) characters to help with the readability of the content.

To define messages:

1. In the orchestration design page, click the **End** node.
2. Click the **Outputs and Assertions** icon.
3. Click the **Define Messages** tab.

4. To define a success message:
 - a. Enable the **Success Message** option.
 - b. Enter the message in the text field next to the Success Message option.
 - c. If you want to add a variable to your success message, click the **Add Variable** drop-down menu next to the success message text field and select the required variable.

The variable will be appended to the end of the message using the `{ }` notation. You can cut and paste the variable from the end of the message to anywhere in the message.

Outputs and Assertions

The screenshot shows the 'Define Outputs' configuration panel. It features three message types: Success Message, Warning Message, and Error Message, each with an enabled toggle switch. The Success Message text area contains a custom message with variables: 'Orchestration executed successfully!', 'input: \${input}', and 'user: \${User Name}\${reqdate}'. Below the message fields are three more toggle switches for 'Status', 'Timestamp', and 'Execution Time', all of which are also enabled. To the right of each message field is an 'Add Variable' dropdown menu.

5. To define a warning message:
 - a. Enable the **Warning Message** option.
 - b. Enter the warning message in the text field next to the Warning Message option.

Note: If you do not enter a customized warning message, the response contains the system-generated warning message.
 - c. If you want to add a variable to your warning message, click the **Add Variable** drop-down menu next to the warning message text field and select the required variable.
6. To define an error message:
 - a. Enable the **Error Message** option.
 - b. Enter the required error message in the text field next to the Error Message option.

Note: If you do not enter a customized error message, the response contains the system-generated error message.
 - c. If you want to add a variable to your error message, click the **Add Variable** drop-down menu next to the error message text field and select the required variable.
7. Enable the **Status** option to add the `jde__status` element to the output.

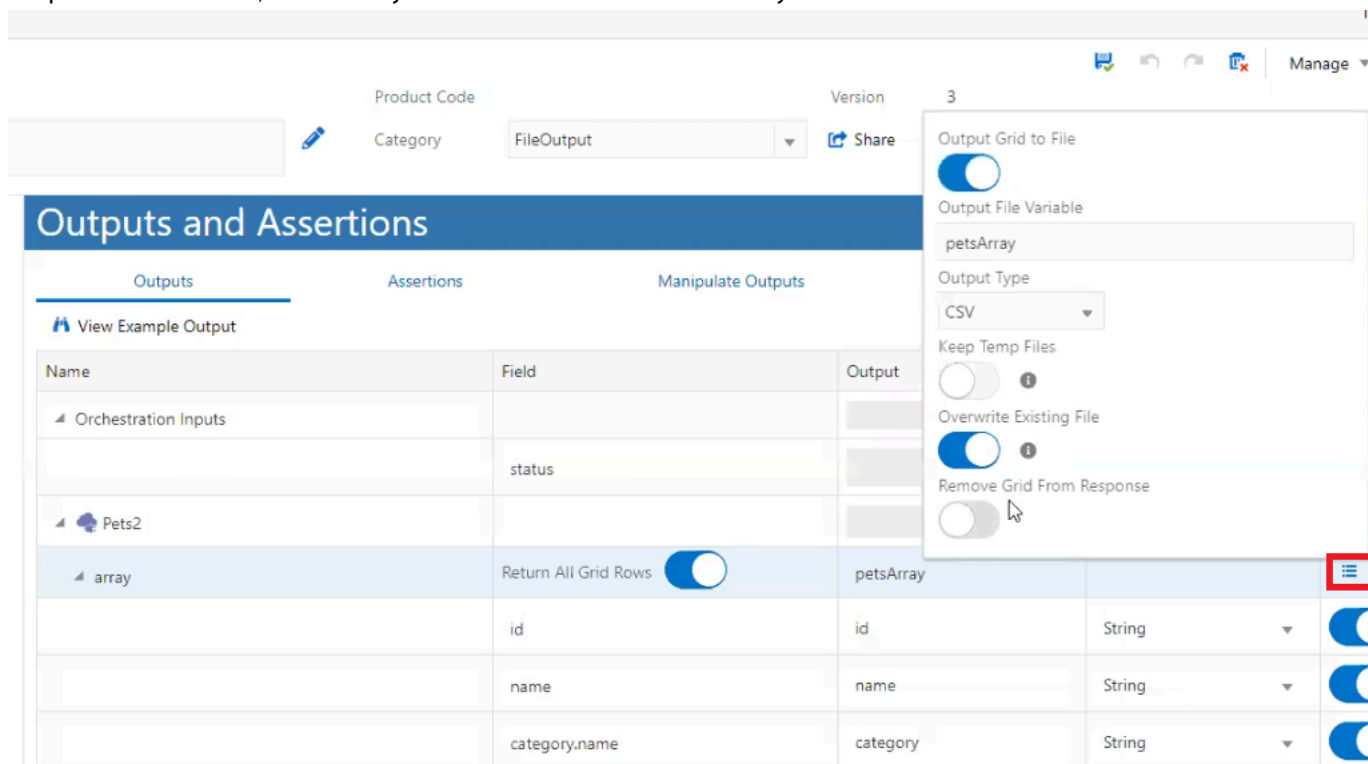
8. Enable the **TimeStamp** option to add the `jde__startTimestamp` and `jde__endTimestamp` element to the output.
9. Enable the **Execution Time** option to add the `jde__serverExecutionSeconds` element to the output.

Defining Files as Orchestration Output (Release 9.2.6)

The data that is collected or generated as part of an orchestration, and subsequently is written as files in the AIS server temporary directory, can be included in the orchestration output. The REST service call provides binary data in the response, similar to any HTTP file download response. For example, you can save the results of a query to a data request or to a form request as a JSON, an XML, or a CSV file, and then include that file in the orchestration output.

To define a file as an orchestration output:

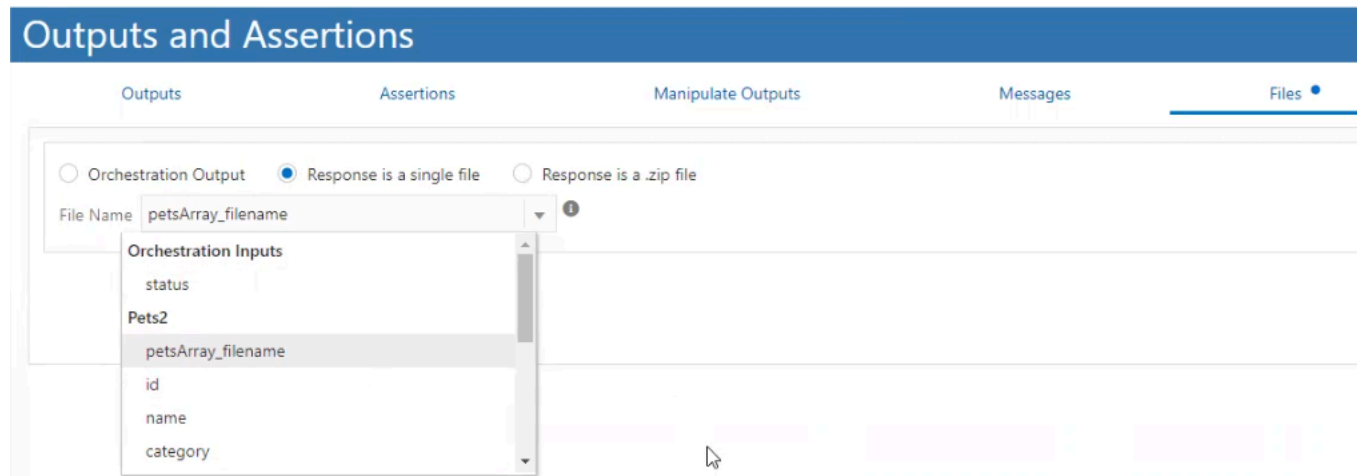
1. On the Orchestrations design page, click the **End** node.
2. Click the **Outputs and Assertions** icon, and select the **Files** tab.
3. Select from these options:
 - a. **Orchestration Output:** This option is selected by default and the response contains the orchestration output in JSON format.
 - b. **Response is a single file:** Select this option to get the response in a single file. For example, if you have enabled the **Output Grid to File** option, entered a variable name, and selected the output type as **CSV** in the Output tab of the Output and Assertions window to define the output of a REST call, then the system will build a CSV file when you run the orchestration.



Select the **Response in a single file** and then select the value of the File Name from the drop-down menu. The output file variable you defined in the Outputs tab is available in the File Name drop-down menu.

Any variable containing a name of a file in either the root or user session temp directory can be used. In this example, the file was generated from grid output. But the file could have been created in a custom

step, or could be downloaded in an attachment step. You can select the available variable that contains a name of a temp file in the File Name field.



In the Run Orchestrations window, select the **application/octet-stream** option from the **Accept** drop-down menu and run the orchestration to download the CSV file.

- c. **Response is a .zip file:** Select this option to get the output in a .zip file. Enable the **Array** option to indicate that the response is an array of file names and select the value for the Output File Variable field from the drop-down menu.

You can enable the **Include Orchestration Output** option if you want the orchestration output as a .json file within the .zip file.

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

Note: [Click here to view an OBE of this feature.](#)

Working with Orchestration Assertions (Release 9.2.5)

The Orchestration Assertions functionality enables you to specify the values that you expect to be produced by an orchestration and validate the data produced by an orchestration. For example, you might assert that a value is expected to be within a certain range or must match a specific numeric value.

When assertions are evaluated during the execution of an orchestration, a collapsible section showing all the assertions and the expected values is displayed and the failed tests are highlighted. An exception is returned only when the orchestration execution fails.

To work with Orchestration Assertions:

1. In the orchestration design page, click the **End** node.
2. Click the **Outputs and Assertions** icon.

The grid displays all the components or steps in the orchestration that contain return fields.

3. Click the **Define Assertions** tab.

Name	Field	Operator	Test Value	Select
Orchestration Inputs				
	Customer Number			<input type="checkbox"/>
Get Customer Credit Limit				<input type="checkbox"/>
Data Browser - V03012F [Credit Granting Inquiry]	Count			<input type="checkbox"/>
	Credit Limit [F03012]		0	<input type="checkbox"/>
	Alpha Name [F0101]-F0101.ALPH		0	<input type="checkbox"/>
Order Totals				<input type="checkbox"/>
Data Browser - F4201 [Sales Order Header File]	Count			<input type="checkbox"/>
	Count-COUNT		0	<input type="checkbox"/>
	Order Amount-Sum		0	<input type="checkbox"/>
Over Credit Limit				<input type="checkbox"/>
	Over Credit Limit			<input type="checkbox"/>

4. Select the Operator value as required in the **Operator** drop-down menu.
5. Define the assertions for the outputs in the **Test Value** column.

Note: You can also define assertions for data that is not selected for output in the **Define Outputs** tab.

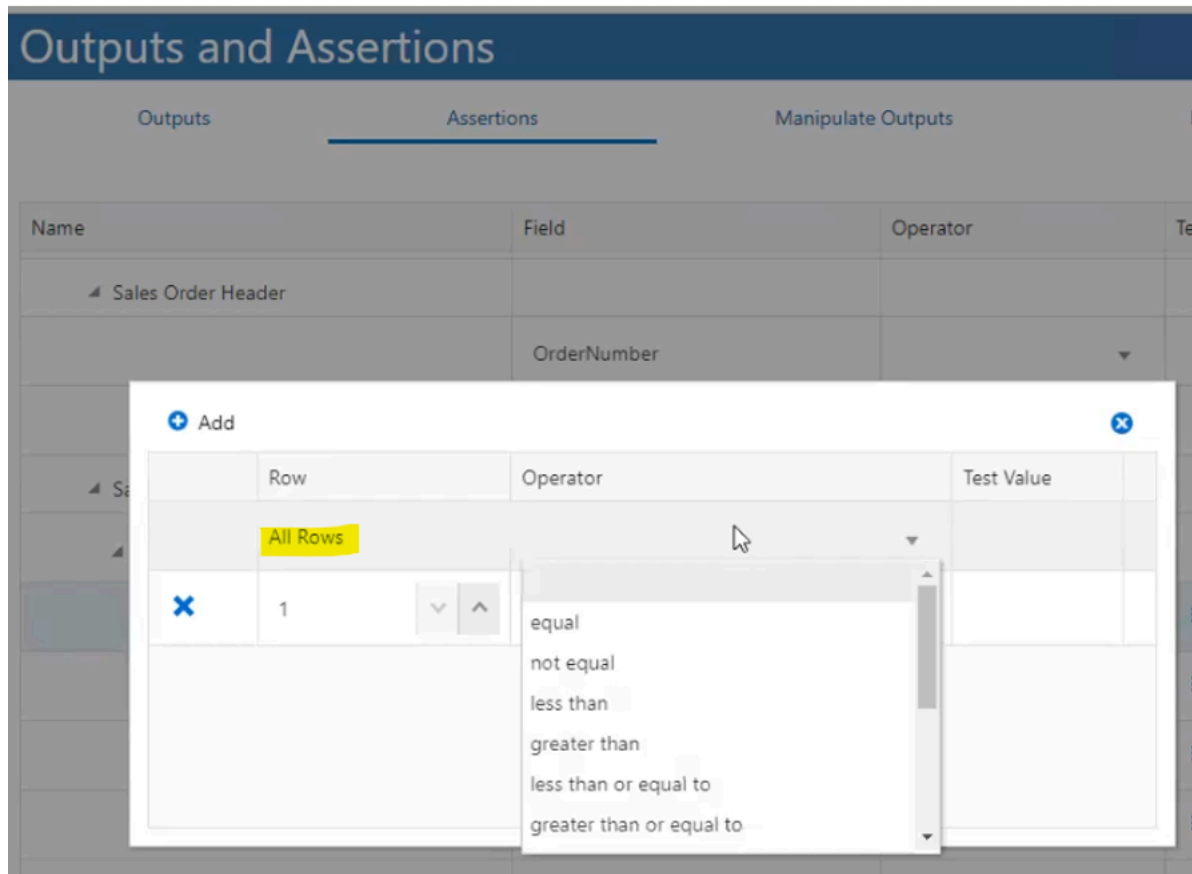
You can also disable the assertions without removing it by disabling the **Select** option. The Test Value is either a literal value or you can choose an orchestration input or variable. Variables generated from other steps are not available for assertions unless it overwrites an input or a variable.

- o If the output data is an array, click the **Define Row Assertions** icon (list icon) in the Test Value column. A window where you can define assertions for specific rows is displayed. If you define an assertion for a row number exceeding the result set, then that assertion will fail. For example, If you define an assertion for row 5 and during the execution only 4 rows are returned, all the assertions for row 5 and above will fail.

(Release 9.2.6.2) You can define an assertion criteria that is evaluated against all the rows returned by an orchestration step. For example, an orchestration might include a data request to return all the sales order lines, and the assertion can evaluate whether all the lines have a line type of S.

Click the **Define Row Assertions** icon (list icon) in the Test Value column for the output data that is an array. A window where you can define assertions for specific rows is displayed

and you can specify the Operator and Test Value as required for the **All Rows** column.



The **Count** option is available for any array generated in the outputs at the group level for the array.

- If the output is a date type, click the drop-down menu in the Test Value column to select a date based on the current date. You can also use a literal value for the date type and this literal value must be in the matching date format.
- If you want to get the test values from another step, then the values must be first declared as variables in the Variables tab of the Orchestration Inputs tab. These variables are then available as Test Values and any step can overwrite them.

6. To save the orchestration assertions, return to the Orchestrations design page and click **Save**.

If your orchestration calls another orchestration, the **Select Assertion** option is available at the called orchestration level on the Define Assertions tab. This option enables you to execute and show results of all the assertions defined in the called orchestration.

Note: You can also define assertions for any top-level outputs of a called orchestration in the calling orchestration but not for data in arrays.

Defining Assertions on a Form Request

You can define assertions in the **Order of Execution** section on a Form Request by clicking the **Edit Returns** icon.

The Recorder can capture the assertions on a Form Request, and during this process you can record the test scripts and the expected values.

For more information see [Using the Process Recorder to Create a Form Request](#)

Assertions on the Form Request are not evaluated directly and hence they must be added to an orchestration. If a form request that is added to an orchestration has assertions, you will see an **Add** button (+) in the Define Assertions tab of the Outputs and Assertions window. You can click this button to load the assertions from the step to make all the values from the Form Request the default values for the orchestration.

Name	Field	Operator	Test Value
Orchestration Inputs			
	Order Number		
Get SO ?			
Enter New Order			
Sales Order Header			
	OrderNumber	>	0
	OrderType	is not blank	
	OrderCompany	=	00001
	OrderDate	<	Today Minus 1 M...
	Customer		
Order Summary			
	Order Total		
	Invoice Total		
Sales Order Detail			
Get SO-Grid	Get SO_188_20	=	4
	Item Number		2

Evaluating Assertions

If you have defined assertions for the orchestration, the **Evaluate Assertions** option is displayed in the Run Orchestrations window.

You can see these details of the assertions in the Assertions collapsible section if you enable the Evaluate Assertion option:

- A Summary of the total assertions evaluated and passed.

- Individual assertions grouped by step. A group of assertions are displayed in a collapsed section if all the assertions in the group are passed.

Click the **Save Assertions** button to download the results of the assertions in the JSON format.

If you are using a third-party software to run your orchestrations, and if you want evaluate the assertions and get the assertion results in the response, you can add **jde_assertions** with any value to the input message.

Creating a Stateful Orchestration (Release 9.2.8.3)

This section contains the following topics:

- [Understanding a Stateful Orchestration](#)
- [Defining the Orchestration as Stateful](#)
- [Understanding the Stateful Orchestrations Application](#)
- [Importing and Transferring Considerations](#)

Understanding a Stateful Orchestration

The lifespan of an orchestration consists of a request, a prescribed set of steps, and a response. However, certain business processes must be long-lived and allow for intermediate processes that are sometimes dependent on third-party services or humans.

By defining an orchestration as Stateful, you can enable it to tolerate time gaps in processing, pauses for external systems to respond, starting and stopping between steps, and other asynchronous behaviors. This is done by using the new Pause step. When a Pause step is encountered, the state of the orchestration is stored with an instance ID in a new table (F980080), which can be viewed using the Stateful Orchestrations application (P980080) through the new Stateful Orchestrations link in the Tools menu of Orchestrator Studio. To resume an instance, you can run the same orchestration while passing the instance ID.

The system will create a new version of the stateful orchestration when making changes if there are active instances. If there are no active stateful instances when you save the changes, the version will remain the same. If there are active instances, the orchestration is saved with a new version number. Any new instances will be processed using the new version. The old version is saved and is used when instances that were started with the old version are resumed. Once all the instances for the old version are completed, it can be deleted.

You can view the version of the stateful orchestration you want by using the Stateful Version drop-down list. Only the latest version is editable. Additionally, the system also displays the number of active instances in the Active Stateful Instances (non-editable) field.

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

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

Note: [Click here to view an OBE of this feature.](#)

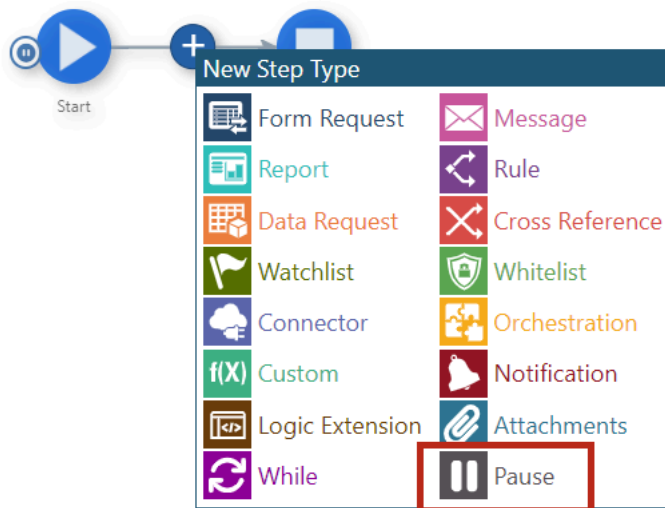
Defining the Orchestration as Stateful

To define the orchestration as Stateful:

1. Click the **Start** node of the orchestration and click **Run Options**.
2. In the Run Options window, enable the **Stateful** option. You can pause or resume the stateful orchestration using the system generated Stateful Instance ID.

When the Stateful option is enabled, the system:

- o generates an input called `stateful_Instance_ID` in the Orchestration Inputs tab of the Inputs and Values window (Start -> Inputs and Values). When you run a stateful orchestration without passing `stateful_Instance_ID`, the system will generate a new one to be saved on Pause. If `stateful_Instance_ID` is passed in the inputs, the state is retrieved using the instance ID to resume.
- o adds the **Key** option next to the non-array inputs in the Inputs and Values window. When you designate one or more inputs as keys, the system will save the passed key values along with the stateful instance. If any other instance with a matching key is started before the prior instance is completed, the system will end that process with an error. Only one active instance for each unique key is allowed. Keys are optional, and therefore, if the keys are not specified, there will be no restrictions when you start new instances.
- o displays the **Require Step ID to Resume** option. When you enable this option, the system generates an input called `step_Instance_ID` in the Inputs tab of the Inputs and Values window. The `step_Instance_ID` identifies a specific Pause point and changes each time the orchestration is paused. This can be used to invalidate old links used to resume a stateful orchestration, so once the link in a message or notification is used, it will no longer work.
- o displays an indicator pause (Stateful) icon besides the Start node of the orchestration.
- o enables you to add the Pause step to your orchestration from the New Step Type window. You can add the Pause step any number of times between the other steps as required. See [Adding Steps to an Orchestration](#).



Optionally, you can designate a resuming orchestration to resume your paused instances from a scheduled orchestration. You can select any Pause step in the orchestration and select **Edit**, click **Resuming Orchestration**, and then search for and select the orchestration. A resuming orchestration is a special orchestration that runs on a schedule and can resume any paused stateful orchestration on that schedule.

Note: For reference, the **JDE_ORCH_H98I_Resume_Stateful_Orchs_Every_15_Mins** orchestration is available as an ESU on *Update Center*. By choosing this orchestration as the Resuming Orchestration for a Pause step, whenever an orchestration instance is paused at that step, it will get automatically resumed every 15 minutes (provided JDE_ORCH_H98I_Resume_Stateful_Orchs_Every_15_Mins is started in the Scheduler). This can be used in conjunction with a While step to periodically check on the status of a long running process and/or send update emails on the status. You can download this orchestration, save it with an appropriate name, and associate it with a different schedule as needed.

Understanding the Stateful Orchestrations Application

The JD Edwards EnterpriseOne provides the Stateful Orchestrations (P980080) application to view and review the cumulative data of the stateful orchestration instances. The table in this application displays the record details such as Orchestration Name, Version, User, Paused Step, Stateful Instance Key, Start and Last Update Data and Time, and so on.

You can access this application from the Tools drop-down menu in the upper-right corner of the Orchestrator Studio. You can filter and view the details based on the Stateful Orchestration ID and Show Completed Instances fields.

Importing and Transferring Considerations

Before (or when) you import and transfer stateful orchestrations into another path code, you must consider the following:

- Object Management Workbench-Web (P98220W) is unable to increase the stateful version and save the old version to Stateful Orchestration Version Table - F980081, when there are active instances of a stateful orchestration. Therefore, you must import and transfer through P98220W only for a new stateful orchestration or when there are no active instances and all old versions have been deleted in the target environment.
- If the stateful orchestration does have active instances in the target environment that cannot be completed or deleted, you can use Orchestrator Studio to export and import the orchestration. If you import through Orchestrator Studio, the system will move the old stateful version to F980081 so the existing instances can resume using the old version and set the current stateful version to one higher than the last for all the new instances.

Creating Schedules for Orchestrations

This section contains the following topics:

- *Understanding Schedules*
- *Creating a Schedule*
- *Adding a Schedule to an Orchestration*

Understanding Schedules

The Orchestrator Studio enables you 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, and sends a notification to the appropriate users when the threshold is exceeded.

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

You must have been granted required UDO security to create schedules. Schedules are managed as UDOs. As a result you need the proper UDO security 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 Scheduler page to start and stop schedules.

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 a.m.).

To create a schedule:

1. Create a schedule as described in [Creating a Component](#)
2. In the **Schedules** design page, enter a short description with a maximum of 200 characters. This description should clearly describe the frequency of the schedule so that the schedule can be attached to notification as needed.
3. Perform one of the following steps:

- o In the **Scheduled to run every** 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.

- o 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 and intervals, and on fixed dates. Many third-party Cron expression generators are available that can help you create a Cron string.

4. Click the **Save** or **Save As**.

A new schedule is saved for the first time 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.

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 the Scheduler page. See, [Working with Scheduler](#) for more information.

Alternatively, you can use the REST API services to manage the schedules. 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 Guide](#) .

Note: In order to use the Scheduler with an orchestration, it must be a version 2 or version 3 orchestration. You can use the Orchestrator Studio 9.2.4.0 to migrate the version 1 and version 2 orchestrations to version 3 orchestration.

To add a schedule to an orchestration:

1. Access the orchestration to which you want to add a schedule.
2. In the canvas of the orchestration, click the **Start** node.
The Orchestrator Studio displays the action control icons.
3. Click the **Schedule** icon.
4. On the Schedules dialog box, click the **Select Schedules** button.
5. Select a schedule from the components list.

The selected schedule is added to the Schedules dialog box. You can edit or remove the schedule and then close the Schedules dialog box.

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

Note: If the message "You don't have authority." is displayed while adding a schedule to an orchestration, check if your token has expired.

A scheduled orchestration may need specific inputs that are different from the default values when running the orchestration in other scenarios. Starting with Tools Release 9.2.7.2, you can override the inputs that are associated with the scheduled runs of an orchestration.

To override the defined inputs when an orchestration is running on a schedule, click the **Start** node and then click the **Schedule** icon. In the **Inputs for Schedule** section of the Schedule window, enter the input values in the **Value** field for the top level inputs (or leave the fields empty to use the default value if defined, shown in italics). If the input is an array, click the **Array Inputs** icon to set array inputs. You can also enable the **Raw** option to add inputs in the JSON format.

Click the **Clear Input** button to delete the overridden input values and reload the list with the currently defined inputs.

Working with Scheduler

This chapter contains the following topics:

- [Understanding Scheduler](#)
- [Prerequisites](#)
- [Accessing the Scheduler](#)
- [Scheduler User Interface Features](#)
- [Working with Scheduler](#)
- [Starting and Stopping Jobs using Scheduler](#)

Understanding Scheduler

The Orchestrator Studio provides a user interface, which is called the Scheduler user interface, where you can view the notification and orchestration jobs along with the attached schedules. Using the Scheduler user interface, you can view all the jobs and perform tasks, such as starting and stopping individual or multiple jobs. The Scheduler user interface eliminates the need to use third-party applications to start and stop jobs. The Scheduler user interface enables you to

designate a job that should be automatically started whenever the AIS server is started. This enables the scheduled jobs to be automatically started when the AIS server is restarted.

You must have been granted proper UDO security to work with the notification and orchestration jobs on the Scheduler user interface.

The Scheduler user interface provides information about the orchestration jobs that are running and the ones that need to be started.

The scheduler runs as a process on the Application Interface Services (AIS) server. You can view only those schedules on the AIS server that you are currently logged-in to. For information on designating the AIS server as a scheduler, see *"Configuring AIS Server Manager Settings" in the JD Edwards EnterpriseOne Application Interface Services Server Reference Guide*.

Note: When a Scheduler server points to a JAS server that supports an environment, which is also supported on another Scheduler server pointing to another JAS server, two instances of the same job could exist. This is because the job is running for a specific environment and that environment is supported in two different scheduler configurations. This is applicable when the scheduler is not resilient.

Prerequisites

Complete the following prerequisites:

- You must be running a minimum of EnterpriseOne Tools Release 9.2.4.
- Deploy an Application Interface Services (AIS) Server.

You can use an existing AIS Server or deploy a new AIS Server instance through Server Manager only for running orchestrations.

Designate an AIS server as the Scheduler server. See *"Configuring AIS Server Manager Settings" in the JD Edwards EnterpriseOne Application Interface Services Server Reference Guide*.

- Deploy the Orchestrator Studio 9.2.4. See *Getting Started* for details on the latest release.
- Ensure that the Notifications and Orchestrations features are enabled and that all related UDO security is set up properly.

For more information, see:

- *Managing UDO Feature Security* in the *JD Edwards EnterpriseOne Tools Security Administration Guide*
- *Define Allowed Actions for UDO Types* in the *JD Edwards EnterpriseOne Tools Security Administration Guide*
- *Managing Orchestrator Studio Security* in this guide

For information on scheduler resilience through the use of a database with your scheduler, see *Configuring AIS Server Manager Settings* in the *JD Edwards EnterpriseOne Tools Security Administration Guide*.

Accessing the Scheduler

To access the Scheduler user interface:

1. Log in to JD Edwards EnterpriseOne Orchestrator Studio.
2. On the Orchestrator Studio Home page, click the Scheduler icon to access the Scheduler user interface.

Set Up User Access to the Scheduler Program

An administrator must use EnterpriseOne application security to enable user access to the Orchestrator Studio (P98I0000) and Scheduler (W98I0000B) programs. See "[Managing Application Security](#)" in the *JD Edwards EnterpriseOne Tools System Administration Guide* and "[Managing Orchestrator Studio Security](#)".

Enable UDO View Security to Orchestrations, Notifications, and Schedules

Users must have UDO view security access to the orchestrations, notifications, and schedules that they want to work with using the Scheduler user interface. For more information about UDO view security, see "[Managing UDO View Security](#)" in the *JD Edwards EnterpriseOne Tools System Administration Guide*.

Scheduler User Interface Features

Use the Scheduler user interface to view, start, and stop notification or orchestration jobs with their associated schedules. This user interface enables you to review the state of the jobs that are running.

- **Jobs** list. The Scheduler user interface displays a list of existing notification or orchestration jobs. By default, the list is displayed in the ascending order of the notification and orchestration names.
 - **Select or Select All** check box. Click the check box in the individual row to select specific jobs, or click the Select All check box to select all the jobs. Note that you can select only those jobs that are created by the currently logged-in user and have a schedule attached.
 - **Sort Order** button. Enables you to sort the jobs in ascending or descending order. When you hover the mouse over the column header, the Sort Order button is displayed next to column name. By default, the jobs are sorted in the ascending order of Notification/Orchestration Name.
 - **Filter** field. Search for specific notification or orchestration jobs in the list. The search runs over values in the Notification/Orchestration Name, Schedule Name, User, Environment, and Role fields. You can also filter the jobs based on the Notification ID and the Orchestration ID.
 - **Scheduler uptime (d:h:m)**. Displayed when the scheduler server is running. Displays the duration for which the scheduler has been running in days, hours, and minutes.
When the scheduler server is not running, the message "Scheduler not started" is displayed. For information on designating the current AIS server as the Scheduler server, see [Configuring AIS Server Manager Settings](#).
- Note:** If the current AIS server is not designated as a scheduler server, a message "AIS Server not designated as Scheduler" is displayed.
- **Restore** button. Updates the notification and orchestration jobs displayed in the table so that the changes to the job information are reflected. For example:
Jobs could have been started or stopped by another user by using the Scheduler user interface.
Changes could have been made to the jobs using REST services.

Changes could have been made to the jobs using the Work With Categories (P980059) application and so on. When using the Restore button, the values in the Filter field and drop-down selections are preserved.

- **i (About).** Displays a dialog box that provides two sets of information: Scheduler information and JAS server information.
- **Scheduler Server.** Indicates whether the AIS server to which you are currently logged-in is designated as a scheduler in Server Manager.
 - **Scheduler Information**
 - **Resilient.** Displays whether the scheduler is resilient.
 - **Started.** Displays whether the scheduler is running.
 - **Running Since.** Displays the date on which the scheduler started running and the duration for which the scheduler has been running.
 - **JAS Server Information**
 - **Host.** Displays the host JAS server URL.
 - **Environment.** Displays the environment to which the user is currently signed in.
 - **Role.** Displays the role that the user has currently signed in as.

Note: If the current AIS server is not designated as a Scheduler server and the Scheduler server has been started using REST API, the Scheduler Server, Started, and Running Since fields are displayed in red color indicating that there is a configuration error. Also, you will not be able to start or stop any of the orchestration and notification jobs.

- **Close.** Exit the user interface.

Working with Scheduler

Using the Scheduler user interface you can perform the following:

- Use the **Start selected** button to start an individual or multiple notification and orchestration jobs that you have selected. The Start selected button is enabled only when you select a job from the notification and orchestration jobs list.

When you click the button, the selected jobs are started and the Scheduler user interface displays the following information in a dialog box:

- **Jobs Selected.** Displays the number of jobs that are selected.
- **Jobs Started.** Displays the number of jobs that have been started.
- **Jobs in Error.** Displays the number of jobs that have errors or were not started.
- **Already Started.** Displays the number of jobs that are already running among the selected jobs.

Alternatively, you can start an individual job by enabling the Start/Stop toggle in the Started column located in each row.

- Use the **Stop selected** button to stop an individual or multiple notification and orchestration jobs that you have selected. The Stop selected button is enabled only when you select a job from the notification and orchestration jobs list.

When you click the button, the selected jobs are stopped and the Scheduler user interface displays the following information in a dialog box:

- Jobs Selected. Displays the number of jobs that are selected.
- Jobs Stopped. Displays the number of jobs that have been stopped.
- Jobs in Error. Displays the number of jobs that have errors or were not stopped.
- Already Stopped. Displays the number of jobs that are already stopped among the selected jobs.

Jobs Selected:	37
Jobs Stopped:	4
Jobs in Error:	0
Already Stopped:	33

Alternatively, you can stop an individual job by enabling the Start/Stop toggle in the Started column located in each row.

In the following example, the user is currently logged in as JDE, so the JDE user can access all the jobs that have JDE as the value in the User column. The JDE user can start the highlighted job by sliding the "Click to start job"

toggle to the right. As the "Demo Get Item Availability" job also has JDE as the user, by sliding the toggle to left the user can stop the job.

The rest of the jobs in this example are disabled because these jobs have LK as the value in the User column. The toggle in the Running column indicates that the job is already running.

<input type="checkbox"/>	Started	Type	Prod Cd ...	Notification/Orchestration Name	Auto Start	Schedule Name	User	Environment	Role
<input type="checkbox"/>	<input checked="" type="checkbox"/>		55	Demo Get Item Availability	<input type="checkbox"/>	Every 5 Minutes	JDE	JDV920	*ALL
<input type="checkbox"/>	<input type="checkbox"/>		55	Demo_Sold to Capital System RAP with Overrides	<input type="checkbox"/>	Every 10 Minutes	JDE	JDV920	*ALL
			55	Demo_Sold to Capital System RAP with Overrides	<input type="checkbox"/>	Every 10 Minutes	LK	JDV920	*ALL
			55	Demo_Sold to Capital System RAP with Overrides	<input type="checkbox"/>	Every 10 Minutes	LK	JDV920	SYSADMIN
			55	Demo_Sold to Capital System RAP with Overrides	<input type="checkbox"/>	Every 10 Minutes	LK	JDV920	TESTROLE1

Note: You can start only those jobs that are created by the currently logged-in user using the current environment, and the current role that the user is logged-in as. You can stop all the jobs that are created by the currently logged-in user. For the rest of the jobs, the Start/Stop toggle is disabled.

- Use the **Started** column to view the status of the notification and orchestration jobs for which you have access.

Select one of the following values from the Started drop-down menu:

- All. To view all the notification and orchestration jobs.
 - Started. To view only those notification and orchestration jobs that are currently being executed. This is the default option in the Started drop-down list.
 - Stopped. To view only those notification and orchestration jobs that are currently stopped.
- Use the **Type** column to filter the notification and orchestration jobs. The icon in the column indicates if the job is an orchestration or a notification.

You can click the notification or orchestration icon to open the corresponding notification or orchestration. To navigate back to the Scheduler user interface from the notification design page or orchestration, click the Scheduler link in the location link displayed at the top of page.

Select one of the following values from the Type drop-down menu:

- All. To view all the notifications and orchestrations.
 - Notifications. To view only the notification jobs.
 - Orchestrations. To view only the orchestration jobs.

- Use the **Product Code** column to view the product code that was associated with the notification or orchestration when it was created. The Product Code drop-down menu displays all the product codes for the notifications and the orchestrations in the table.
Selecting a product code from the drop-down menu displays only the associated notifications and orchestrations.
- Use the **Notification/Orchestration Name** column to view the name of the notification and orchestration. When you click the notification or orchestration name, the corresponding description is displayed.
Use the Filter field to look for specific notifications or orchestrations. Use the Notification/ Orchestration Name sort button to sort the jobs in ascending or descending order of the notifications and orchestrations names.
- Enable the Auto Start toggle to designate a notification or an orchestration job to automatically start whenever the Scheduler server is started. For information on the autostart job records, see *Scheduler Autostart Jobs Manager in the JD Edwards EnterpriseOne Tools Notifications Guide* .

Note: Enabling the Auto Start toggle does not start a job, but only designates the job to be started automatically whenever the Scheduler server is started. Use the Started toggle to start an individual job.

- Use the **Schedule Name** column to view the name of the schedule when a notification or an orchestration has an attached schedule. When you click the schedule name, its corresponding description is displayed.
Select one of the following values from the Schedule Name drop-down menu:
 - All. To view all the notification and orchestration jobs.
 - Schedule. To view only those notification and orchestration jobs that have a schedule attached. This is the default option in the Schedule Name drop-down list.
 - No schedule. To view only those notification and orchestration jobs that do not have any attached schedule.

Note: When you select the No Schedule option for the Schedule Name column, you have to select the All option for the Started column to view the jobs in the list.

- Click the **Info** icon to view the information about the job. When you click the Info icon, a dialog box appears displaying two sets of information: scheduled job information and health monitor.

Scheduled Job Information:

Displays the following data for the selected job that is associated with a specific user, environment, and role:

- Notification or Orchestration Name
- Notification or Orchestration ID
- UDO Group
- Last Run
- Last Run Duration
- Next Run
- Total Runs
- Schedule Name
- Schedule ID
- Interval

- o Consecutive Errors
- o % Errors
- o Total Errors

For the jobs that are not executing currently, only the Notification or Orchestration Name and ID, Schedule ID, and Interval information are displayed.

Health Monitor:

Displays high-level information about the performance of the selected job. A bar-chart shows up to the last 10 instances that the job 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 time taken to process the job. Move your cursor over each bar to see the date and time when the job was executed and the time in seconds taken to complete processing the job.

The screenshot displays the Orchestrator Studio interface. On the left is a table of jobs with columns for 'Started', 'Type', 'Prod Cd ...', and 'Notification/Orchestration Name'. The job 'Demo_Sold to Capital System RA' is highlighted. In the center, the 'Scheduled Job Information' panel shows details for the selected job, including Notification Name, ID, UDO Group, Schedule Name, ID, Interval, Last Run, Next Run, and Total Runs. Below this is the 'Health Monitor' section, which contains a bar chart titled 'Recent Job Executions' showing the duration of the last 10 job instances. On the right, a table shows the Scheduler uptime and a list of users, environments, and roles.

For more information, see [Creating Orchestrations with Orchestrator Studio 9.2.4](#).

The information that is displayed in the bar-chart for a job is based on the name of the notification or orchestration regardless of the user, environment, and role values.

As shown in the following example, the Scheduled Job Information section displays the information only for the highlighted job with respect to the specific user, environment, and role. In the Health Monitor section, each bar in the chart could be for any of the instances of the job with the notification name "Demo_Sold to Capital System RAP with Overrides" regardless of the user, environment, or role values.

The Info icon indicates an error when the icon is displayed in red.

- Use the **User**, **Environment**, and **Role** columns to view the user, environment, and roles information for the jobs.
 - **User**

The user who created the orchestration or notification. In the case of jobs that are currently executing, the User, Environment, and Role columns display the user who started the job.
 - **Environment**

The environment that the user who created the orchestration or notification is signed in to. The environments listed in this column are the environments that are available on the JAS server that is connected to the current AIS server. Therefore, if you log in to a different AIS server, the environments are listed based on the environments that are supported on the JAS server for that AIS server.
 - **Role**

The role of the user who created the orchestration or notification.

Use the drop-down menus to filter the jobs based on user, environment, and role. The user who is currently logged in will not be able to select, start, or stop the jobs that have been created by another user.

When you have jobs with the same notification or orchestration name running multiple times based on different user, environment, and role combinations, the Scheduler user interface lists the job that is associated with the currently logged-in user, the currently used environment, and current role of this user, as the primary record.

Next, the jobs that are associated with the currently logged-in user but created in different environments and through other roles are displayed. Followed by the jobs with the same notification or orchestration name, but associated with different user, environment, and role values. These instances of the job with the same name are displayed in italics.

The Orchestrator Studio preserves the options that are selected for the drop-down menus in the Scheduler user interface. The next time you log in, the Scheduler user interface will display the jobs according to the selection made in the previous session.

Note: For the User drop-down list, only the All Jobs and My Jobs options are preserved.

When you filter a job based on a Role or an Environment, the system preserves the filter values. However, if the next time you log in and no data exists for that selection (for example, you filtered based on a role that no longer has any jobs that are currently executing), the drop-down menu will display the value All.

Starting and Stopping Jobs using Scheduler

Using the Scheduler user interface, you can start and stop the orchestration jobs that have a schedule attached.

To start an individual orchestration job:

1. Open the **Scheduler** user interface.

2. Select the orchestration job you want to start.

Note: You can only start the jobs that are associated with the currently logged-in user, the currently used environment, and the user's current role. The currently logged-in user cannot start the jobs that are created by another user.

3. For the selected job, slide the toggle to the right in the **Started** column.

Alternatively, you can click the **Start selected** button to start the selected orchestration job.

To start multiple orchestration jobs:

1. Open the **Scheduler** user interface.
2. Click the check box next to the **Type** column to select the orchestration jobs you want to start, or click the **Select All** check box to select all the jobs.

Note: You can only start the jobs that are created by the currently logged-in user, the currently used environment, and the user's current role. The currently logged-in user cannot start the jobs created by another user.

3. For the selected orchestration jobs, click the **Start selected** button to start the jobs.

To stop orchestration jobs:

1. Open the **Scheduler** user interface.
2. Select the orchestration jobs you want to stop.

Note: You can stop all the jobs that are created by the currently logged-in user, regardless of the environment in which the user is logged in, and role the user is logged in as. The currently logged-in user cannot stop the jobs created by another user.

3. For the selected orchestration jobs, click the **Stop selected** button to stop the selected jobs.

To stop an individual orchestration job, slide the toggle to right in the Started column for selected row.

Updating to Version 3 Orchestrations

All orchestrations that you create with Orchestrator Studio 9.2.4 use version 3 AIS services and are referred to as version 3 orchestrations. Version 3 AIS services include all version 1 and 2 services available on the AIS, plus additional services that enable you to create an orchestration.

Version 3 orchestrations are compliant with the OpenAPI 2.0 and 3.0 standards. This allows the defined outputs of a version 3 orchestration to be discoverable when using an OpenAPI catalog service. In addition, outputs can now be typed with a version 3 orchestrations, whereas version 1 and 2 outputs must be strings. To comply with the OpenAPI specification, the format of orchestration output has changed in Version 3, particularly the format for arrays. Before upgrading an orchestration to Version 3, review your existing integrations that use Version 1 or Version 2 output and revise them to accept Version 3 output.

See also, "Updating Version 1 Orchestrations to Version 2 Orchestrations" in the *JD Edwards EnterpriseOne Tools Orchestrator Guide for Studio Version 8 and Prior*.

Restoring Orchestrations and Orchestration Components

Restore option 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 restore all files, click the Manage drop-down menu in the Orchestrations design page and then click the **Restore** link.

Each individual orchestration component design page also contains a Restore link in the Manage drop-down menu that you can use to restore an individual component.

Supported Input Message Formats

The Orchestrator supports two input message formats for orchestrations: a standard JD Edwards EnterpriseOne format and a generic format. The following shows an example of the code for each format.

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"
    }
  ]
}
```

Generic Input Message Format

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

```
    "temperature": "99"  
  }
```

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",  
  "equipmentDetail": {  
    "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
```

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 the following example, "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 Cross Reference 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.

```
{  
  "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"
            }
          ]
        }
      ]
    }
  ]
}
```

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.

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.

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.

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 of orchestration components, the modifying of shared orchestration components, and 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 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.

(Release 9.2.5) The Orchestrator Studio exports the source XML files in a zip file which is then saved with the name of the UDO.

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

To export orchestration components:

1. On a component design page, select Export File from the Manage drop-down menu.
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.

Exporting by Category (Release 9.2.4.2)

The Orchestrator Studio Export tool enables you to export orchestrations belonging to a specific category. The Orchestrator Studio also gives you the option to export only the orchestrations or the orchestration along with all of the components that are associated with an orchestrator category.

To export orchestrations by category:

1. On the Orchestrator Studio Home page, click the **Import/Export** icon.
2. On the Import/Export page, click the **Export by Category** tab.
3. From the Category drop-down list, select the orchestrator category associated with the orchestrations and the components that you want to export.
4. Click the Search button

The Export by Category page displays all the orchestrations and the orchestration components associated with the selected category in the grid. The grid displays the Type, Name, and Description of the components.

5. Slide the Include Dependencies toggle to the right to export the orchestrations along with all the dependent components.
6. Click the Export button to export all the components displayed in the grid.

The Orchestrator Studio checks for the orchestration components against the selected category and displays the number of files to be exported. In some cases, the file count can exceed the number of files identified in the export because the Orchestrator Studio looks for components that are not in the category but are dependencies of the components that have categories.

The Orchestrator Studio saves all the components as a zip file to a location on your local machine. The zip file is saved with the name of the category. Any special character in the category name is replaced with an underscore (_).

Note: Because new features may be added to new releases of Orchestrator, exporting objects from a newer release and importing them into an earlier release (backward compatibility) is not supported. Importing objects from an earlier release into a newer release or into the same release is supported.

Importing Orchestration Files in the Orchestrator Studio

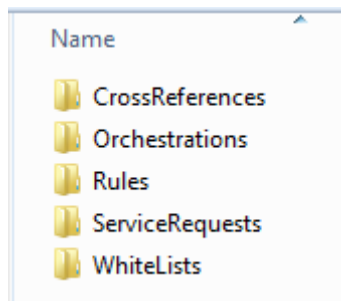
CAUTION: Do not use the Export and Import tools to share orchestration component files with other users. With orchestration components stored as UDOs in EnterpriseOne, the sharing of orchestration components, and the modifying of shared orchestration components, as well as the promoting of orchestration 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 and zip files containing XML files. You can select multiple files to be imported at a time.

CAUTION: You cannot import orchestration component files that were exported from the EnterpriseOne Object Management Workbench - Web application. The zip file that is 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. You can import it as a new UDO with a status of "Personal."

To import files:

1. On the Orchestrator Studio Home page, click the **Import/Export** icon.
2. On the Import/Export page, click the **Import** tab. On the Import tab, click the **Upload** icon.
3. Locate the orchestration component XML files that you want to import.

With Tools release 9.2.4.3, you can select multiple orchestration component XML files and multiple compressed files containing orchestration XML files that you want to import. When the upload list contains multiple instances of the same UDO, only the last instance of the UDO is imported.

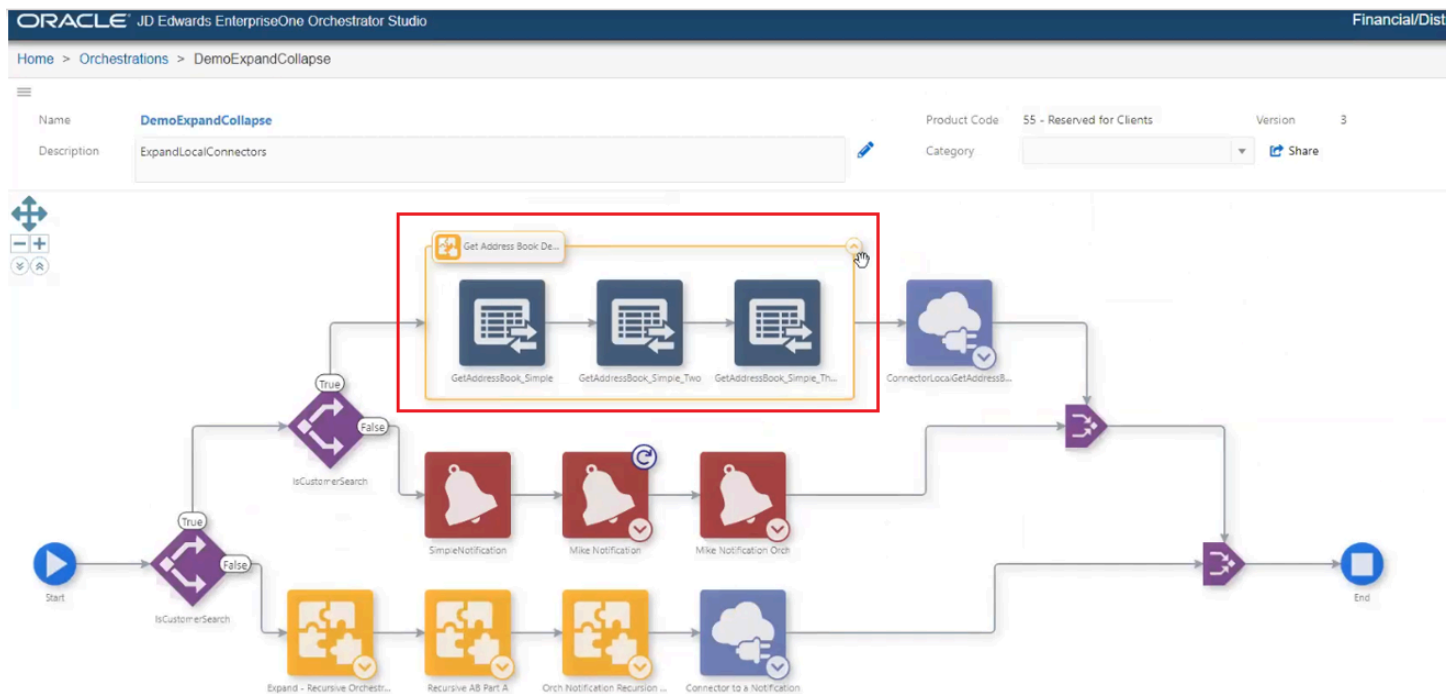
After you select the files to import, the Import tool checks the XML files that you selected against the current UDOs in the Orchestrator Studio and displays the options that are available for importing the files.

4. Review and select the appropriate 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 that of 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, do not select the Import check box. Next, change the status of the current component to "Reserved" and reimport the XML file to overwrite the component.

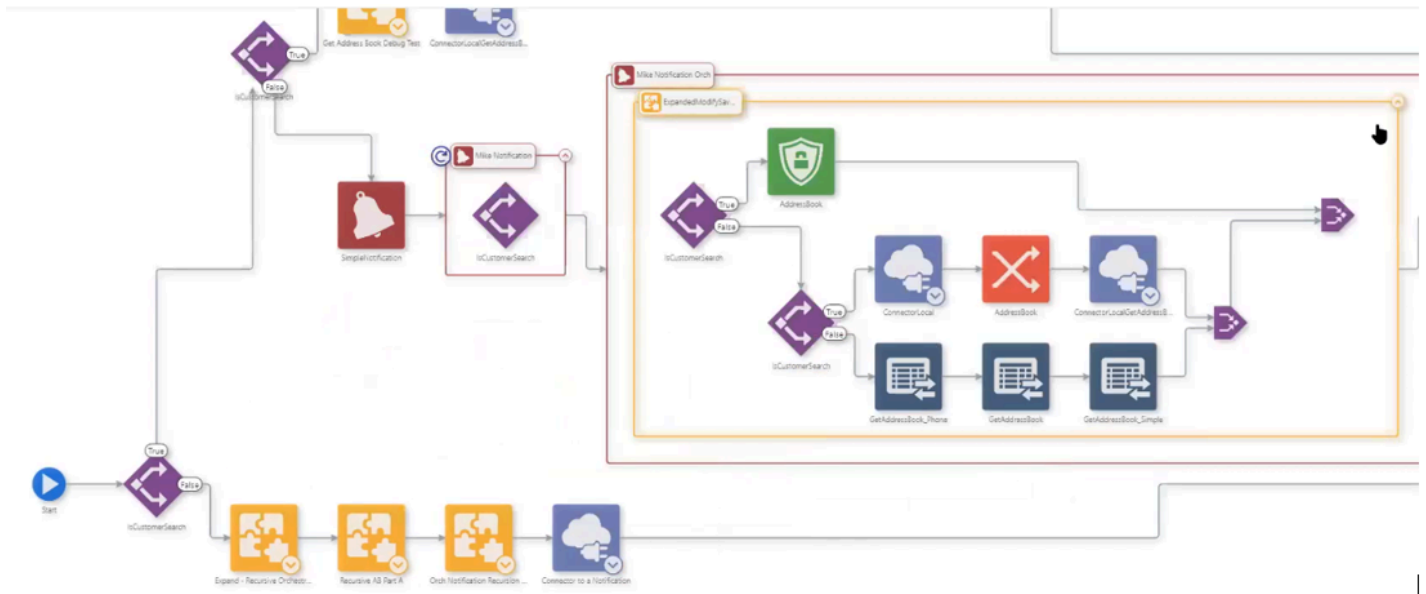
- o File already exists.
The XML file matches a component that is in a "Personal" or "Reserved" status. Click the check box to import file and overwrite the current component.
 - o File is ready to submit.
The XML file is unique and does not already exist in the Orchestrator Studio. Click the check box to import it.
5. You can also click the **Select All** check box to import all the XML files that are available for importing.
 6. Click the **Submit** button.
The Orchestrator Studio imports the selected components into the components list on the respective design page.

Expanding and Collapsing the Nested Orchestration Steps (Release 9.2.4.4)

When an orchestration is included as a step within another orchestration, you can click the **Expand** icon to expand the included orchestrations inline. When you click the **Expand** icon, the element representing the orchestration is replaced by a bounding box labeled with the name of the orchestration and all the steps for the now-expanded orchestration are displayed within this box. This representation enables you to see what the sub-orchestrations accomplish without navigating away from the parent orchestration.



You can expand the notifications that contain a rule, an orchestration, or both in a similar manner and see the child elements.



Note: Notifications that do not include a rule or an orchestration will not have an Expand icon.

Similarly, a connector to an orchestration and a connector to a notification can be expanded.

It is possible to expand multiple layers of elements. For example, if an orchestration contains a connector to a notification that calls an orchestration, you can first expand the connector, then expand the notification, and finally expand the orchestration. You can see three nested bounding boxes representing the logical relationship between these elements.

Similarly, if a child orchestration calls another orchestration, you can expand the child orchestration and then expand the "grandchild" orchestration if required.

If an orchestration calls itself in a recursive scenario, you can expand the orchestration within itself.

Note: You will see only the saved state of the orchestration in the expanded view.

Modifications can be made to the parent orchestration while the child steps are in the expanded mode; however, all the expanded steps will revert to the collapsed mode when you save the changes. For more information, see [Editing the Nested Orchestration Steps \(Release 9.2.4.4\)](#).

You can also use the **Expand All Orchestration Steps** and **Collapse All Orchestration Steps** icons to expand and collapse all the steps.

Editing the Nested Orchestration Steps (Release 9.2.4.4)

You can double-click the collapsed orchestration step or double-click the white space within the bounding box of an expanded orchestration to edit the child orchestration. Alternatively, you can click the orchestration step and then click the **Pencil** icon to open the step for editing. This will cause the editor to navigate away from the parent orchestration to edit the child orchestration instead, and triggers a prompt to save changes if applicable.

If you want to review or edit a non-orchestration element (for example, a rule or a form service request) within the child orchestration, double-click the element. The system displays a modal window for editing without navigating away from the non-orchestration element.

Note: You must save the child orchestrations after you edit them. When you attempt to navigate away without saving the changes, the system displays the message that you will lose your unsaved changes if you proceed. If you are editing a step associated with a recursive scenario, and simultaneously try to open the same orchestration, the system declines your request and displays an explanatory message. You will not see the **Pencil** icon for editing if you are more than one layer deep in the orchestration. However, you can double-click the step and open it for editing.

Printing an Orchestration (Release 9.2.4.4)

You can use the print functionality to capture your orchestrations on a standard letter page (8.5 x 11 inches) or as a PDF file. The printed document will include a summary page with the graphical representation of the orchestration flow and supplemental pages with details regarding orchestration inputs, variables, outputs, scheduling, and so on. The supplemental pages also provide details on each orchestration element, including transformations, and error handling.

The printed orchestration document can be used for comprehensive auditing, for communicating orchestration function through an offline medium, and for recreating the orchestration from a human-readable specification.

To print an orchestration:

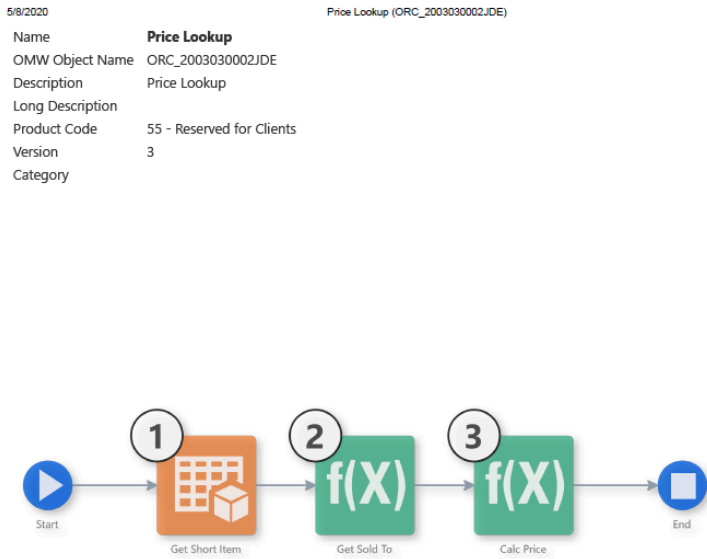
1. On the Orchestrator Studio Home page, click the **Orchestrations** icon.
2. Using the Orchestrations side panel, select the orchestration you want to print.
3. Open the Manage menu and click **Print**. You can also use the Control+P shortcut key.
4. On the Print window, verify that you have selected the option to print background graphics.

Note: The orchestration diagram will not be displayed properly if you do not select this option.

5. From the Destination drop-down menu, select Save as PDF or your printer name as required.
6. Click the **Save** button. The value in the File Name field on the Save As window is specific to the browser. For example, the file is saved with the name of the orchestration by default if you are using Google Chrome.

The first page contains the name, description, long description, product code, version, and category details. The first page also displays the diagram of the orchestration as shown in this example.

The second page contains details about Orchestration Inputs, Example Inputs, Variables, Schedule, and Orchestration Outputs if available. Each orchestration step is labeled with a number to facilitate finding the corresponding step details provided in the subsequent pages.



5/8/2020 Price Lookup (ORC_2003030002.JDE)

Orchestration Inputs

Input	Value Type	Default Value	Required
SoldTo	Numeric		
2nd Item Number	String		
mnQtyOrdered	Numeric	1	
Branch	String		
Company	String		

Example Input

```

    {
      "SoldTo": 0,
      "2nd Item Number": "string",
      "mnQtyOrdered": 0,
      "Branch": "string",
      "Company": "string"
    }
  
```

Variables
None

Schedule
None

Orchestration Outputs

Field	Output Name	Value Type	Selected
Orchestration Inputs			
SoldTo			
2nd Item Number			
mnQtyOrdered			
Branch			
Company			
Get Short Item			
Data Browser - F4101 (Item Master)			
Short Item No-F4101.ITM	Short Item No	Numeric	✓
Ln Ty-F4101.LNTY	Ln Ty	String	✓
Get Sold To			
szAdjustmentSchedule	szAdjustmentSchedule	String	✓
szCustomerPricingGroup	szCustomerPricingGroup	String	✓
Calc Price			
mnUnitPrice	mnUnitPrice	Numeric	✓
mnExtendedPrice	mnExtendedPrice	Numeric	✓
mnListPrice	mnListPrice	Numeric	✓
szListPriceUOM	szListPriceUOM	String	✓
jdTransactionDate	jdTransactionDate	ISO Date	✓
szTransactionUom	szTransactionUom	String	✓
szPricingUom	szPricingUom	String	✓
jdPriceEffectiveDate	jdPriceEffectiveDate	ISO Date	✓

Monitoring Orchestrations (Release 9.2.7.4)

Starting with Tools Release 9.2.7.4, you can enable Orchestration Monitoring by configuring it from the Run Options page. You can configure to monitor just orchestration details or also include step details. This enables you to effectively control the orchestrations that are logged for success and failure, along with additional logged data and visualizations in the Orchestrator Monitor.

Configuring Orchestration Monitoring

To set up monitoring:

1. Open the orchestration in Orchestrator Studio.
2. On the Orchestrations design page, click the **Start** node.
The action control icons are displayed above the Start node.
3. Click **Run Options**. The system displays the Run Options window.
4. In the Monitoring section, select these options as required:

- **Orchestrator Details:** Records the start time, end time, duration, status, and assertion count for the orchestration.
- **Orchestration Input/Output:** Records the requested input and response output of the orchestration.
- **Step Details:** Records the start time, end time, duration, and status of each orchestration step.
- **Step Input/Output:** Records the input and output of each orchestration step.

CAUTION: Enabling these options may generate a very high volume of records for processes that run very frequently or have many steps. It is recommended to leave these options disabled unless there is a specific need for monitoring. It is also recommended to review the Orchestrator Exceptions program (P980060) periodically and purge unnecessary records. See *Managing Orchestrator Health and Exception Records in EnterpriseOne*.

5. Enable the **Monitor on Start** option if required.

CAUTION: It is recommended to enable this option only for long-running processes. This setting starts logging at the beginning of the process and incurs additional overhead.

6. Enable the **All Environments** and **All Users** options to monitor all the environments and users. Disable these options to verify or add specific environment and user details.

You can enter specific environments and users in the Environment to Monitor and User/Role to Monitor tables. Click the **X** icon in the table to delete a row.

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

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.

Work with Business Service Cross Reference

✓ 🔍 + 🗑️ ✕ 🔄 Form ⚙️ Tools

Cross Reference Types KEY CODE AIS All

Cross Reference Object Type

Records 1 - 10 > >

	Cross Reference Type	Cross Reference Object Type	Third Party App ID	Third Party Value	EOne Value
<input checked="" type="radio"/>	AIS	ABMASTER	ABFilter	25	<25
<input type="radio"/>	AIS	ABMASTER	ABFilter	50	<50
<input type="radio"/>	AIS	ABMASTER	WHITELIST	OpenScript C	OpenScript C

- 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					
✓ ✖ ✕ ↶ Form ⚙️ Tools					
Records 1 - 2					
	Cross Reference Type	Cross Reference Object Type	Third Party App ID	Third Party Value	EOne Value
<input checked="" type="radio"/>	AIS	ADDRESS	WHITELIST	1234	NA
<input type="radio"/>					

- 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					
✓ ✖ ✕ ↶ Form ⚙️ Tools					
Records 1 - 2					
	Cross Reference Type	Cross Reference Object Type	Third Party App ID	Third Party Value	EOne Value
<input checked="" type="radio"/>	AIS	ITEM_BRANCH	TPApp	A-293-B2	200 M30
<input type="radio"/>					

- 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.

7 Orchestrator Health and Exception Monitoring

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:

- Orchestrations
- Notifications
- Schedules
- AIS REST APIs

Starting with Tools Release 9.2.7.3, you can view exceptions for AIS REST API requests in Orchestrator Monitor. All the REST APIs that are exposed on the AIS Server will be monitored for exceptions. Exceptions are logged in the exception table and they are displayed in Orchestrator Monitor.

For schedules and REST APIs, 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* .

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*.

Accessing the Orchestrator Monitor

You can access the Orchestrator Monitor from EnterpriseOne or from the Orchestrator Studio.

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.

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**.

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.

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.

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.

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.

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



or a  REST Service.

- 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 in seconds 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.

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.

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* .

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**, **Schedule**, or **REST**. 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.

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

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.

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 Response
Click this button to view the exception response.
- Message (Release 9.2.6)
Click this button to view the **custom message** configured in Orchestrator Studio.
- 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.

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.

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. The following image shows an example of exceptions displayed in a vertical bar chart.



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.

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.

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.

Monitoring Orchestrator Run Details (Release 9.2.7.4)

Orchestrator Monitor gives you visibility into the overall health of your orchestrations, notifications, and AIS REST services as well as details about any exceptions. However, depending on the nature and frequency, some orchestrations and notifications might require more fine-grained monitoring as well as a more comprehensive record of all successful transactions.

When monitoring is enabled, the records generated for each execution are displayed in the Run Details tab of the Orchestrator Monitor.

To know more about how to set up monitoring for orchestrations, see [Configuring Orchestration Monitoring](#).

To know more about how to set up monitoring for notifications, see [Monitoring Notifications \(Release 9.2.3\)](#).

Understanding the Run Details Tab

The Run Details tab in Orchestrator Monitor displays monitoring details such as:

- Notification input and output, subscriber details, subscriber input and output, and so on for notifications that are enabled for monitoring.
- Orchestration input and output and step details for orchestrations that are enabled for monitoring.
- REST service details.

The default view in the Run Details tab of the Orchestrator Monitor displays high-level information about the name, duration, status and so on. Expand each row to find additional run details.

High-Level Run Details Information

The Run Details tab displays the following high-level information:

- Request Received

This is the date and time when the monitoring request was received.

- 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



or a  REST service.

- Status

Displays the status as Running, Queued, Successful, Failed, Warning, or Successful Some Steps Failed.

- Duration(ms)

This is the run time duration of the UDO displayed in milliseconds.

- User

This is the name for the user.

- Environment

The environment in which the UDO was processed.

- Product Code

The product code associated with the UDO.

Detailed Monitoring Information

Additional details may be displayed based on the settings in the Monitoring section of the Run Options window. Expand a row to view the following additional details:

Note: When you expand a row with the Queued status, you can click the Refresh icon to see the progress and review the changes as and when the process is running.

- UDO Name

This is the ID of the UDO in EnterpriseOne.

- Host Address

This is the IP address of the host machine.

- Host Name

This is the name of the host machine.

- Status

Displays the status as Running, Queued, Successful, Failed, Warning, or Successful Some Steps Failed. Click the **More Details** button next to the Status field, to view the input and output details in the JSON format.

- URL

- Type

This is the UDO type. The system displays if the UDO is an orchestration, notification, or a REST service.

- Duration

This is the run time duration of the UDO displayed in seconds.

- Run Id

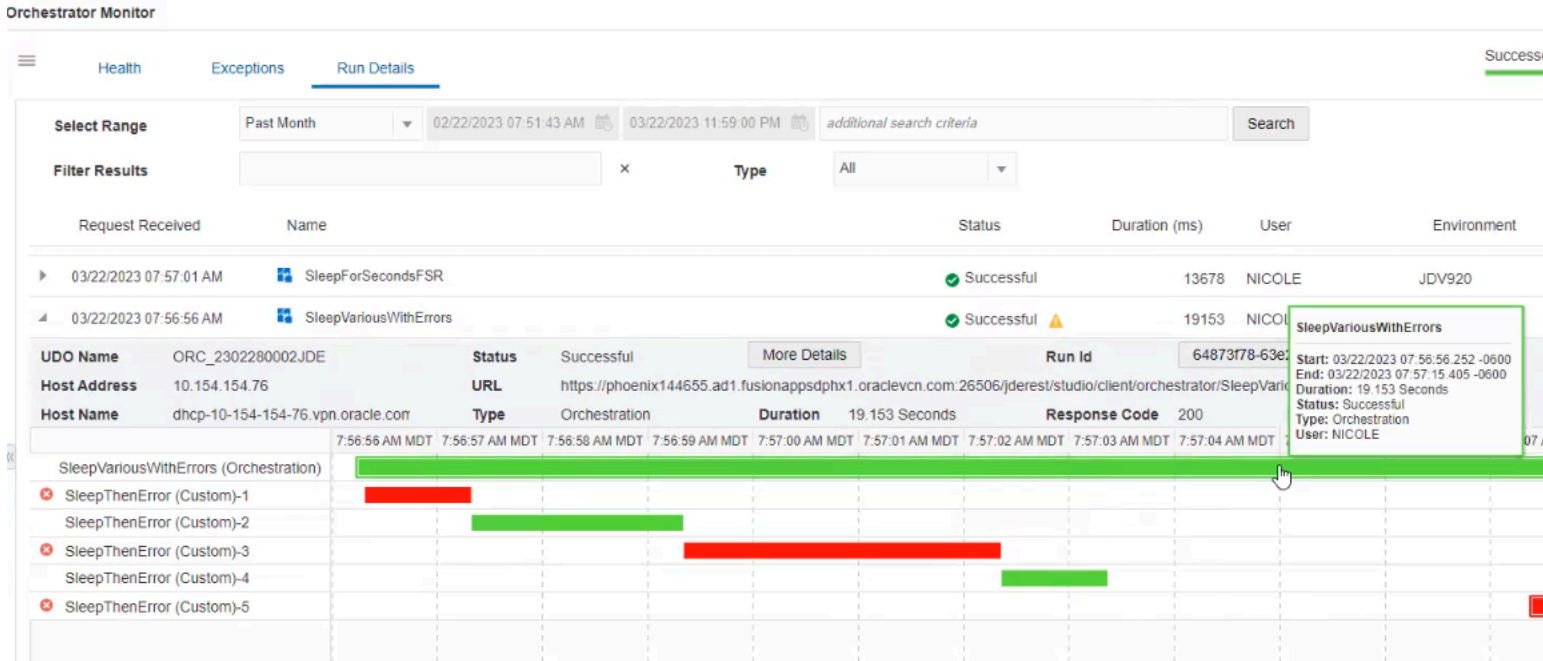
- Response Code

This is the HTTP response code to indicate how server responded to the client's request.

- Queue

The system displays a bar chart with details for each step if you are tracking the steps of the orchestration. The steps that are processed successfully are displayed in green and the failed steps are displayed in red. You can hover over the bar chart to view more information about a particular step.

Click the + and - icons to zoom in or zoom out the longer steps. Hover over each step to review more details.



If the Step Input/Output option is selected for monitoring an orchestration, the system records the input and output of each orchestration step. When you click the individual bar, the system displays a window with input and output values in the JSON format.

Triggering Monitoring

You can configure monitoring for Orchestrations and Notifications at design time in Orchestrator Studio. See the following pages for details:

- [Configuring Orchestration Monitoring.](#)
- [Monitoring Notifications \(Release 9.2.3\).](#)

Monitoring can also be triggered by sending headers as part of the notification, orchestration, or REST request.

For the case of REST services, headers are the only way to trigger monitoring.

If the request contains the following headers, the passed value overrides the design time configuration when calling an orchestration or a notification.

- `jde-AIS-MonitorRequest` : This header is used to trigger monitoring. The following values allow you to control the level of monitoring:

```
monitorRequest , monitorRequestIO,
monitorRequestAndSteps , monitorRequestIOAndSteps ,
monitorRequestAndStepsIO , monitorRequestIOAndStepsIO
```

Note: The values containing `steps` are only valid for orchestration and notification requests.

- `jde-AIS-MonitorOnStart` : This header takes a value of true or false and only applies if monitoring is triggered.

Filtering the Run Details

To change the search criteria in the Run Details tab:

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**, **Schedule**, or **REST**. 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. For example, you can enter the name of a UDO, status code, user, or other values displayed in the Run Details header. The filter criteria is not case sensitive.

Managing Orchestrator Health and Exception Records in EnterpriseOne

Note: 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

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 in this guide for download instructions.

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:
 - o **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.
 - o **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.
 - o **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**.

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.

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:
 - o **Object Type**
To find records based on a particular UDO type, enter `ORCH`, `NTF` (for notifications), or `SCHEDULE`.
 - o **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.
 - o **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**.

8 Creating Orchestrations for System Administration with Orchestrator Studio

Understanding the Orchestrations for System Administration

The JD Edwards EnterpriseOne Server Manager is REST enabled to perform various provisioning, monitoring, and management activities. REST enablement in Server Manager exposes these functionalities as REST services. The JD Edwards EnterpriseOne Orchestrator Studio can be used for JD Edwards System Administration. System administrators can call the Server Manager REST services from the Orchestrator Studio to automate the administrative tasks.

An orchestration for system administration can invoke the REST services on the Server Manager. This functionality enables the system administrators to leverage various orchestrator features such as invoking chained orchestrations, defining conditions, rules and logic, and sending messages and notifications. The orchestration for system administration simplifies the automation of various administration tasks.

The documentation for the Server Manager Console can be accessed from: https://<server_manager_console_host:port>/manage/mgmtrestservice/v1/open-api-catalog

The components available on the Server Manager Console are present at: `<Server_Manger_Console_installed_path>/SCFMC/components`.

You can update the components using the FTP connection. You can configure a connector service request to transfer files by using FTP. With the FTP connector, you can transfer the component to the Server Manager Console. See, [Configuring a Connector to Transfer Files Using File Transfer Protocol \(FTP\)](#).

The orchestrations for system administration enable the system administrator to perform the following:

- Execute Server Manager tasks (APIs) from the JD Edwards Orchestrator Studio.
- Create, configure, and send notifications for system administration tasks through the orchestrator.
- Schedule the orchestrations created using the REST services.

Creating an orchestration for system administration involves:

1. Creating a connection to the Server Manager
2. Creating the connector service request
3. Adding the service request to the orchestration

Creating a Connection to the Server Manager

You can create an Open API connection that lists all the REST services (API calls) that can be invoked through the Server Manager Console.

You can add this connection to the service request that uses the information that is retrieved from the Open API connection. You can then create an orchestration that invokes the Server Manager REST services.

A connector service request requires a connection that provides access to the Server Manager Console, In this case the server where the REST service resides.

For detailed information on adding the Open API connection, see [Creating an Open API Connection](#).

Creating the Connector Service Request

You can create an Open API connector service request to call a REST service. For detailed information on adding the Open API connector service request, see [Configuring an Open API Connector to Invoke a REST Service](#).

Adding the Service Request to an Orchestration

You can create an orchestration and then add service requests to the orchestration, and map the orchestration input to the input defined in the service request.

Creating an orchestration in the Orchestrator Studio involves:

1. Naming the orchestration and specifying the input format for the orchestration.
2. Adding inputs to the orchestration.
3. Adding steps to the orchestration.
4. Configuring transformations.

For detailed information about creating an orchestration, see [Creating Orchestrations](#).

9 Creating Custom Java for Orchestrations

Understanding Custom Java for Orchestrations

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.

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 11.0 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.e1.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.e1.aisclient.LoginEnvironment` and a `process()` method that takes no parameters and returns a `javax.ws.rs.core.Response`.

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.

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.

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*

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>
```

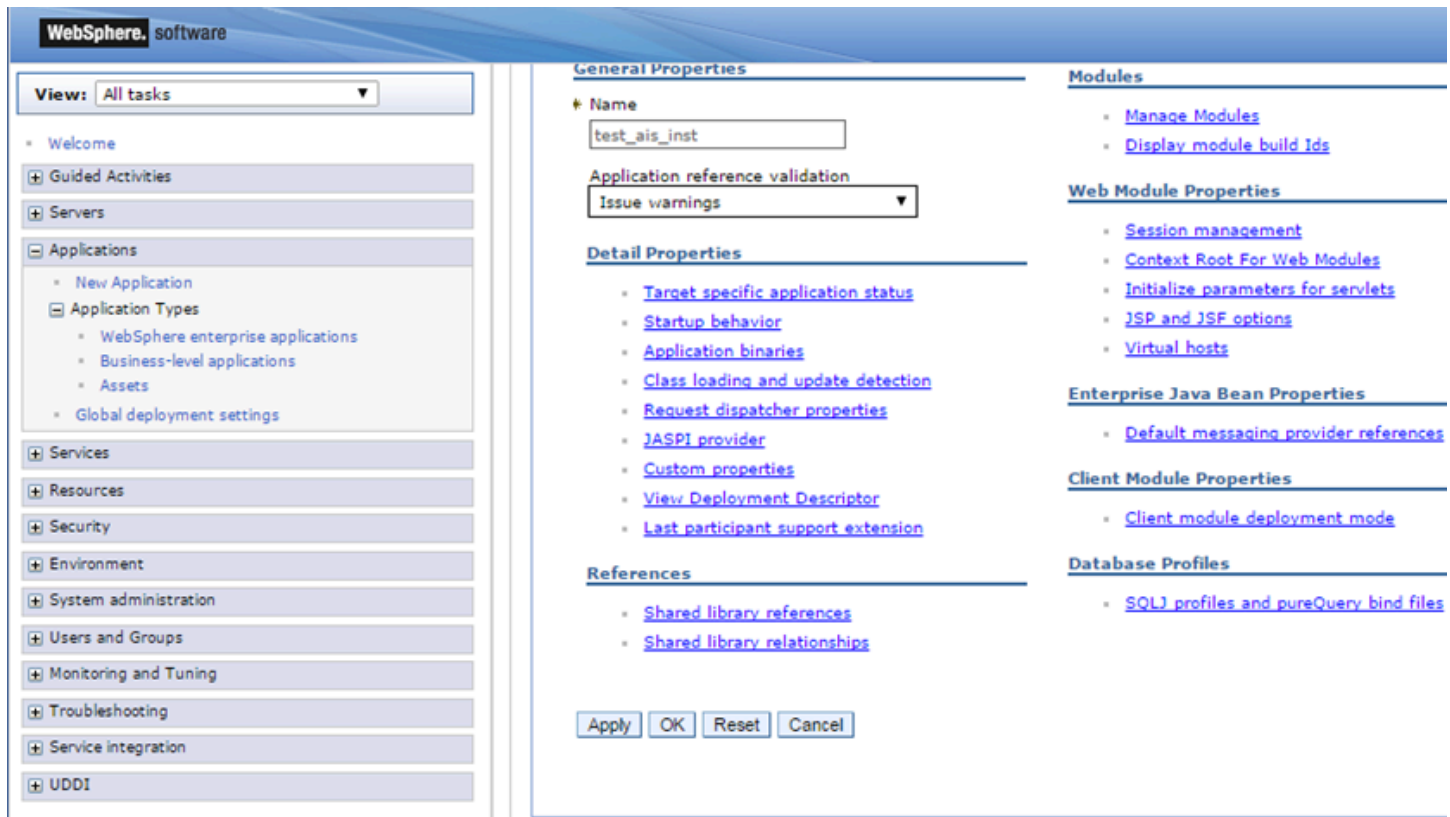
4. Redeploy JDERestProxy from Server Manager.

Deploying Custom Java on AIS Server on IBM WebSphere Application Server

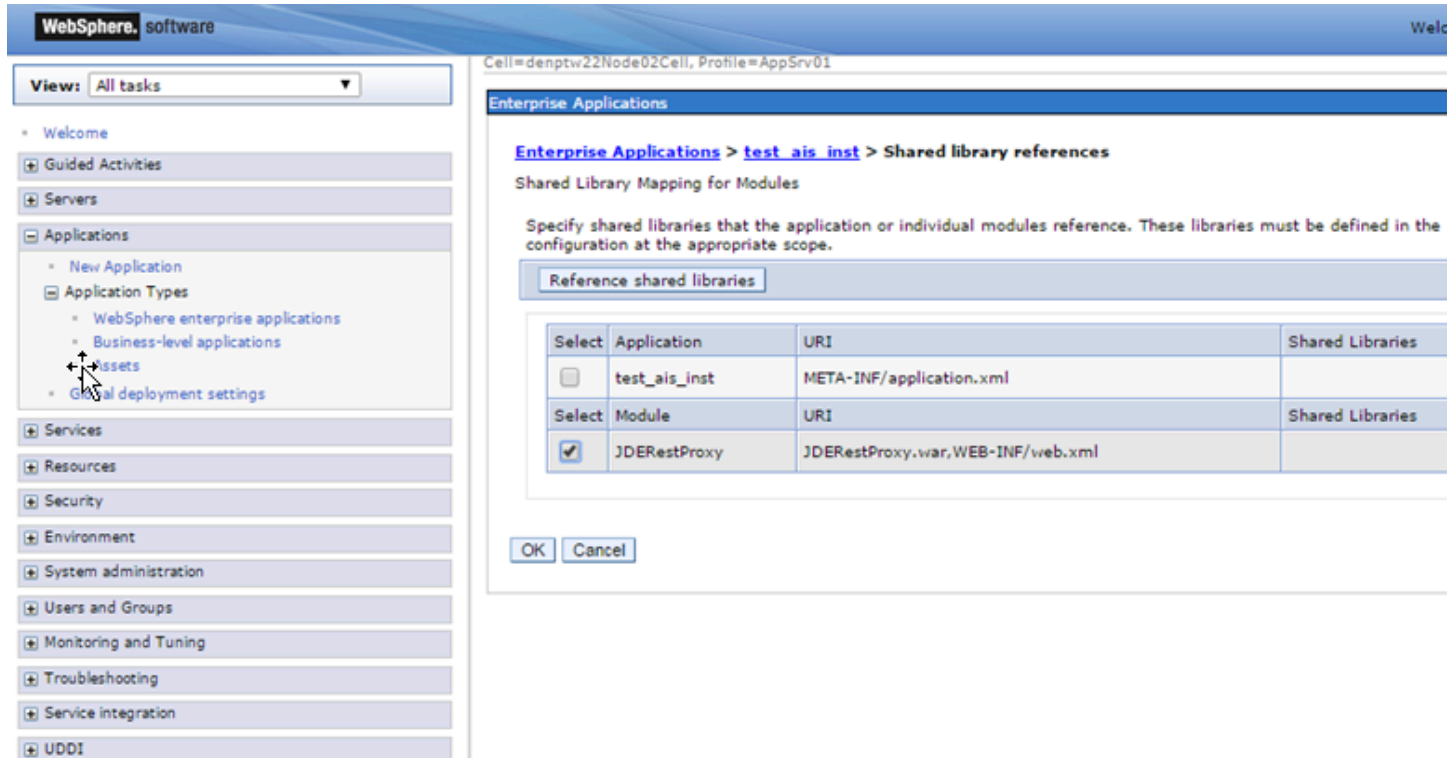
To deploy the custom Java JAR file to the AIS Server on Websphere Application Server:

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.

10 Using Scripting Languages for Custom Service Requests, Rules, and Manipulating Output

Using Scripting Languages

In the Orchestrator Studio, programmers can use the Apache Groovy, JRuby, and Jython scripting languages to extend the functionality of orchestrations. In the Orchestrator Studio, you can use these scripting languages 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.
- Read and write data in a database using a database connector.

Understanding Groovy for Orchestration Components

In the Orchestrator Studio, programmers can use the Apache Groovy, JRuby, and Jython scripting languages, to extend the functionality of orchestrations.

Using scripting languages has two advantages over using custom Java. First, the Orchestrator Studio provides an editing window for creating the script, so you do not have to use an external editor to create and test your custom program. The editing window contains a scripting template that you can use to help get you started. Second, the Orchestrator includes the script natively as part of the orchestration, so you do not have to deploy a custom program like you do with custom Java. The script simply executes as part of the orchestration.

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.

```
1  { 1 import groovy.transform.CompileStatic;
2  { 2 import com.oracle.e1.common.OrchestrationAttributes;
3  { 3 import java.text.SimpleDateFormat;
4  { 4 @CompileStatic
5  { 5 HashMap<String, Object> main(OrchestrationAttributes orchAttr, HashMap inputMap)
6  { 6 {
7  { 7     HashMap<String, Object> returnMap = new HashMap<String, Object>();
8  { 8     // Add logic here
9  { 9     // String stringVal = (String)inputMap.get("inputStringVal");
10 { 10    // BigDecimal numVal = new BigDecimal((String)inputMap.get("inputNumVal"));
11 { 11    // SimpleDateFormat format = new SimpleDateFormat(orchAttr.getSimpleDateFormat());
12 { 12    // Date dateVal = format.parse((String)inputMap.get("inputDateVal"));
13 { 13
14 { 14    // orchAttr.writeWarn("custom log entry - warning");
15 { 15    // orchAttr.writeDebug("custom log entry - debug");
16 { 16
17 { 17    // returnMap.put("Limit Status", (String)inputMap.get("inputStringVal"));
18 { 18
19 { 19    return returnMap;
20 { 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:
 - o Parameters for defining a string, numeric value, date format, and date value.
 - o Parameters for including a warning or log statements in the AIS Server log, which can be used for debugging script issues.
 - o 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](#).

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:

```
1  { 1 import groovy.transform.CompileStatic;
2  { 2 import com.oracle.e1.common.OrchestrationAttributes;
3  { 3 import java.text.SimpleDateFormat;
4  { 4 @CompileStatic
5  { 5 Boolean main(OrchestrationAttributes orchAttr, HashMap inputMap)
6  { 6 {
7  { 7 Boolean result=false;
8  { 8 // Add logic for the rule here
9  { 9 // String stringVal = (String)inputMap.get("inputStringVal");
10 { 10 // BigDecimal numVal = new BigDecimal((String)inputMap.get("inputNumVal"));
11 { 11 // SimpleDateFormat format = new SimpleDateFormat(orchAttr.getSimpleDateFormat());
12 { 12 // Date dateVal = format.parse((String)inputMap.get("inputDateVal"));
13 { 13
14 { 14 // orchAttr.writeWarn("custom log entry - warning");
15 { 15 // orchAttr.writeDebug("custom log entry - debug");
16 { 16
17 { 17 return result;
18 { 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](#).

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.

```
1  { 1 import groovy.transform.CompileStatic;
   2 import groovy.json.JsonSlurper;
   3 import groovy.json.JsonBuilder;
   4 import com.oracle.e1.common.OrchestrationAttributes;
2  { 5 @CompileStatic
   6 String main(OrchestrationAttributes orchAttr, String input)
   7 {
   8     def jsonIn = new JsonSlurper().parseText(input);
   9     // modify jsonIn
  10     def jsonOut = new JsonBuilder(jsonIn).toString();
3  { 11 // orchAttr.writeWarn("custom log entry - warning");
   12 // orchAttr.writeDebug("custom log entry - debug");
2  { 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, 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](#).

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.

This image shows the Groovy template that you can use to manipulate orchestration output:


```

1  {
    1  import groovy.transform.CompileStatic;
    2  import groovy.json.JsonSlurper;
    3  import groovy.json.JsonBuilder;
    4  import com.oracle.e1.common.OrchestrationAttributes;
2  {
    5  @CompileStatic
    6  String main(OrchestrationAttributes orchAttr, String input)
    7  {
    8      def jsonIn = new JsonSlurper().parseText(input);
    9      // modify jsonIn
   10      def jsonOut = new JsonBuilder(jsonIn).toString();
3  {
   11      // orchAttr.writeWarn("custom log entry - warning");
   12      // orchAttr.writeDebug("custom log entry - debug");
2  {
   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](#).

Defining the Input and Output Variables

The Orchestration Outputs design page provides a script template that you can use. The references to values in the code are related to the inputs and outputs defined in the table below the script.

In the following image of the Groovy template sample, the:

- Blue mapping indicates that for every value you expect to get from the `inputMap` in the code, a matching entry must exist in the **Input** column of the custom step.
- Green mapping indicates that for every value you expect to get from the `returnMap` in the code, a matching entry must exist in the **Output** column of the custom step.
- Values highlighted in yellow are the variable names for either an individual output or an array output. These names are available for mapping in transformations of the subsequent steps.

Name **GroovyExample**
Product Code

Description GroovyExample
Category

Groovy

```

3 HashMap<String, Object> main(OrchestrationAttributes orchAttr, HashMap inputMap)
4 {
5   HashMap<String, Object> returnMap = new HashMap<String, Object>();
6   // Add logic here
7   String stringVal = (String)inputMap.get("inputStringVal");
8   BigDecimal numVal = new BigDecimal((String)inputMap.get("inputNumVal"));
9   SimpleDateFormat format = new SimpleDateFormat(orchAttr.getSimpleDateFormat());
10  Date dateVal = format.parse((String)inputMap.get("inputDateVal"));
11
12  orchAttr.writeWarn("custom log entry - warning");
13  orchAttr.writeDebug("custom log entry - debug");
14
15  def array = '[{"Column1":"val1","Column2":"val2"}, {"Column1":"val3","Column2":"val4"}]';
16  returnMap.put("Output Array", array);
17  returnMap.put("Output String", stringVal);
18  returnMap.put("Output Date", dateVal);
19
20  return returnMap;
                
```

Load Outputs
Test

Input	Test Value	Output	Variable Name	Test Output	Member Name
inputStringVal	abc	Output String	OutputStringVar		Column1
inputNumVal	123	Output Date	OutputDateVar		Column2
inputDateVal	10/12/2021				

Variable Names For Transformations in Subsequer

Additional Attributes and Methods Available in the Groovy Script Templates

All scripts are passed an `OrchestrationAttributes` instance called `orchAttr` that contains additional information that can be used in the script. This table provides a description of the attributes.

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.

Attribute	Type	Description
dateFormat	String	The date format of the execution user.
dateSeperator	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.
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. (Release 9.2.4.0+ The user session temp directory, a dynamic sub directory of the one configured in the rest.ini.)

This table describes the methods available in the `OrchestrationAttributes` class.

Method	Response Type	Description
<code>writeWarn(String message)</code>	String	Write a message as a warning to the log.
<code>writeDebug(String message)</code>	String	Write a message as a debug to the log.
<code>writeWarn(String message, Exception e)</code>	String	Write a message as a warning to the log with an exception.
<code>writeDebug(String message, Exception e)</code>	String	Write a message as a debug to the log with an exception.
<code>getOrchestrationName()</code>	String	Get the name of the currently running orchestration.

Method	Response Type	Description
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.
getDateSeperator()	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. (Release 9.2.4.0+ returns the user session temp directory)
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. (Release 9.2.4.0+ returns file name in the user session temp 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.

Method	Response Type	Description
		(Release 9.2.4.0+ returns a unique file name in the user session temp directory)
toString() (Orchestrator Studio 6.1.0)	String	Outputs a JSON string representing all values in this class.
getTempRootFileName(String fileName) (Release 9.2.6.2)	String	Build a fully qualified file name in the root temp directory configured in the rest.ini. The fileName parameter is the name of the file to be appended to the directory.
getMachineKey() (Orchestrator Studio 9.2.4)	String	Get the machine key (first 10 characters) of the AIS server. (Release 9.2.4.2) The maximum length for this field is ten characters. If the string is longer than ten characters, it will be truncated.
getMachineName() (Orchestrator Studio 9.2.4)	String	Get the machine name of the AIS server. (Release 9.2.4.2)
getLongUserId () (Orchestrator Studio 9.2.4)	String	Get the long User Id for the currently logged in user. (Release 9.2.4.2)
mapResultSetToDataSet(ResultSet resultSet, String dataSetName) (Orchestrator Studio 9.2.5.4)	HashMap<String, Object>	Used in Database Connector scripts. This helper method allows users to easily add the SQL rowset data to the output map of the connector. It returns a map with the dataset populated.
touchSession() (Orchestrator Studio 9.2.6.4)	void	Used for long running scripts that do not actively use the AIS token. This method allows callers to manually keep the session alive during the runtime of the script so that subsequent orchestration steps that use the session will not fail (so that the session will not timeout).
mapResultSetToDataSetAsStrings(ResultSet resultSet, String dataSetName)(Orchestrator Studio 9.2.7.0)	HashMap<String, Object>	Used in Database Connector scripts. This helper method allows users to easily add the SQL rowset data to the output map of the connector. It returns a map with the dataset populated with all the values included as string values (toString method called for each object returned).

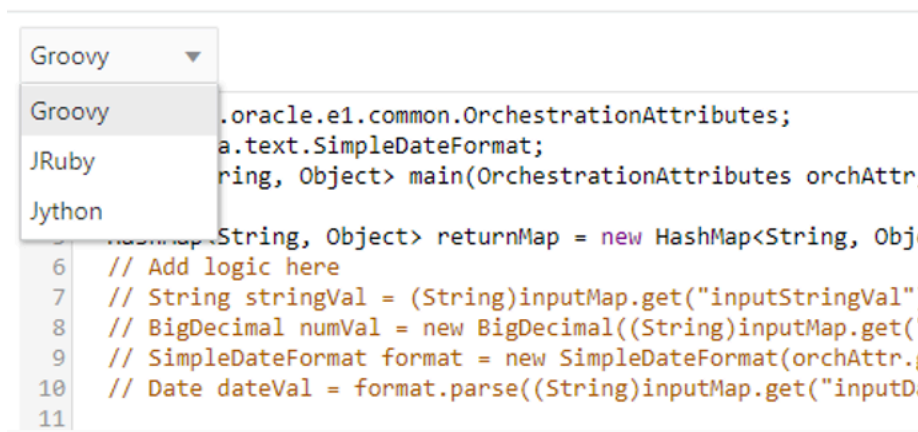
Installing Optional Scripting Languages on the AIS Server (Release 9.2.5.4)

Starting with Tools Release 9.2.5.4, orchestrations support **JRuby** and **Jython** as additional scripting languages along with Groovy.

In the Orchestrator Studio, you can use JRuby and Jython scripts for any processes that support Groovy scripts. For example, you can use the JRuby and Jython scripts in the custom service requests, rules, connectors (REST and Database), and while manipulating the orchestration output.

To enable the use of additional scripting languages for an AIS Server instance, you must provide an associated JAR file and perform the configuration steps.

After you complete the required configuration, you can see the languages listed in the drop-down menu in the script section in the Orchestrator Studio. See *Enabling JRuby* and *Enabling Jython*.



Enabling JRuby

Download the JRuby JAR file from the website at this URL: <https://www.jruby.org>.

This website also provides information on integration between Ruby and Java.

The `jruby-complete-9.2.14.0.jar` file is used as an example in this section.

Deploying JRuby on AIS Server on Oracle WebLogic Server

Depending on your requirements, choose one of these options for deploying the JRuby JAR file on the WebLogic Server:

- Shared Library Method
- AIS Instance Classpath Method
- Domain Library Method

Using the Shared Library Method

1. Deploy the JRuby JAR file as a shared library on the same WebLogic Server on which the AIS Server (otherwise referred to as the JDERestProxy) is deployed. Ensure that the AIS Server is included as a target for the library.

2. Before uploading the AIS server component to the Server Manager console, edit the `weblogic.xml` in the deployment package.
 - a. Open the AIS server component (par or jar) in a zip file editor and navigate to the `weblogic.xml` file as shown in this example: `\E1_AISServer_9.2.5.4_mm-dd-yyyy_hh_mm.jar\JDERestProxy.ear\JDERestProxy.war\WEB-INF\weblogic.xml`
 - b. Edit the `weblogic.xml` to include the shared library by the name it was created with in WebLogic, as illustrated in the following 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>JRubyLibrary</library-name>
  </library-ref>
</weblogic-web-app>
```

- c. Save the `weblogic.xml` file and re-zip the AIS component.

3. Upload and then deploy the component from Server Manager.

Using the AIS Instance Classpath Method

1. Save the JRuby JAR file to the machine where WebLogic Server and the AIS Server are installed.

Note: Make sure that the WebLogic Server has access to the folder where you have saved the JRuby JAR file.

2. Click **Environment**, click **Servers**, and then click **AISInstance** to access the AIS Server instance from the WebLogic Console.
3. Click the **Server Start** tab.
4. Lock and edit the server configuration. In the **Class Path** field, enter the fully qualified location of the JRuby JAR file that you saved in Step 1.
5. Save the configuration. Commit your changes to release the lock.
6. Restart the AIS Server.

Using the Domain Library Method

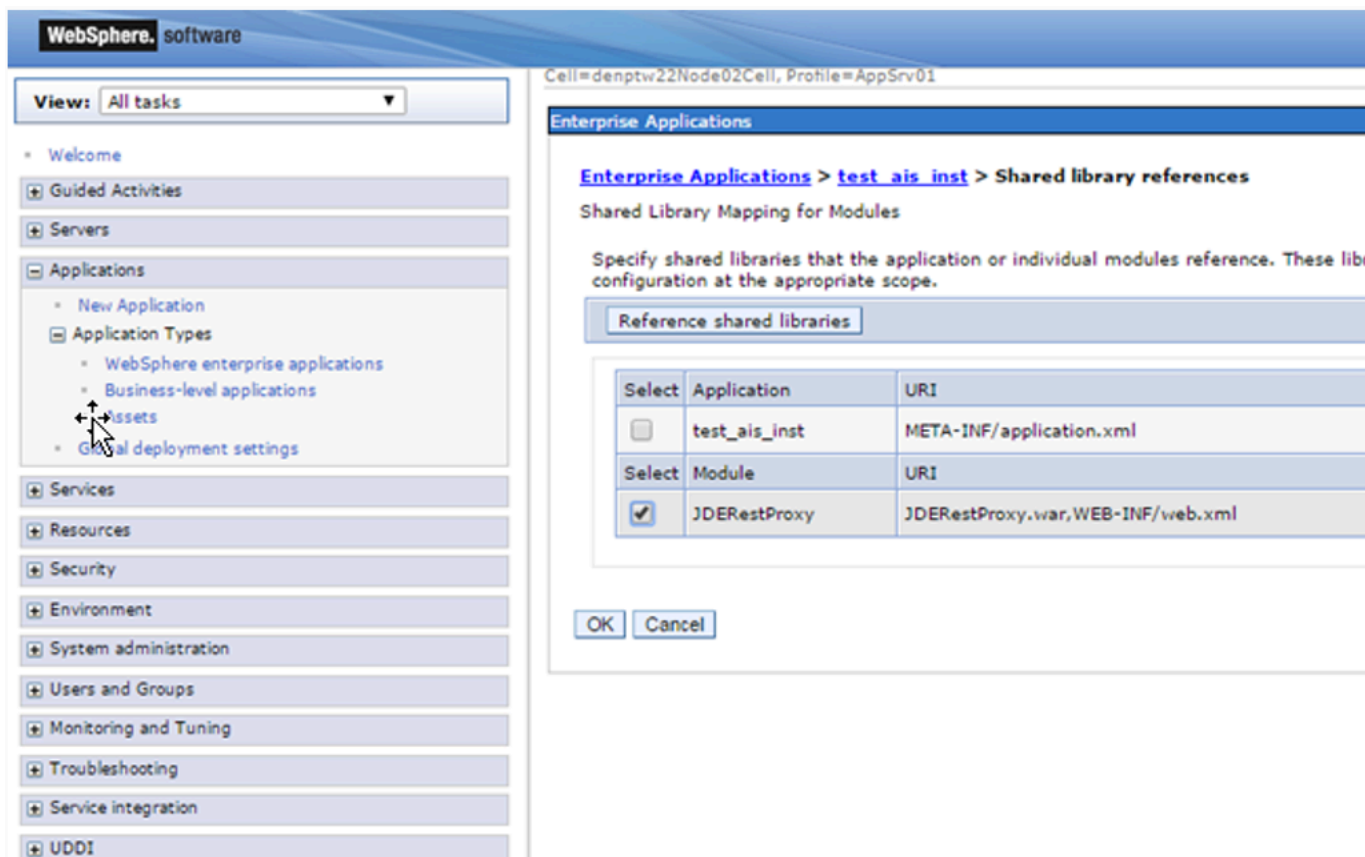
1. Save the JRuby JAR file to the WebLogic Lib folder on the machine where WebLogic, with the AIS Server, is installed (for example: `.../oracle/Middleware/user_projects/domains/<ais_domain>/lib`).
2. Restart the AIS Server.

Deploying JRuby on AIS Server on IBM WebSphere Application Server

You can use the Shared Library method or the classpath method to deploy the JRuby JAR file to the AIS Server on the WebSphere Application Server.

Using the Shared Library Method

1. Save the JRuby JAR file on the machine where the WebSphere Application Server is installed. Make sure that the path is accessible.
2. To add the JRuby JAR file as a shared library:
 - a. On the WebSphere Application Server home page, expand the Environment node and click **Shared Libraries**.
 - b. In the **Class Path** field, enter the fully qualified location of the JRuby JAR file that you saved in Step 1.
3. To associate the shared library with the JDERestProxy application:
 - a. In the left pane, click **Applications**, click **Application Types**, and then click the **WebSphere enterprise applications** link.
 - b. Select the appropriate AIS deployment.
 - c. Under References, select **Shared library references**.



- d. Select the **JDERestProxy** option and then click the **Reference shared libraries** button.
- e. Use the directional arrow to move the JAR file to the selected group.

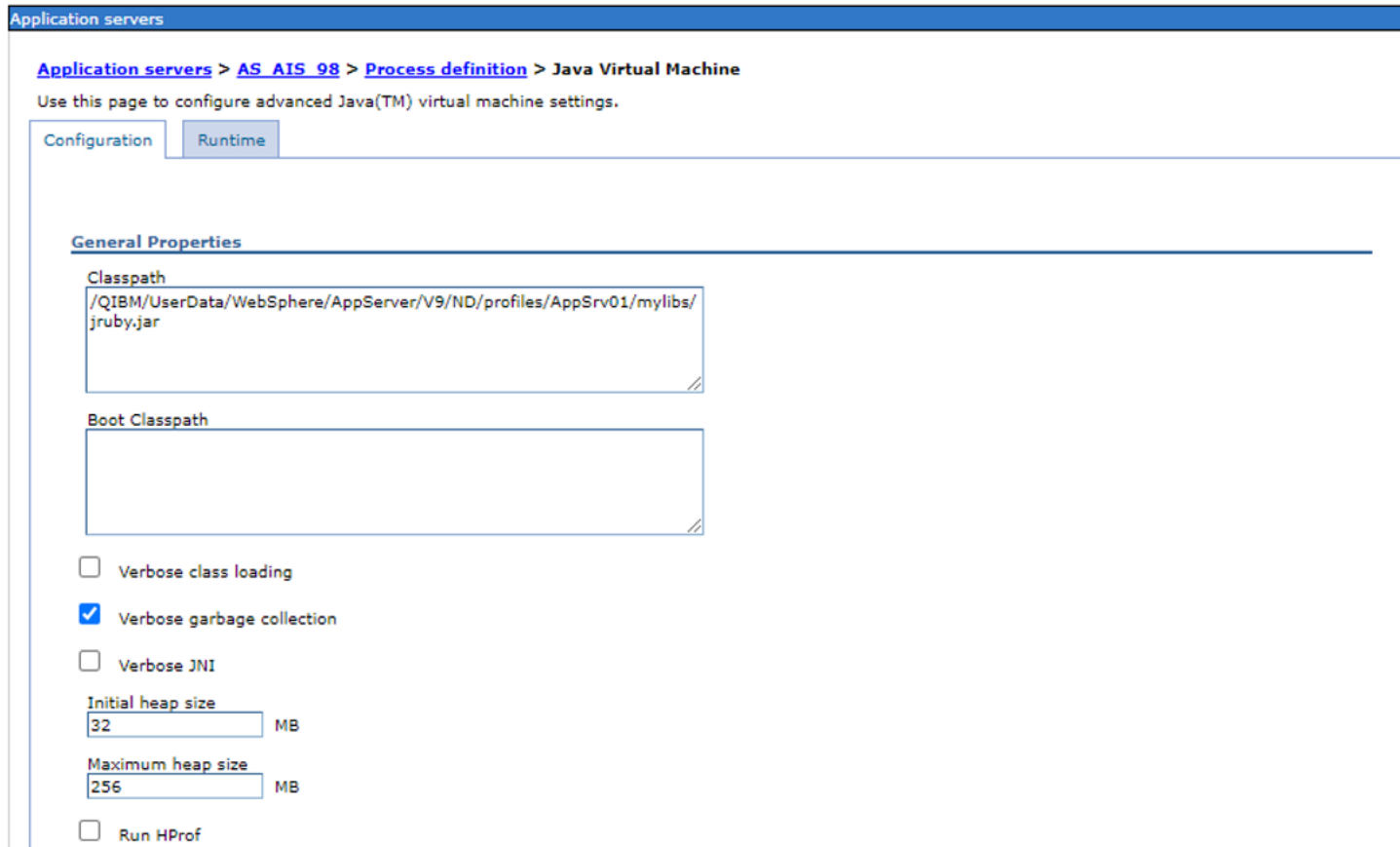
- f. Click **OK**.
- 4. To synchronize Server Manager with the deployed application:

Note: Saving the configuration in the WebSphere Application Server 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 the setting in the **Configuration** section. This step is required so that Server Manager detects a change in the AIS Server when you click the **Synchronize Configuration** button.
- b. Click the **Apply** button to apply the changes and then return to the AIS Server home page.
- c. Click the **Synchronize Configuration** button to restart the AIS Server.

Using the Classpath Method

1. Save the JRbuy Jar file on the machine where WebSphere Application Server is installed. Make sure that the path is accessible.
2. Navigate to **Servers**, click **Server Types** , click **WebSphere application servers** and select the server instance to configure.
3. Click **Java and Process Management** and then click **Process definition**.
4. Click **Java Virtual Machine**.
5. In the Configuration tab, enter the path to the library in the **Classpath** field. For example, enter: `/QIBM/UserData/WebSphere/AppServer/V9/ND/profiles/AppSrv01/mylibs/jruby.jar`



6. Click **OK** and then click **Save** in the Java Virtual Machine page.
7. Click **OK** and then click **Save** in the Process definition page.

8. To synchronize Server Manager with the deployed application:

Note: Saving the configuration in the WebSphere Application Server 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 the setting in the **Configuration** section. This step is required so that Server Manager detects a change in the AIS Server when you click the Synchronize Configuration button.
- b. Click the **Apply** button to apply the changes and then return to the AIS Server home page.
- c. Click the **Synchronize Configuration** button to restart the AIS Server.

Enabling Jython

Download the Jython JAR file from the website at this URL: <https://www.jython.org>.

Note: Download either Jython 2.7.2 (tested with AIS 9.2.5.4 through 9.2.7.3) or Jython 2.7.3 (tested with AIS 9.2.7.4 and later).

This website also provides information on integration between Python and Java.

The `jython-standalone-2.7.2.jar` file is used as an example in this section.

Deploying Jython on AIS Server on Oracle WebLogic Server

You can deploy the Jython JAR file on the WebLogic Server by using the shared library with the library reference option in the `weblogic.xml` file of the AIS Server.

Additionally, you must configure a Java property for each AIS Server.

Using the Shared Library Method and Configuring the Java Property

1. Edit the `.jar` file manifest to include `Specification-Version` so that the manifest file can be uploaded as a library on WebLogic.
 - a. Add this path in the manifest file: `jython-standalone-2.7.2.jar\META-INF\MANIFEST.MF`
 - b. Add this line after **Implementation-Version: 2.7.2**: `Specification-Version: 2.7.2`
 - c. Save the changes to the file and update the file in JAR.
2. Deploy JAR as a shared library on the same WebLogic Server on which the AIS Server (otherwise referred to as the JDERestProxy) is deployed. Ensure that the AIS Server is included as a target for the library.
3. After the library is deployed, select the library to view the library details.
 - a. Note the location of the path, as it will be used in the next step.
 - b. Note the name of the library. For example, `JythonLibrary`.
4. Access the AIS Server instance from the WebLogic console. Click **Environment**, click **Servers**, and then click **AISInstance**.
5. Click the **Server Start** tab.
6. Lock and edit the server configuration. In the **Arguments** field, add the following example argument at the end of the existing argument list. This is the same path where the JAR file is uploaded on the WebLogic Server when you deployed it as a library (See Step 3a). Make sure to add the `Lib` folder at the end of the argument.

```
-Dpython.path= /slot/user/oracle/Middleware/user_projects/domains/<ais_domain>/servers/AdminServer/  
upload/jython-standalone-2.7.2.jar/app/jython-standalone-2.7.2.jar/Lib
```
7. Save the configuration and commit the changes to release the lock.

8. Before uploading the AIS server component to the Server Manager console, edit the `weblogic.xml` in the deployment package.
 - a. Open the AIS server component (par or jar) in a zip file editor and navigate to the `weblogic.xml` file as shown in this example: `\E1_AISServer_9.2.5.4_mm-dd-yyyy_hh_mm.jar\JDERestProxy.ear\JDERestProxy.war\WEB-INF\weblogic.xml`
 - b. Edit the `weblogic.xml` to include the shared library by the name it was created with in WebLogic (See Step 3b), as illustrated in the following 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>JythonLibrary</library-name>
  </library-ref>
</weblogic-web-app>
```

- c. Save the `weblogic.xml` file and re-zip the AIS component.

9. Upload and then deploy the component from Server Manager.

Deploying Jython on AIS Server on IBM WebSphere Application Server

You can use the Shared Library method or the classpath method to deploy the Jython JAR file to the AIS Server on the WebSphere Application Server.

Using the Shared Library Method

1. Save the Jython JAR file in an accessible location on the machine where the WebSphere Application Server is also installed.
2. To add the JAR file as a shared library:
 - a. On the WebSphere Application Server page, expand the Environment node and select **Shared Libraries**.
 - b. In the **ClassPath** field, enter the path to the JAR file location on the server.

3. To 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** option 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**.
4. To synchronize Server Manager with the deployed application:

Note: Saving the configuration in the WebSphere Application Server 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 the setting in the Configuration section. This step is required so that Server Manager detects a change in the AIS Server when you click the **Synchronize Configuration** button.
- b. Click the **Apply** button to apply the changes and then return to the AIS Server home page.
- c. Click the **Synchronize Configuration** button to restart the AIS Server.

Using the Classpath Method

1. Save the Jython Jar file on the machine where WebSphere Application Server is installed. Make sure that the path is accessible.
2. Navigate to **Servers**, click **Server Types**, click **WebSphere application servers** and select the server instance to configure.
3. Click **Java and Process Management** and then click **Process definition**.
4. Click **Java Virtual Machine**.
5. In the Configuration tab, enter the path to the library in the **Classpath** field. For example, enter: `/QIBM/UserData/WebSphere/AppServer/V9/ND/profiles/AppSrv01/mylibs/jython.jar`

- Enter the generic JVM argument in the Generic JVM arguments field. For example, enter: -
Dpython.path= /QIBM/UserData/WebSphere/AppServer/V9/ND/profiles/AppSrv01/mylibs/jython.jar/Lib

Application servers

Application servers > AS_AIS_98 > Process definition > Java Virtual Machine

Use this page to configure advanced Java(TM) virtual machine settings.

Configuration Runtime

General Properties **Additional Properties**

Classpath
/QIBM/UserData/WebSphere/AppServer/V9/ND/profiles/AppSrv01/mylibs/jython.jar

Boot Classpath

Verbose class loading

Verbose garbage collection

Verbose JNI

Initial heap size
32 MB

Maximum heap size
256 MB

Run HProf

HProf Arguments

Debug Mode

Debug arguments
-agentlib:jdwp=transport=dt_socket,server=y,suspend=n,address=7777

Generic JVM arguments
-Djavax.net.ssl.trustStoreType=JKS -Djavax.net.ssl.trustStorePassword=changeit -Djavax.net.ssl.trustStore=/QOpenSys/QIBM/ProdData/JavaVM/jdk80/64bit/jre/lib/security/cacerts -Djavax.net.ssl.keyPassword=WebAS -Djavax.net.ssl.keyStoreType=PKCS12 -Djavax.net.ssl.keyStorePassword=WebAS -Djavax.net.ssl.keyStore=/oracacert/key.p12 -Djdk.tls.client.protocols=TLSv1.2 -DcloneId=AS_AIS_98 -agentlib:getClasses -Dpython.path=/QIBM/UserData/WebSphere/AppServer/V9/ND/profiles/AppSrv01/mylibs/jython.jar/Lib

[Custom properties](#)

- Click **OK** and then click **Save** in the Java Virtual Machine page.
- Click **OK** and then click **Save** in the Process definition page.
- To synchronize Server Manager with the deployed application:

Note: Saving the configuration in the WebSphere Application Server will redeploy the application, but you must synchronize Server Manager to recognize the deployed application.

- In Server Manager, locate the AIS Server and update the setting in the **Configuration** section. This step is required so that Server Manager detects a change in the AIS Server when you click the Synchronize Configuration button.
- Click the **Apply** button to apply the changes and then return to the AIS Server home page.
- Click the **Synchronize Configuration** button to restart the AIS Server.

Installing Optional Scripting Languages on the AIS Server (Release 9.2.6)

Starting with Tools Release 9.2.6, the AIS Server is no longer preconfigured to create and run Groovy scripts. To edit or run existing Groovy scripts or to create new ones you must first configure the AIS servers with the Groovy libraries. This is also true for Jython. The AIS servers are preconfigured for JRuby.

Note: Note that the following requirement applies to Tools Release 9.2.7.4 and later only. The JD Edwards EnterpriseOne AIS Server is shipped along with JRuby 9.4.2.0 in Tools Release 9.2.7.4. Therefore, if you are using Jython as your scripting language, you must move to the latest Jython release (at least to Jython 2.7.3 release). Delete the previous Jython JAR file in Server Manager Console and upload the latest one. For more information, see [Configure the Scripting Libraries](#) in the *JD Edwards EnterpriseOne Tools Server Manager Guide* for information about how to configure Groovy and Jython.

To enable the use of additional scripting languages for an AIS Server instance, you must provide an associated JAR file and perform the configuration steps.

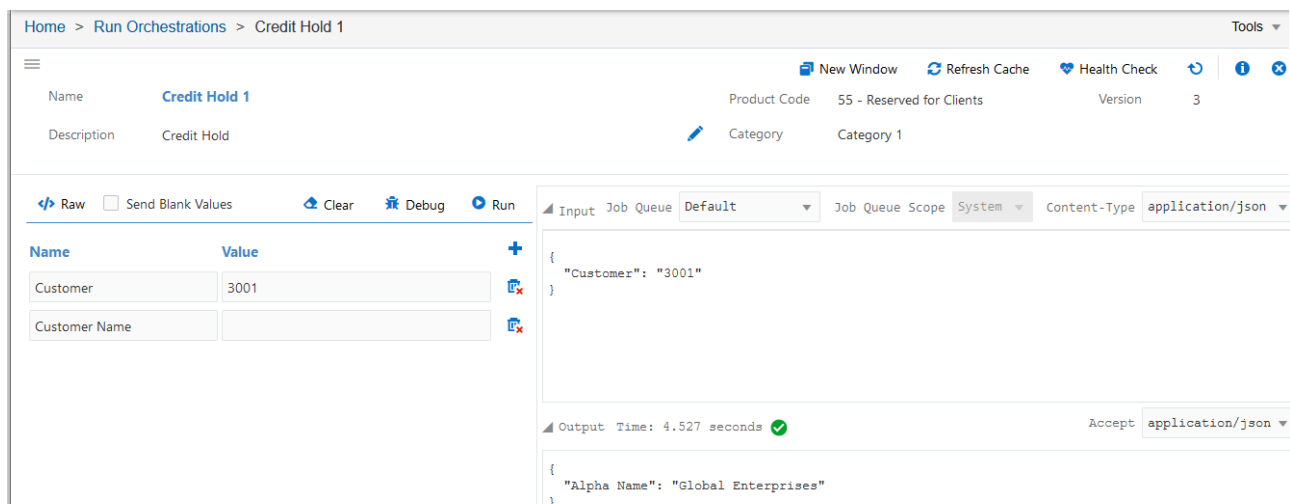
After you complete the required configuration, you can see the languages listed in the drop-down menu in the script section in the Orchestrator Studio.

11 Testing Orchestrations

Understanding Run Orchestrations

You can use the Run Orchestrations page to test the orchestrations. You can access the Run Orchestrations page from the Orchestrator Studio Home page. Alternatively, you can access the Run Orchestrations for a specific orchestration by clicking the Start node of the orchestration, and then clicking the Run Orchestration icon. Oracle recommends that you use the Run Orchestrations page with an EnterpriseOne test environment, because any tests that are performed result in EnterpriseOne transactions that add data to the database.

In the Run Orchestrations page, you can test your personal orchestrations or shared orchestrations to which you have access. When you select an orchestration to test, the Run Orchestrations page displays the inputs that are defined for the orchestration. You can enter values for the inputs in the Name and Value fields and then run the test. Alternatively, the Run Orchestrations page gives you the option to enter raw JSON or XML code for the input. The following figure shows an example of testing an orchestration. The orchestration has been designed to place a customer on credit hold, and customer number 3001 has been entered for the Customer input.



The Run Orchestrations page displays a green check mark if the orchestration completes successfully. The Input sections displays the orchestration request in JSON or XML format depending on the Content-Type option that you have selected. Similarly, the Output section displays the orchestration response in either JSON or XML format depending on the Accept option that you have selected. If the orchestration is unsuccessful, the Run Orchestrations page displays an "x" symbol and the Output area displays the error.

To execute the test, the Run Orchestrations page 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

Run Orchestrations page passes the request to the AIS Server in JSON or XML format. The AIS Server provides a JSON over REST interface through HTTP. The REST interface is a lightweight interface that enables AIS clients to interact with EnterpriseOne applications and forms.

Starting with Tools release 9.2.4.3, you can launch the Run Orchestrations page in a new window. This enables you to switch back and forth between the orchestration design page and the Run Orchestrations page.

As orchestrations become more complex, orchestration designers need a simple view into the data that is passed from one step to another to ensure that the orchestration is working as designed. With Tools release 9.2.4.3, you track and inspect an orchestration on a step-by-step basis. You can debug an orchestration by setting breakpoints at strategic points throughout an orchestration and testing the orchestration until you find the problem. This enables you to view the values of input parameters, output parameters, and variables at the specified point.

Starting with Tools Release 9.2.8.2, the exception response from an error on a form service request includes the field name, data item alias, and table where the data is stored. For grid errors, the exception response also includes the row title and row number on which the error occurred. This additional information enables you to identify and correct errors that occur on form service requests.

Using cURL to Simulate Testing from a Third-Party Application or IoT Device

As an alternative to testing orchestrations, 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 Run Orchestrations page, which displays the JSON string in the Inputs section when you test an orchestration.

Testing an Orchestration

To test an orchestration:

1. On the Orchestrator Studio Home page, click the **Run Orchestrations** icon.

You can also open an orchestration, click the **Start** icon, and select **Run Orchestrations**.

2. In the Run Orchestrations side panel, select an orchestration from personal orchestrations or the shared orchestrations to which you have access.

The Run Orchestrations page displays a sample request and a response for the selected orchestration in the Input and Output sections.

The Run Orchestrations page displays the inputs or the name-value pairs, which should be defined in the orchestration as the expected inputs to the orchestration. If the inputs do not appear, return to the corresponding orchestration design page and define the inputs for the orchestration before continuing. An input displayed in italics indicates that it is a required input.

Starting with Tools Release 9.2.6.2, you can use the **Array Inputs** icon to define inputs for an array. Click the **Load Defaults** icon to populate the default values of the array. You can enable the **Raw** option to edit the values in the JSON format.

All orchestrations that are created in Orchestrator Studio 9.2.4 or later are saved as version 3 orchestrations. You cannot change this value. If you want to update a version 3 orchestration that was created in a previous version of the Orchestrator Studio, see [Updating to Version 3 Orchestrations](#).

3. In the Input section, in the Content-Type down-drop list, select one for the following to display the input request in either JSON or XML format:

- o `application/json`
- o `application/xml`

4. In the Output section, in the Accept down-drop list, select one for the following to display the output response in either JSON or XML format:

- o `application/json`
- o `application/xml`
- o `application/octet-stream` (Release 9.2.6)

(Release 9.2.8.3) The system displays the default value as `application/octet-stream` in the Accept Type drop-down list when the orchestration is configured to return a file.

5. Select the **Send Blank Values** check box to send inputs with blank values to the orchestration.

By default, the Send Blank Values check box is deselected. Therefore, only the inputs for which you have entered the values are passed on to the orchestration and the default values are not be applied to the orchestration.

6. In the Value column, enter a value for each input.

These are the values that the orchestration will pass to EnterpriseOne.

7. If needed, you can click the **Add** icon (+) to add and define additional inputs.

For an orchestration that contains an array input, the Name column displays the input as `array name[index].array input`. This enables you to identify the input as an array. To add multiple array inputs to the same orchestration, add multiple rows to the input grid, copy the array input, paste the array input in the Name column for the multiple rows, and update the index number.

You can click the **Clear** button to restore the inputs to its original state and undo any changes that you made in the input area.

8. Click **Run** to test the orchestration.

Depending on the Content-Type and Accept drop-down menu selection, the Run Orchestrations page displays the orchestration input and response in the Input and Output sections, respectively, in JSON or XML format.

If the test is successful, a green check mark is displayed next to the Output section. If desired, you can access EnterpriseOne and confirm that the transaction was completed by the orchestration.

If the test is unsuccessful, an "x" symbol is displayed and the Output area displays the error response in the format that you have selected. If the orchestration fails, modify the orchestration and test it again.

You can use the Refresh Cache button on the Run Orchestrations page to access the components (such as service requests, orchestrations, and so on) that are created outside of your current session. To access the objects that were shared while the session was open, you have to close the session and sign in again.

Starting with Tools release 9.2.4.3, you can click the New Window button to launch the Run Orchestrations page in a new window. This functionality enables you to switch between the orchestration design page and the Run Orchestrations page. Therefore, you can make changes to the orchestration on the design page while retaining the values that you have entered for the inputs on the Run Orchestration page. Only a single Run Orchestrations child window is launched at any time.

Starting with Tools release 9.2.4.4, you can override the queues while testing the orchestrations. Select the values in the **Job Queue** and the **Job Queue Scope** drop-down menus as required and click **Run** to test the orchestration with the new values.

To navigate back to the design page of the parent orchestration, click the **Back** button on the menu bar of the Run Orchestrations child window.

9. To test another orchestration or start over, click the **Clear** button to reset, which clears all the values in the form. To use JSON input for the orchestration test:

1. In the Input section, select **application/json** from the Content -Type drop-down list.
2. In the Output section, select **application/json** from the Accept drop-down list.
3. Click the **Raw** button.
4. Enter the input in JSON format directly in the input grid.

You can click the Raw button again to enter the inputs in the Name-Value input grid.

The views between the JSON format and Name-Value grid are in sync. The input you enter in the JSON format reflects in the input area in the Name-Value grid or vice versa. If you delete any of the inputs in the JSON format, the deleted inputs are not visible in the Name-Value grid

5. Click the **Format** button to verify that the syntax of the JSON is correct.

If the JSON input is not formatted correctly, the system displays an error message about the JSON.

6. Click **Run** to test the orchestration.

The Input and Output sections display the input and response, respectively, in the JSON format.

To use XML input for the orchestration test:

1. In the Input section, select **application/xml** from the Content -Type drop-down list.
2. In the Output section, select **application/xml** from the Accept drop-down list.
3. Click the **Raw** button.
4. Enter the input in XML format directly in the input grid.

You can click the **Raw** button again to enter the inputs in the Name-Value input grid. The input you enter in the XML format is in sync with input entered in the Name-Value pair or vice versa. Similarly, if you delete any of the inputs in the XML format, the deleted inputs are not visible in the Name-Value grid.

5. Click the **Format** button to verify that the syntax of the XML input is correct.

If not formatted correctly, a dialog box displays an error message.

6. Click **Run** to test the orchestration.

The Input and Output section displays the input and response, respectively, in the XML format.

Debugging an Orchestration (Release 9.2.4.3)

This section contains the following topics:

- [Understanding Debug Orchestration](#)
- [Debug Orchestration Features](#)
- [Debugging an Orchestration](#)

Understanding Debug Orchestration

The JD Edwards EnterpriseOne Orchestrator provides a way to debug your orchestrations. Debug Orchestrations is the tool that you can use to determine the state of your orchestration at any point during execution. Debug Orchestrations provides a view into an orchestration, enabling the designer to run an orchestration step-by-step and ensure that the data is correct at each step until the final orchestration output.

As orchestrations become more complex, orchestration designers need a view of the data that is passed from one step to another to ensure that the orchestration is working as designed. Use the Debug Orchestrations to stop execution so that you can see the state of the orchestration at a specific point by reviewing the values of input parameters, output parameters, and variables. When the orchestration execution is stopped, you can review the output line by line to check for issues.

Note: As Debug Orchestrations enables you to change the live values of variables and orchestration inputs, it is recommended that you do not use this tool in a production environment.

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

Debug Orchestration Features

Debug Orchestrations enables you to track and inspect an orchestration on a step-by-step basis. You can debug an orchestration by setting breakpoints at strategic points throughout the orchestration and testing the orchestration until you find the problem. When you debug an orchestration and encounter a step at which the orchestration fails, you can view the output data, and inspect the live values of the variables used in that particular step.

You can repeatedly run the steps and view the variables of a specific step in an orchestration. By observing specific variables while the orchestration runs, you can isolate where the orchestration begins to fail and what exactly it is doing at that stage.

You can debug an orchestration by clicking the Debug button on the Run Orchestrations page. In the debug mode, the Orchestrator Studio renders a view of an orchestration in which you can set breakpoints. The debug mode also contains the Debug Orchestration panel.

The options available on the Debug Orchestration panel are described in the following table:

Icons	Description
Run/Resume	Runs an orchestration until completion unless you set up breakpoints. If you set breakpoints, this option executes an orchestration till the breakpoint is reached or resumes a suspended orchestration.
Step Forward	Executes the orchestration one step at a time. When a step has been executed successfully, a green check mark appears on the step.
Stop Orchestration	Stops the execution of an orchestration.
Show/Hide Orchestration Inputs	Shows or hides the dialog box to enter or edit the initial orchestration inputs.
Clear All Breakpoints	Removes all the breakpoints that you have set for the orchestration in the debug mode.
Quit Debugging	Exits Debug Orchestrations.

For each step, the Debug Orchestration panel displays the following information:

Data

The Data tab displays all the available data that the orchestration consumes and generates, which includes the data that the orchestration consumes from a step. This tab displays the following information:

- **Step Variables.** Displays all the variables created by the steps that have been executed. Variables from steps are added to the top of the list, so the most recent variables are shown at the top. Variables of single values like strings and numbers can be changed before resuming the orchestration. Variables containing arrays or JSON objects are read only.
- **Orchestration Inputs.** Displays the values for inputs that were defined initially to start the orchestration.
- **Debug Overrides (Release 9.2.5).** Displays the number of records to debug. Currently, the only override field is Debug Max Records and it displays the number of records to fetch from a Data Request or a Form Request grid while debugging. The default value is 10 and hence the system returns 10 records by default when you debug.
- **System Values.** Displays the values for the default inputs: User Address Book Number, User Name, System Date, and Orchestration Input. The values for these inputs represent the originator of the orchestration when executed at runtime. The system values are not editable as these are fixed for an orchestration.
- **History.** Displays a record of the values for a variable that is overwritten. For each overwritten variable, click the History icon displayed in its row. The History dialog box lists the orchestration steps where the specific variable is used along with the values. Similarly, the Output tab displays the history of the overwritten output values. (Release 9.2.4.4) The values and the history of inputs and variables are also available during the debug. You can also view the number of times the inputs and variables were changed after the orchestration started.

Last Step

Displays the outputs generated from the previous step that was executed in the orchestration.

(Release 9.2.4.4) You can click the **Raw Output** button to view the complete response of the step in the JSON format.

Output

Displays the accumulative output generated for all the steps in the orchestration until the breakpoint.

Setting Breakpoints

Breakpoints enable you to define where or when to halt the execution of an orchestration. You use breakpoints to run the orchestration until it reaches a certain step.

In the debug mode, a blue circle appears to the right of the step in the orchestration flow. You can set a breakpoint by clicking either the blue circle or the step in the orchestration flow. A blue dot appears next to the orchestration step indicating that the breakpoint has been set.

If you set a breakpoint and click the Run/Resume icon on the Debug Orchestrations panel, the orchestration runs until it encounters that step. To continue execution after the breakpoint, you can use the Run/Resume or Step Forward icons on the Debug Orchestrations panel.

Debugging an Orchestration

To debug an orchestration:

1. On the Orchestrator Studio Home page, click the **Run Orchestrations** icon.
2. In the Run Orchestrations side panel, select an orchestration from personal orchestrations or shared orchestrations.

On the Run Orchestrations page, the system a sample request and a response for the selected orchestration.

3. In the **Value** column, enter a value for each input.
4. Click the **Debug** button.

The orchestration is displayed in the debug mode with blue circles next to the orchestration steps.

Alternatively, you can access Design Orchestrations from the orchestration design page by choosing the Run icon from the Start node of the orchestration. That takes you to Run Orchestration page on which you can click the Debug button.

5. Set a breakpoint by clicking the blue circle on the right of the orchestration step.

The Breakpoint icon is a toggle button. Therefore, you can remove the breakpoint also by clicking the blue dot next to the orchestration step.

6. On the Debug Orchestration panel, click the Run/Resume icon.

When the execution stops at a breakpoint, you can use the Data, Last Step, and Output tabs to review and modify the values of the runtime variables.

7. From the Debug Orchestrations panel, select one of these options:
 - o Run/Resume
 - o Step Forward
 - o Stop Orchestration
 - o Show/Hide Orchestration Inputs
 - o Clear All Breakpoints
 - o Quit Debugging

12 Administering the Orchestrator Studio and Orchestrations

Understanding Orchestration Life Cycle Management

Orchestration components that are 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. The Orchestrator Studio provides the capability to create notification and to schedule components, *Setting Up UDO Security for Orchestrator Studio Users* which are also stored and managed as UDOs in EnterpriseOne.

You can 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, P98220W can be used 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 a 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* 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.

Managing Orchestrator Studio Security

An administrator must use EnterpriseOne application security to enable the user to access the Studio (W98I0000A), the Scheduler page (W98I0000B), and the Admin services (W98I0000C).

P98I0000 (Orchestrator Studio) is a proxy application that is associated with the Orchestrator Studio. The P98I0000 application enables you to determine the different levels of security for the Orchestrator Studio, the Admin service, and

the Scheduler that includes access to the Scheduler services and the Scheduler page. The application represents the service level security as well as the Orchestrator Studio access security.

You can set up security for the Orchestrator Studio application (P98I0000) using the standard application security in the EnterpriseOne Security Workbench (P00950). In the Security Workbench, the only application security option that applies for managing the Orchestrator Studio security is the **Run** security option.

If you are secured out of the P98I0000 application, you cannot log in to the Orchestrator Studio, run the Admin services, or run any of the Scheduler services.

The following tables describes the forms used to define the Orchestrator Studio security.

Form Name	Form ID	Usage
Studio	W98I0000A	Log in to the Orchestrator Studio.
Scheduler	W98I0000B	Run the Scheduler services or use the Scheduler page in the Orchestrator Studio.
Admin Services	W98I0000C	Run the Admin services or clear caches using the Refresh Cache button on the Run Orchestrations page.

See *Managing Application Security* in the *JD Edwards EnterpriseOne Tools Security Administration Guide* for instructions on how to set up application security in EnterpriseOne.

The following table lists some security scenarios:

Permission to				Access				
P98I0000 (All)	W98I0000A (Studio)	W98I0000B (Scheduler)	W98I0000C (Admin service)	Studio (Login)	Studio (Scheduler)	Studio (Clear Cache)	Scheduler (REST Service)	Admin (REST Service)
N	N	N	N	Not allowed	Not allowed	Not allowed	Not allowed	Not allowed
Y	N	N	Y	Not allowed	Not allowed	Not allowed	Not allowed	Allowed
Y	N	Y	N	Not allowed	Not allowed	Not allowed	Allowed	Not allowed
Y	N	Y	Y	Not allowed	Not allowed	Not allowed	Allowed	Allowed
Y	Y	N	N	Allowed	Not accessible	Not allowed	Not allowed	Not allowed
Y	Y	N	Y	Allowed	Not accessible	Allowed	Not allowed	Allowed

Permission to				Access				
Y	Y	Y	N	Allowed	Allowed	Not allowed	Allowed	Not allowed
Y	Y	Y	Y	Allowed	Allowed	Allowed	Allowed	Allowed

Setting Up UDO Security for Orchestrator Studio Users

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. Therefore, 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 types of orchestration components (orchestration, service request, rule, cross reference, and white list), you must activate all 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.

You 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* .

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.

Alternatively, you can use the Clear Data Cache functionality for clearing the data cache on the AIS and HTML Servers. See "Clear Data Cache" in the *JD Edwards EnterpriseOne Tools Server Manager Guide* .

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.

Activating the AIS Administration Service

Before you can use the AIS Administration Service, you must set up the AIS Administration Service users. For information on setting up users for AIS Administration Service, see *Managing Orchestrator Studio Security*.

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"}
```

Setting Up Orchestrator Health and Exception Monitoring

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](#)

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).

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.

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:

- o **FSR_ERROR**

Enter if you want **errors** returned by a form request to be processed as a failure.

- o **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.

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* .

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* .

13 Understanding the AIS Server Discovery Service

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 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 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 or Generic.
- Input. The input value.
- Input type. String, numeric, or date.

Example: Orchestration Details in the JSON Response from the Discovery Service provides an example of the orchestration details in a JSON response.

Example: Orchestration 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 Measuement
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"
  }
]
}
```


14 Open Standards for Orchestrations

Understanding Open Standards for Orchestrations

Open standards for orchestrations allow external systems to discover orchestrations, including their inputs and outputs, by using the OpenAPI 2.0 standard (formerly known as "Swagger") or OpenAPI 3.0. This also enables products that conform to this standard to discover and execute orchestrations natively using this standard. Many products, including development, integration, automation, testing, and documentation tools, have libraries that support these standards.

Open standards for orchestrations give you the ability to:

- Return a list of orchestrations available on an AIS server secured by standard authentication.
- Return a list of inputs and input types for an orchestration.
- Return a list of outputs and output types for an orchestration.
- Return the text description of an orchestration.
- Discover orchestrations from any OpenAPI 2.0 or 3.0 compliant consumer.

This applies to any orchestrations that you have created.

You can generate a document that lists all available orchestrations, including their inputs, in a format that complies with specifications such as OpenAPI and Swagger. This feature enables integrations with third-party systems that need to discover and invoke orchestrations as JD Edwards EnterpriseOne REST APIs.

To generate the Swagger or the OpenAPI documents, in the Orchestrator Studio home page, click the **Tools** menu, click **Definitions**, and then select the **Swagger** or **OpenAPI** options. The system downloads the swagger.json or the openapi.json file respectively.

However, there are instances in which you may want to exclude certain orchestrations from the JSON document. For example, some orchestrations might not be intended to be directly called from an external system but rather only to be called as child orchestrations from a parent orchestration. Starting with Tools Release 9.2.7.2, you can mark orchestrations as private, thus excluding them from the list of available orchestrations in the OpenAPI and Swagger documents.

To exclude an orchestration from the Swagger and OpenAPI documents, open the orchestration and click the **Do not publish in Open API** (lock) button at the upper corner of the page. The **Publish in OpenAPI** (unlock) button is enabled by default for the existing orchestrations.

Discovering Orchestrations Using Open Standards

To discover orchestrations, you use the open-api-catalog endpoint on the AIS Server.

The endpoint URL for OpenAPI (Swagger) 2.0 is:

```
http://<server>:<port>/jderest/v3/open-api-catalog
```

The endpoint URL for OpenAPI 3.0 is:

```
http://<server>:<port>/jderest/v3/open-api-catalog3
```

Note: If you do not specify the version, the current version is used by default.

The discovery request must include authentication credentials for security. The request returns a standard Swagger 2.0 or OpenAPI 3.0 JSON formatted response with all of the orchestrations and related metadata that are available to the user whose credentials were used, subject to EnterpriseOne user-defined object security.

You can also add the orchestration name to the end of the URL to retrieve information only for that specific orchestration.

For example:

```
http://<server>:<port>/jderest/v2/open-api-catalog/MyOrchestration
```

You can import the JSON response into a third-party tool to review all the orchestrations. For example, importing into a documentation editor such as Swagger provides descriptions and other information, which enables you to document your own orchestrations.

Using a third-party tool, you can perform a POST or GET to execute a specific orchestration. However, if the orchestration inputs include an array, only the POST operation is available. A GET requires that the inputs are on the URL, while a POST requires the inputs as JSON in the request body.

XML Inputs and Outputs

You have the ability to execute an orchestration using XML as the input or output parameter content type.

For those applications that require producing and/or consuming XML rather than JSON, XML can be used as the transport format.

Setting the 'Accept' header on the orchestration execution request will return the output in XML format. Setting the 'Content-Type' header on the orchestration execution POST request will require the input to be in XML format.

Each orchestration input should be a single XML element, and the inputs should be contained within some outer parent element.

For example:

```
<inputs>
  <input1>value1</input1>
  <input2>value2</input2>
</inputs>
```

Note: It is recommended that inputs do not have spaces or special characters as these characters require tokenizing in the XML.

Orchestrator Studio enables you to execute an orchestration using XML as the input or output parameter content type. You can use the Run Orchestrations page to test the orchestration with XML format. Setting the '**Accept**' drop-down option to `application/xml` in the Output section will return the output in XML format. Setting the '**Content-Type**' drop-down option to `application/xml` in the Input section will require the input to be in XML format.