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

## Preface

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Audience</td>
<td>ix</td>
</tr>
<tr>
<td>Documentation Accessibility</td>
<td>ix</td>
</tr>
<tr>
<td>Related Information</td>
<td>ix</td>
</tr>
<tr>
<td>Conventions</td>
<td>x</td>
</tr>
</tbody>
</table>

## 1 Understanding the JD Edwards EnterpriseOne Orchestrator

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>JD Edwards EnterpriseOne Orchestrator Overview</td>
<td>1-1</td>
</tr>
<tr>
<td>1.1.1 Invoking Orchestrations and Notifications from an EnterpriseOne Interactive Application Using Form Extensibility</td>
<td>1-3</td>
</tr>
<tr>
<td>1.1.2 Invoking Orchestrations and Notifications from an EnterpriseOne Interactive or Batch Application</td>
<td>1-3</td>
</tr>
<tr>
<td>1.2 How It Works</td>
<td>1-3</td>
</tr>
<tr>
<td>1.3 EnterpriseOne Architecture for Orchestrator</td>
<td>1-5</td>
</tr>
</tbody>
</table>

## 2 Getting Started

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Certifications (Formerly Known as Minimum Technical Requirements)</td>
<td>2-1</td>
</tr>
<tr>
<td>Prerequisites</td>
<td>2-1</td>
</tr>
<tr>
<td>Before You Begin</td>
<td>2-3</td>
</tr>
<tr>
<td>Accessing the Orchestrator Studio</td>
<td>2-3</td>
</tr>
<tr>
<td>Set Up a Temporary Directory on the AIS Server for File Transfers</td>
<td>2-3</td>
</tr>
</tbody>
</table>

## 3 Designing an Orchestration

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Understanding the Orchestration Design Process</td>
<td>3-1</td>
</tr>
<tr>
<td>Identifying the Problem and Solution</td>
<td>3-2</td>
</tr>
<tr>
<td>Identifying the Data for the Orchestration</td>
<td>3-2</td>
</tr>
<tr>
<td>Identifying the Rules for the Orchestration</td>
<td>3-3</td>
</tr>
<tr>
<td>Identifying the Cross Reference and White List Information for the Orchestration</td>
<td>3-3</td>
</tr>
<tr>
<td>Identifying the Service Request Information for the Orchestration</td>
<td>3-4</td>
</tr>
</tbody>
</table>

## 4 Configuring Orchestration Components using Orchestrator Studio 9.2.4

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Understanding the Orchestrator Studio and Orchestrations</td>
<td>4-1</td>
</tr>
<tr>
<td>Invoking an Orchestration</td>
<td>4-3</td>
</tr>
<tr>
<td>Reusable Orchestration Components</td>
<td>4-3</td>
</tr>
<tr>
<td>Section</td>
<td>Title</td>
</tr>
<tr>
<td>---------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>4.1.3</td>
<td>Orchestrations as User Defined Objects</td>
</tr>
<tr>
<td>4.2</td>
<td>Navigating the Orchestrator Studio</td>
</tr>
<tr>
<td>4.2.1</td>
<td>Orchestrator Studio Features</td>
</tr>
<tr>
<td>4.2.2</td>
<td>User Defined Object (UDO) Features in the Orchestrator Studio</td>
</tr>
<tr>
<td>4.2.3</td>
<td>Working with the Graphical Representation of an Orchestration</td>
</tr>
<tr>
<td>4.3</td>
<td>Creating Service Requests</td>
</tr>
<tr>
<td>4.3.1</td>
<td>Understanding Service Requests</td>
</tr>
<tr>
<td>4.3.2</td>
<td>Creating a Component</td>
</tr>
<tr>
<td>4.4</td>
<td>Working with Orchestrator Categories (Release 9.2.4.2)</td>
</tr>
<tr>
<td>4.4.1</td>
<td>Working with Categories</td>
</tr>
<tr>
<td>4.5</td>
<td>Configuring a Form Request in the Orchestrator Studio</td>
</tr>
<tr>
<td>4.5.1</td>
<td>Loading EnterpriseOne Application Form Fields and Controls</td>
</tr>
<tr>
<td>4.5.2</td>
<td>Configuring Form Request Actions</td>
</tr>
<tr>
<td>4.5.3</td>
<td>Configuring the Order of Execution</td>
</tr>
<tr>
<td>4.5.4</td>
<td>Populating Multiple Grids with Repeating Inputs (Optional)</td>
</tr>
<tr>
<td>4.5.5</td>
<td>Creating a Form Request with the JD Edwards EnterpriseOne</td>
</tr>
<tr>
<td></td>
<td>Orchestrator Process Recorder</td>
</tr>
<tr>
<td>4.5.5.1</td>
<td>Rules for Creating a Form Request in the Process Recorder</td>
</tr>
<tr>
<td>4.5.5.2</td>
<td>Prerequisites</td>
</tr>
<tr>
<td>4.5.5.3</td>
<td>Using the Process Recorder to Create a Form Request</td>
</tr>
<tr>
<td>4.6</td>
<td>Configuring a Data Request</td>
</tr>
<tr>
<td>4.6.1</td>
<td>Understanding Data Request</td>
</tr>
<tr>
<td>4.6.2</td>
<td>Configuring a Data Request to Return Field Data</td>
</tr>
<tr>
<td>4.6.3</td>
<td>Configuring a Data Request with Data Aggregation</td>
</tr>
<tr>
<td>4.6.4</td>
<td>Viewing and Copying JSON Code of a Data Request</td>
</tr>
<tr>
<td>4.6.5</td>
<td>Configuring Form Request and Data Request Processing</td>
</tr>
<tr>
<td>4.6.5.1</td>
<td>Page Size Considerations</td>
</tr>
<tr>
<td>4.6.5.2</td>
<td>Processing Settings</td>
</tr>
<tr>
<td>4.7</td>
<td>Configuring a Report Service Request</td>
</tr>
<tr>
<td>4.8</td>
<td>Configuring a Watchlist Service Request</td>
</tr>
<tr>
<td>4.9</td>
<td>Configuring a Message Request</td>
</tr>
<tr>
<td>4.10</td>
<td>Configuring a Connector Service Request</td>
</tr>
<tr>
<td>4.10.1</td>
<td>Understanding Connector Service Requests</td>
</tr>
<tr>
<td>4.10.2</td>
<td>Before You Begin</td>
</tr>
<tr>
<td>4.10.3</td>
<td>Configuring a Connector Service Request to Invoke an Orchestration or</td>
</tr>
<tr>
<td></td>
<td>Notification</td>
</tr>
<tr>
<td>4.10.4</td>
<td>Configuring an Open API Connector to Invoke a REST Service</td>
</tr>
<tr>
<td>4.10.5</td>
<td>Configuring a REST Connector to Invoke a REST Service</td>
</tr>
<tr>
<td>4.10.6</td>
<td>Configuring a REST Connector to Invoke REST Services from a Local AIS</td>
</tr>
<tr>
<td>4.10.7</td>
<td>Configuring a REST Connector to Transfer Files to a REST Service</td>
</tr>
<tr>
<td>4.10.8</td>
<td>Configuring a Database Connector</td>
</tr>
<tr>
<td>4.10.9</td>
<td>Configuring a Connector to Transfer Files Using File Transfer Protocol (FTP)</td>
</tr>
<tr>
<td>4.10.9.1</td>
<td>Configure an FTP Connector to Transfer the Output of a Report Service Request</td>
</tr>
<tr>
<td>4.10.9.2</td>
<td>Configure an FTP Connector to Transfer a File from an FTP Server to the</td>
</tr>
<tr>
<td></td>
<td>AIS Server</td>
</tr>
<tr>
<td>4.10.9.3</td>
<td>Configure an FTP Connector to Transfer a File from the AIS Server to an</td>
</tr>
<tr>
<td></td>
<td>External Server</td>
</tr>
</tbody>
</table>
8.1.3 Adding the Service Request to an Orchestration .......................................................... 8-2

9 Creating Custom Java for Orchestrations
9.1 Understanding Custom Java for Orchestrations .......................................................... 9-1
9.2 Creating Custom Java ....................................................................................................... 9-1
9.2.1 Using Custom Java for the Orchestration Service Request ...................................... 9-2
9.2.2 Using Custom Java for the Orchestration Rule ......................................................... 9-2
9.3 Deploying Custom Java .................................................................................................. 9-2
9.3.1 Deploying Custom Java on AIS Server on Oracle WebLogic Server ......................... 9-2
9.3.2 Deploying Custom Java on AIS Server on IBM WebSphere Application Server....... 9-3

10 Using Apache Groovy for Custom Service Requests, Rules, and Manipulating Output
10.1 Understanding Groovy for Orchestration Components ............................................... 10-1
10.2 Groovy Template for a Custom Service Request ....................................................... 10-1
10.3 Groovy Template for a Custom Rule ............................................................................ 10-2
10.4 Groovy Template for Manipulating Output from a REST Connector Response .......... 10-3
10.5 Groovy Template for Manipulating Output from an Orchestration Response .......... 10-4
10.6 Additional Attributes and Methods Available in the Groovy Script Templates .......... 10-5

11 Testing Orchestrations
11.1 Understanding Run Orchestrations ............................................................................. 11-1
11.1.1 Using cURL to Simulate Testing from a Third-Party Application or IoT Device .... 11-2
11.2 Testing an Orchestration .............................................................................................. 11-2

12 Administering the Orchestrator Studio and Orchestrations
12.1 Understanding Orchestration Life Cycle Management ................................................. 12-1
12.2 Managing Orchestrator Studio Security ..................................................................... 12-2
12.3 Setting Up UDO Security for Orchestrator Studio Users ........................................... 12-3
12.3.1 UDO Feature Security for the Orchestrator Studio ............................................... 12-3
12.3.2 UDO Action Security for Orchestrator Studio Users ............................................ 12-3
12.3.3 UDO View Security for Orchestrator Studio Users .............................................. 12-4
12.4 Clearing Orchestration Cache on the AIS Server ....................................................... 12-4
12.4.1 Activating the AIS Administration Service ............................................................ 12-5
12.4.2 Using the AIS Administration Service to Clear Cache ......................................... 12-5
12.5 Setting Up Orchestrator Health and Exception Monitoring .................................... 12-6
12.5.1 Downloading and Installing the Orchestrator Monitor ....................................... 12-6
12.5.2 Enabling Orchestrator Health and Exception Tracking in Server Manager .......... 12-6
12.5.3 Set Up User Access to the Orchestrator Monitor and Orchestrator Health and Exceptions Program ............................................................. 12-7
12.5.4 Enable UDO View Security to Monitor Orchestrations, Notifications, and Schedules..... 12-8
13 Understanding the AIS Server Discovery Service

14 Open Standards for Orchestrations

14.1 Understanding Open Standards for Orchestrations .................................................. 14-1
14.2 Discovering Orchestrations Using Open Standards .............................................. 14-1
14.2.1 XML Inputs and Outputs .................................................................................. 14-2
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 release 9.2.4.2.

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.

**Documentation Accessibility**


**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](http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info) or visit [http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs](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](http://learnjde.com)
This guide references related information in the following guides:

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

Conventions

The following text conventions are used in this document:

<table>
<thead>
<tr>
<th>Convention</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>boldface</strong></td>
<td>Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.</td>
</tr>
<tr>
<td><em>italic</em></td>
<td>Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.</td>
</tr>
<tr>
<td><strong>monospace</strong></td>
<td>Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.</td>
</tr>
<tr>
<td>&gt; Tutorial</td>
<td>Indicates a link to a recording of the described feature. These recordings are in MP4 format so ensure that you have an appropriate player installed. Access to these recordings requires a valid Oracle account.</td>
</tr>
</tbody>
</table>
This chapter contains the following topics:

- Section 1.1, "JD Edwards EnterpriseOne Orchestrator Overview"
- Section 1.2, "How It Works"
- Section 1.3, "EnterpriseOne Architecture for Orchestrator"

1.1 JD Edwards EnterpriseOne Orchestrator Overview

The JD Edwards EnterpriseOne Orchestrator is 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.
How It Works

- 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.

1.1.1 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*.

1.1.2 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*.

1.2 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 illustration in Figure 1–1 shows how the EnterpriseOne Orchestration processes orchestrations. This illustration depicts an orchestration that includes rules, cross-references, form requests, and a connector to a notification.

Figure 1–1 EnterpriseOne Orchestration Processing

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."

1.3 EnterpriseOne Architecture for Orchestrator

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

For an illustration of the AIS Server architecture, see "AIS Server Architecture" in the JD Edwards EnterpriseOne Tools System Overview Guide.
This chapter describes how to access and get started with the Orchestrator Studio 9.2.4. It contains the following topics:

- Section 2.1, "Certifications (Formerly Known as Minimum Technical Requirements)"
- Section 2.2, "Prerequisites"
- Section 2.3, "Before You Begin"
- Section 2.4, "Accessing the Orchestrator Studio"
- Section 2.5, "Set Up a Temporary Directory on the AIS Server for File Transfers"

### 2.1 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](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](https://support.oracle.com/rs?type=doc&id=745831.1)

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

### 2.2 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 Security Administration Guide*.

- Ensure that the HTML Server is configured as a trusted node for the Security Server.


**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 Orchestrator Guide for Studio Version 8 and Prior.*
2.3 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.

2.4 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.

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

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

**See Also:**
- Configuring a REST Connector to Invoke a REST Service
- Configuring a REST Connector to Transfer Files to a REST Service
This chapter contains the following topics:

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

3.1 Understanding the Orchestration Design Process

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

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

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

You can use a simple worksheet for your analysis or you could use a storyboard, flow chart, or a combination of methods depending on the complexity of your orchestration. Use the information captured from your analysis to configure orchestrations in the Orchestrator Studio as described in.
3.2 Identifying the Problem and Solution

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

Example: Company A’s Problem and Solution

Problem

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

Solution

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

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

3.3 Identifying the Data for the Orchestration

After you identify the problem and determine the EnterpriseOne applications or processes that you want to invoke, you need to identify the device data or payload to
use in the orchestration. For example, a reading from a sensor might include a vibration measurement, a temperature measurement, and the date and time of the readings.

**Example: Company A's Data Analysis**
The "Input - Data" area in Figure 3–1 highlights the data that Company A identified for their orchestration.

### 3.4 Identifying the Rules for the Orchestration

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

You will use the information from this part of the analysis to create a rule for the orchestration as described in the Creating 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 Figure 3–1 shows the rules and conditions that Company A is using to determine which data should be processed by the orchestration.

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

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

Use the information from this part of the analysis to configure cross references for an orchestration as described in the Creating 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:

<table>
<thead>
<tr>
<th>Input</th>
<th>EnterpriseOne Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>sensor ID</td>
<td>equipment number</td>
</tr>
<tr>
<td>equipment number</td>
<td>warning recipient</td>
</tr>
</tbody>
</table>
The "Cross Reference" area in Figure 3–1 highlights the cross references Company A is using in the orchestration.

### 3.6 Identifying the Service Request Information for the Orchestration

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

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

Use the information gathered from this phase of the analysis to design the service request for the orchestration as described in the Creating a Component section in this guide.

**Example: Company A’s Service Request Analysis**
The "Service Request" area in Figure 3–1 highlights the data Company A is using for the service request.
Important: 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 Orchestrator Guide for Studio Version 8 and Prior.

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

- Section 4.1, "Understanding the Orchestrator Studio and Orchestrations"
- Section 4.2, "Navigating the Orchestrator Studio"
- Section 4.3, "Creating Service Requests"
- Section 4.4, "Working with Orchestrator Categories (Release 9.2.4.2)"
- Section 4.5, "Configuring a Form Request in the Orchestrator Studio"
- Section 4.6, "Configuring a Data Request"
- Section 4.7, "Configuring a Report Service Request"
- Section 4.8, "Configuring a Watchlist Service Request"
- Section 4.9, "Configuring a Message Request"
- Section 4.10, "Configuring a Connector Service Request"
- Section 4.11, "Configuring a Custom Service Request"
- Section 4.12, "Creating Connection Soft Coding Records for Connector Service Requests"
- Section 4.13, "Creating Rules"
- Section 4.14, "Creating Cross References"
- Section 4.15, "Creating White Lists"

4.1 Understanding the Orchestrator Studio and Orchestrations

The Orchestrator Studio is a web-based application for creating orchestrations and the components that comprise an orchestration. 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.

- **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.

- **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.

Figure 4–1 shows the list of the steps you can add to an orchestration. Each step in an orchestration represents one of the components listed above.
4.1.1 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.

4.1.2 Reusable Orchestration Components

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

To determine if a component is used by other orchestrations, 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.

4.1.3 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.
4.2 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.

Figure 4–2 Orchestrator Studio Home

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, 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.

- 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

4.2.1 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

### 4.2.2 User Defined Object (UDO) Features in the Orchestrator Studio

Orchestration components are saved and managed as UDOs in EnterpriseOne. The Orchestrator Studio includes UDO 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.

<table>
<thead>
<tr>
<th>UDO Options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Save As</td>
<td>Saves the orchestration component to a status of &quot;Personal.&quot; Components with a status of &quot;Personal&quot; are components that are being developed and have not been shared for publishing to the AIS Server.</td>
</tr>
<tr>
<td>Request to Publish</td>
<td>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 &quot;Pending Approval&quot; in the component list and then changes to &quot;Shared&quot; when the component is approved. If rejected, the status changes to &quot;Rework.&quot; At that point, you can edit the component and then use the Request to Publish button to send it for approval again.</td>
</tr>
<tr>
<td>Reserve</td>
<td>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 &quot;Reserved.&quot;</td>
</tr>
<tr>
<td>Unreserve</td>
<td>Cancels the reserved component. This action changes the status of the component back to &quot;Shared.&quot;</td>
</tr>
<tr>
<td>Notes</td>
<td>Available when the component is in the &quot;Pending Approval&quot; 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 &quot;Request to Publish&quot; option. This feature enables you to add an addendum to the original note.</td>
</tr>
</tbody>
</table>

4.2.3 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.

Figure 4–3 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.

  ![Figure 4–4 Start Node](image)

  - **Run Orchestration.** Takes you to the Run Orchestrations page to test only the current orchestration. You can click the Run Orchestrations icon on the Orchestrator Studio Home page to test all the other personal or shared orchestrations.

  - **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.

  - **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.

    ![Figure 4–5 Schedule](image)

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

- **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.
Navigating the Orchestrator Studio

Configuring Orchestration Components using Orchestrator Studio 9.2.4

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

Figure 4–6   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.

Figure 4–7  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.

- **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.
4.3 Creating Service Requests

This section contains the following topics:

- Understanding Service Requests
- Creating a Component

4.3.1 Understanding Service Requests

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

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

- Data Request
  
  Use a data request in an orchestration to query and 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 a local orchestration or an orchestration on another AIS Server, such as an AIS Server on another EnterpriseOne system in a multisite operation.
Also, you can configure a connector to use FTP or a REST call to transfer report output or other files to an external server.

■ Custom Request
Use custom Java or Groovy to execute a custom process.

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

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

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

4.3.2 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
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:

   1. **Name.** Enter a name for the component. Do not include special characters in the name.

   2. **Short description** field. Enter a description with a maximum of 200 characters. This description will appear below the component name in the component list.

   3. (Optional) Click **Long Description** to provide additional details about the purpose of the components.

   4. **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.

   __________

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

   __________

   **Notes:**

   - 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.

4.4 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.

4.4.1 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:
   - Category. Required. Enter a description of the category. This field is case-sensitive.
   - Category Type. Required. For orchestrations and its components, use ORCH.
   - 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.

### 4.5 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

**Recommended:**

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

### 4.5.1 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 modes ensure 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.

- **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 an form design aid (FDA) event that is executed only when an application is run through a form service request from the AIS server.

- **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.jpg)

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).

### 4.5.2 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.

- **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.

- **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.

  Do not use this action if the transaction involves adding rows to a grid.
4.5.3 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.

- **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:

  1. 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 retuned value and also remove a retuned value.
2. 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.

3. Click OK.

4.5.4 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.

4.5.5 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 EnterpriseOne Tool 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.

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.

4.5.5.1 Rules for Creating a Form Request in the Process Recorder

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

- Start the recording and then launch the first application from the carousel, Favorites, a link on an EnterpriseOne page, or Fast Path. Or you can open an
application and then start the recording before performing the first action in a form. You cannot start recording after performing an action in an EnterpriseOne application.

- If the process involves adding multiple records, perform only the steps to add a single record. Later, when you add the form request to an orchestration, you can use the Iterate Over feature in the Orchestrator Studio to configure the form request to add multiple records.

- Some business processes entail using forms in more than one application. You can record this as long as you access the other forms from a Form or Row menu. However, you cannot return to the EnterpriseOne home page and launch another application. If the business process requires launching a second application from the EnterpriseOne home page, then record it as a separate form request.

- When recording a process in the Process Recorder, each transaction is saved in EnterpriseOne just as it would be if you were not using the Process Recorder. To ensure that you do not sully production data, it is highly recommended that you use the Process Recorder in an EnterpriseOne test environment. If recording in a production environment, delete any transactions that were created while using the Process Recorder.

### 4.5.5.2 Prerequisites

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

- UDO feature security.
  
  A system administrator must enable the RECORDER feature in UDO feature security. See "Managing UDO Feature Security" in the *JD Edwards EnterpriseOne Tools Security Administration Guide*.

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

### 4.5.5.3 Using the Process Recorder to Create a Form Request

1. On the EnterpriseOne home page, click the user drop-down menu in the upper right corner and select **Record a Process**.

2. In the **Process Recorder** pop-up box, click **Start** to start the recording.

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

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

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

   When you enter a value in a field, the Process Recorder automatically records the field name as an input and the data that you enter as a default value. Later, you can change the name of the input and delete or change the default value by editing the form request in Orchestrator Studio.
Important: If you return to the EnterpriseOne home page or try launching a separate application from the Carousel while recording, the Process Recorder pauses the recording and will not record those steps. To resume the recording, click the Paused button and the Process Recorder will resume recording, returning you to the point in the process in which you initially paused the recording.

5. 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.

6. After performing the final step, click Stop.
   If you stopped the recording prematurely, click Cancel, and then continue to record the remaining actions. To discard the process and start over, click Discard.

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

8. Click Save.
   If you need to modify the form request, see Configuring a Form Request in the Orchestrator Studio. To add it to an orchestration, see Adding Steps to an Orchestration.

4.6 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

4.6.1 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.

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

### 4.6.2 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.

**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:
   - 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.

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.

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.
4.6.3 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:
   1. 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.
   2. 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.
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:

- **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.

- **Group By** icon. Click this icon next to any field that you want the aggregate results grouped by. In the "Group By and Variable Name" section on the right, you can enter a variable name.

  Using 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.

- **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.

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.

### 4.6.4 Viewing and Copying JSON Code of a Data Request

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

### 4.6.5 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.

### 4.6.5.1 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.

### 4.6.5.2 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.

**Important:** 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.

- **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.

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

<table>
<thead>
<tr>
<th>Max Records</th>
<th>Page Size</th>
<th>Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 - All records</td>
<td>0 - Paging is off</td>
<td>System will attempt to get all records in one request.</td>
</tr>
<tr>
<td>0 - All records</td>
<td>&gt;0 - Size of page defined</td>
<td>System will attempt to get records in sections for the defined page size until it gets all records.</td>
</tr>
<tr>
<td>&gt;0 - Set number of records</td>
<td>0 - Paging is off</td>
<td>System will attempt to get the max number of records defined in one request.</td>
</tr>
<tr>
<td>&gt;0 - Set number of records</td>
<td>&gt;0 - Size of page defined</td>
<td>System will attempt to get records in sections for the defined page size until it gets to the maximum number of records defined.</td>
</tr>
</tbody>
</table>

- **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.
4.7 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.

To configure a report service request:

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. 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.

4. 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.

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

   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.

   Set the Fire and Forget toggle to Off 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
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)

4.9 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.

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:

   - 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.
- 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.

- 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: ${tovariable}, ${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*.

- Select **Email** to enter an email address for the recipient.

- 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 8.

5. To include boilerplate text from a message template in the data dictionary:
   a. Expand the **Data Dictionary** section.
   b. 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.

6. To include a link to an application:
   a. In the **Application Links** section, click Add and expand the section.
   b. Complete the **Application**, **Form**, and **Version** fields to specify the form that you want the application link to launch.
   c. 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.
   d. Use the **Pre Text**, **Link Text**, and **Post Text** fields to define the text that appears before, as, and after the link, respectively, in the message.
   e. In the grid, you can use variables to pass data to the application when the application is launched from the shortcut.
   f. Click the Add button in the Application Links section and repeat these steps to include multiple application links in a message.
   g. Click the Remove button (X) at the end of the New Shortcut section header to delete the application link that is added.
7. To include other links (for example, to launch an orchestration or notification):
   a. In the Other Links section, click Add and expand the section.
   b. Select the type of link you would like to add from the Type drop-down menu. Valid values are Orchestration, Notification, or URL.
   c. Depending on the type you have selected, either select the orchestration name, notification name, or enter the URL.
   d. In the Link Text field, enter the text you would like to appear as link in the message.
      This link text appears in the notification message.
   e. In the Pre Text and Post Text fields, enter the text you want to appear before and after the link text.
   f. 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.
   g. 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.

8. To include report output as an attachment:
   a. In the Attachments section, click Add and expand the section.
   b. 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.
   c. 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.
   d. 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.

9. 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}

10. Click the Preview button to preview the message. The preview dialog displays the subject and body of the message including the data dictionary boilerplate text, links to applications, URLs, orchestrations, notifications, and reports.

11. Click Save to save your changes.
4.10 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 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)

4.10.1 Understanding Connector Service Requests

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

- Invoke another orchestration or invoke a notification either on your local system or on an AIS Server on another EnterpriseOne system in a multisite operation.
- Invoke a REST service. Referred to as a REST connector, this 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.
4.10.2 Before You Begin

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

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

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 an connection to the AIS server from the Orchestration drop-down list.
   
   After you select an 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 the notifications or orchestrations list. 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 Allow Input JSON Object toggle if you want to map a JSON object to the orchestration connector. This action enables you to pass arrays to an orchestration that is called through a connector. You can view the format of the JSON object to be passed to this variable by navigating to the Run Orchestrations page and then viewing the sample input for the selected orchestration.
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.
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.

### 4.10.4 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 scripting to refine the data in the connector output.

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 and Parameters sections are automatically filled with the appropriate values.
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.
8. In the **Output** section, use the following sections to identify the outputs:

---

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

---

- **Manipulate Output** (advanced). Use Groovy scripting to 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](#)
Configuring a Connector Service Request

for more information. Click the Show Shortcut Command icon (?) to view the commands to edit the script.

- **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 ([]).

9. To test the connector:
   1. Click the **Test** button.
   2. If the connector includes parameters, enter values for the parameters in the pop-up box.
      
      Clicking **Execute** displays the REST service response.
   3. 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.

---

4.10.5 Configuring a REST Connector to Invoke a REST Service

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

- The path to a particular endpoint in the REST service.
- Parameters, if required by the endpoint.
- Key-value pairs for the HTTP header.

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

To configure a connector to invoke a REST service:

1. 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.
7. 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.

8. 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 ${}.

9. 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.

10. 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.

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

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

   ■ Manipulate Output (advanced). Use Groovy scripting to manipulate the output of the REST call. For example, if you only need certain data in the output, you can use Groovy scripting to refine the data displayed in the output. See Groovy Template for Manipulating Output from a REST Connector Response for more information.

   ■ Output grid. If the value you want is nested in JSON objects, you can get to the value by separating each JSON object name by a period (.). If the value you want is in an array, you can access the value by adding the proper index of the array inside brackets ([]).

12. 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 the Groovy script if specified.

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

4.10.6 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.

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

You can configure a REST connector to transfer files to a REST service. When added to an orchestration that includes a report service request, a REST connector can transfer the output of the report service request to an external system. Also, if an orchestration includes a custom service request that uses a Groovy script to generate and save a file to the AIS Server, you can create a REST connector to transfer the generated file to an external system.

1. 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.

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:
   - **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.
   - **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.
   - **File Type**: Select the file type of the output.

   For File, complete these fields:
   - **Source File Name**: Enter the name of the file that was created in the custom Groovy service request.
4.10.8 Configuring a Database Connector

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

1. 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 Groovy script template. 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. You can enter default values or include variables using the ${} notation, where you enter the variable name within the braces.
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.

4.10.9 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.

- **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.
- Click the Use Temporary File Location check box to save the file to the temporary directory on the AIS Server.

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.
- 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.

**4.10.9.1 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**.

### 4.10.9.2 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**.

### 4.10.9.3 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 Location** 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**.

### 4.10.9.4 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 Orchestration 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. Click the **Import CSV as Array** check box,
   
   The Orchestration Studio displays additional fields to configure the importing of data from a CSV file.

8. If the CSV file has a header row, click the **CSV Has Headers** check box to exclude the header row.
   
   This action ensures that at runtime, the Orchestration will not import the column headers as data in the array.

9. Complete the following fields:
   
   - **Delimiter**. Enter the delimiter used in the CSV file to separate values.
   - **Source File Name**. Enter the name of the CSV file.
4.11 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)

4.11.1 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.

---

- **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.

- **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.

- **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.

10. 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.

11. Click Save.
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:
   
   - **Input**: Enter the name of the field in the custom Java class.
   - **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.
   - **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.

### 4.11.2 Configuring a Custom Service Request with Groovy (Advanced)

You can create a service request that uses Groovy scripting to execute a custom process or to route data into another database. To use Groovy for a service request, the Orchestrator Studio provides an edit area that contains a sample Groovy script with instructions on how to modify the script. You can click the Show Shortcut Command icon located above the edit area to view the commands to work with the script.

Chapter 9, "Using Apache Groovy for Custom Service Requests, Rules, and Manipulating Output" provides information on how to work with the sample Groovy script.

To configure a customer 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 and select Groovy.

3. Create and name the custom request as described in Creating a Component.

4. On the Custom Requests design page, configure the script to perform the desired action.

5. 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.

6. 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.
7. (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.

8. 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.

9. 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.

### 4.12 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

### 4.12.1 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.

### 4.12.2 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:
Creating Connection Soft Coding Records for Connector Service Requests

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

**IMPORTANT:** Oracle strongly recommends that all external calls use the SSL protocol (https) with a certificate from a reputable certificate authority.

5. On the **Security** tab, enter the credentials for accessing the external system.
   - **Security Policy**
     Select the authentication type 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.** Slide the toggle to the right to enable the proxy.

7. Click **Save.**

### 4.12.3 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 Endpoint field, enter the URL for accessing the Open API - 2.0 documentation through which you can access the Server Manager REST services.
For example: https://<server_manager_console_host:port/ 
manage/mgmtrestservice/v1/open-api-catalog>

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

5. 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**

6. 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.

7. Click **Save**.

**Notes:**
- 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.

4.12.4 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.

**IMPORTANT:** 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.
     
     1. In the Token Endpoint URL field, enter the address of the server that will provide the OAuth token.
     
     2. In the OAUTH Client ID field, enter ID of the client.
     
     3. In the OAUTH Secret field, enter the value for client secret you have received after registering the application.
     
     4. If the client requires additional header parameters, in the OAUTH Parameters area, click Add.
        
        The system adds a new row in the OAUTH Parameters table.
        
        5. Enter a name and value for the parameter in the table.

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.** 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.**
4.12.5 Creating a Soft Coding Record for a Database Connection

**Important:** For a database connection to work, the JDBC driver for the database must be accessible in the AIS Server 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.

   **IMPORTANT:** Oracle strongly recommends that all external calls use the SSL protocol (https) with a certificate from a reputable certificate authority.

   - User. Enter the database user.
   - Password. Enter the database user password.
   - Driver. Enter the driver for the type of database you are accessing. You need to make sure that the driver is specified on the AIS Server class path.
5. Click Save.

4.12.6 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 .ppk 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 .ppk file in OpenSSH format into the box to upload the file.

5. Click **Save**.

### 4.13 Creating Rules

This section contains the following topics:

- **Section 4.13.1, "Understanding Rules"**
- **Section 4.13.2, "Creating a Rule"**
- **Section 4.13.3, "Creating a Custom Rule with Java (Advanced)"**
- **Section 4.13.4, "Creating a Custom Rule with Groovy (Advanced)"**

#### 4.13.1 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.

4.13.2 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.
   - Literal. Slide the toggle to the right to indicate a literal value.
   - Value 2. If you have enabled the Literal toggle, manually enter a value.
   - 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.

4.13.3 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:
   - Input. Enter the name of the field in the class.
   - 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.
   - 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.

4.13.4 Creating a Custom Rule with Groovy (Advanced)

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

To create a custom rule with Groovy:

1. 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.
4.14 Creating Cross References

This section contains the following topics:

- Section 4.14.1, "Understanding Cross References"
- Section 4.14.2, "Creating a Cross Reference"

4.14.1 Understanding Cross References

The Orchestrator uses a cross-reference component to map incoming data (third-party data) to an EnterpriseOne value. The EnterpriseOne value 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.

4.14.2 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. Complete the Input Key and Output Key columns to map the inputs to EnterpriseOne fields:
   - **Input Key.** The inputs can be one or many values. The inputs must correspond to the 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.
   - **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.
6. If you enter any value by mistake, you can click the Remove (X) button to delete the entry.
7. 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
4.15 Creating White Lists

This section contains the following topics:

- Section 4.15.1, "Understanding White Lists"
- Section 4.15, "Creating White Lists"

4.15.1 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.

4.15.2 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 White Lists
Creating Orchestrations with Orchestrator Studio 9.2.4

This chapter contains the following topics:

- Section 5.1, "Creating Orchestrations"
- Section 5.2, "Creating Schedules for Orchestrations"
- Section 5.3, "Working with Scheduler"
- Section 5.4, "Updating to Version 3 Orchestrations"
- Section 5.5, "Restoring Orchestrations and Orchestration Components"
- Section 5.6, "Supported Input Message Formats"
- Section 5.7, "Orchestration Security Considerations"
- Section 5.8, "Exporting Orchestration Components from the Orchestrator Studio"
- Section 5.9, "Importing Orchestration Files in the Orchestrator Studio"

5.1 Creating Orchestrations

This section contains the following topics:

- Understanding Orchestrations
- Creating an Orchestration
- Adding Steps to an Orchestration
- Adding Inputs to an Orchestration
- Mapping Orchestration Inputs
- Retrieving and Passing Data Sets in an Orchestration
- Working with Orchestration Output

5.1.1 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, or service request component.

**Important:** Remember that when you are ready to "request to publish" an orchestration, you need to make sure that you also request to publish all 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.

### 5.1.2 Creating an Orchestration

To create an orchestration:

1. On the Orchestrator Studio Home page, click the **Orchestrations** icon.
2. On the Orchestration side panel, click the **New** button.
3. On the Orchestration 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.

6. 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:

7. 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.

8. 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.

### 5.1.3 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.

#### 5.1.3.1 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.
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:
5.1.3.2 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:

   ![Orchestration Diagram](image)

   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*.

5.1.3.3 Removing a Step from an Orchestration

---

**Caution:** Before you delete a step, make sure the step is not used by any other component in the orchestration.
1. 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.

5.1.3.4 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.

5.1.3.5 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.

5.1.3.6 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:
   - Abort
   - 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.
   b. Map the inputs as appropriate.

### 5.1.4 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 Address Book Number, User Name, System Date, and Orchestration Input. 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:
   - JDE Standard
   - Generic

   See **Supported Input Message Formats** for more information about which input format to use.
4. In the **Orchestration Inputs** tab of the Inputs and Values dialog box, enter the name of the input in the Input column.
5. In the **Value Type** column, select the input value type. Valid values are:
■ 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.

<table>
<thead>
<tr>
<th>Inputs and Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input Format</td>
</tr>
<tr>
<td>Orchestration Inputs</td>
</tr>
<tr>
<td>GridData</td>
</tr>
<tr>
<td>Update Row</td>
</tr>
</tbody>
</table>

■ String
■ Numeric
■ Boolean
■ 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:

■ dd/MM/yyyy
■ dd/MM/yy
■ yyyy/MM/dd
■ MM/dd/yyyy
■ MM/dd/yy
■ yy/MM/dd

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

■ Milliseconds
■ yyyy-MM-dd'T'HH:mm:ss.SSSZ

6. Optionally, use the Default Value column to enter a default value that you want to pass through the orchestration input.

7. 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.

### 5.1.5 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.

**Figure 5–2 Transformations Example**

![Transformations Grid](image)

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.

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.

5.1.6 Retrieving and Passing Data Sets in an Orchestration

You can configure a form request or a data request to return a data set. You can also configure an orchestration to pass a data set 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.
5.1.6.1 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.

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

For more information about configuring a data request, see Configuring a Data Request.

5.1.6.2 Configuring a Form Request to Retrieve a Data Set

---

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

---

1. In the Form 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.
For more information about configuring a form request, see Configuring a Form Request in the Orchestrator Studio.

### 5.1.6.3 Passing a Data Set to a Subsequent Orchestration Step

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

1. Add the form request or data request defined with a data set to the orchestration.
2. In the 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.

### 5.1.7 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 Groovy scripting to manipulate the contents of the response to refine the orchestration output as required by the parameters in the consuming device or program.

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 Orchestration Outputs icon.

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

   ![Orchestration Outputs Table]

3. 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.

4. You can edit the provided Groovy script template in the **Manipulate** tab to manipulate the contents of the response to refine the orchestration output.

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

5. To save the orchestration outputs, return to the Orchestrations design page and click **Save**.

### 5.2 Creating Schedules for Orchestrations

This section contains the following topics:

- Understanding Schedules
- Creating a Schedule
- Adding a Schedule to an Orchestration

#### 5.2.1 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.

#### 5.2.2 Creating a Schedule

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

5.2.3 Adding a Schedule to an Orchestration

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

**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.
5.3 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

5.3.1 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.

5.3.2 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 Application Interface Services Server Reference Guide*.

### 5.3.3 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 Security 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 Security Administration Guide*.

### 5.3.4 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.

### 5.3.5 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.
Figure 5–4  Jobs Stopped

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.

Figure 5–5  User, Role, and Environment

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
- Consecutive Errors
- % Errors
- 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.

*Figure 5–6 Info*

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.
5.3.6 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, 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 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.

5.4 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.


5.5 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 Orchestration 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.

5.6 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": "142774400000"
   },

   {
      "name": "time",
      "value": "12:15:15"
   },

   {
      "name": "temperature",
      "value": "99"
   }
   
   ]
}
```

Generic Input Message Format
5.6.1 Additional Supported Input Message Formats for the Generic Input Format

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

```
{
    "equipmentNumber": "41419",
    "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
```

5.6.2 Differences Between Input Message Formats

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

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

As shown in 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"
        }
    ]
}```
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:

```json
{
   "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.

5.7.1 Restricting Access to Exposed Orchestrations

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

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

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

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

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

5.8 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. The Orchestrator Studio exports the source XML files in a zip file.

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.

5.8.1 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 ( _ ).

5.9 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 or a zip file containing XML files.

**Caution:** You cannot import orchestration component files that were exported from the EnterpriseOne Object Management Workbench - Web application. The zip file 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:

![Folder Structure Example]

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** icon.
2. On the Import page, click the **Upload** icon.
3. Locate the orchestration component XML files or zip file that contains the orchestration component files that you want to import.

   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.

- 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.

- 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.
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.

- 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.
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.

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.
This chapter contains the following topics:

- Section 7.1, "Understanding the EnterpriseOne Orchestrator Monitor"
- Section 7.2, "Prerequisites"
- Section 7.3, "Accessing the Orchestrator Monitor"
- Section 7.4, "Monitoring UDOs Based on Different EnterpriseOne User, Environment, or Role"
- Section 7.5, "Refreshing the Data Displayed in the Orchestrator Monitor"
- Section 7.6, "Resetting the Data Displayed in the Orchestrator Monitor"
- Section 7.7, "Monitoring Orchestrator Health"
- Section 7.8, "Monitoring Orchestrator Exceptions"
- Section 7.9, "Managing Orchestrator Health and Exception Records in EnterpriseOne"

7.1 Understanding the EnterpriseOne Orchestrator Monitor

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

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

- Orchestrations
- Notifications
- Schedules

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

Orchestrator Health Pane in My Worklist

Oracle provides a downloadable My Worklist composed page, which includes a Watchlist pane, Message Center, and the Orchestrator Health pane. The Orchestrator Health pane in My Worklist provides another way to monitor the health of your orchestrations and notifications without having to access the Orchestrator Monitor. However, it does not provide exception information. For information on how to
download and access My Worklist, see "My Worklist" in the *JD Edwards EnterpriseOne Tools Foundation Guide*.

### 7.2 Prerequisites

Before users can monitor UDOs, an administrator must install the Orchestrator Monitor and enable Orchestrator health and exception tracking in Server Manager. See Setting Up Orchestrator Health and Exception Monitoring.

### 7.3 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.

### 7.4 Monitoring UDOs Based on Different EnterpriseOne User, Environment, or Role

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

To change the UDOs displayed in the Orchestrator Monitor:

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

### 7.5 Refreshing the Data Displayed in the Orchestrator Monitor

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

- Press F5.
- Switch between the Health tab and the Exceptions tab.
- Switch between the Exception Chart view and the Exceptions List view.
- After you enter a custom date range and click *Search*.
- Change your search criteria or filter results.
After you use the additional search field and click **Search**.

Click or expand an exception record.

### 7.6 Resetting the Data Displayed in the Orchestrator Monitor

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

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

See *Managing Orchestrator Health and Exception Records in EnterpriseOne* for information on how to view and delete health and exception records in EnterpriseOne.

### 7.7 Monitoring Orchestrator Health

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

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

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

#### 7.7.1 Filtering the UDOs Displayed in the Health Tab

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

#### 7.7.2 Understanding Orchestrator Health Data

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

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

**High-Level Health Information**

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

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

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

### 7.8 Monitoring Orchestrator Exceptions

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

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

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

#### 7.8.1 Possible Causes of Orchestrator Exceptions

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

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

For schedules, the following circumstances can generate an exception:

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

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

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

To change the search criteria:

1. Click the Select Range drop-down list and select a preset range.
   
   You can also select Custom Range from the list and click the date fields to manually set the date and time for the range.
   
2. Click Search.
   
3. You can enter a value in the "additional search criteria" field to display exception records based on Name, Exception, UDO Name, Host Address, or Host Name. This search is case sensitive.
   
4. To display exceptions for a particular UDO type, click the Type drop-down list and select Orchestration, Notification, or Schedule. Or select All to view all exceptions.

   The Orchestrator Monitor automatically refreshes the list based on the search criteria.

To filter over the search results:

In the Filter Results field, enter a value to filter on any of the values in the exception record header (information in the collapsed exception record). For example, you can enter the name of a UDO, a status code, user, or other values displayed in the exception record header. The filter criteria is not case sensitive.

You cannot use the Filter Results field to filter on exception record details.

7.8.3 Reviewing General Exception Information in the List View

The List view displays the following general information about orchestration, notification, and schedule exceptions:

- Request Received
  
  The date and time the request to the orchestration, notification, or schedule on the AIS Server was received.

- Name
  
  The name of the UDO. The icon next to it indicates if it is an orchestration , notification , or schedule.

- Exception
  
  The name and details of the exception.

- Status Code
  
  The HTTP status that was returned when the orchestration or notification was called.

- User
The user who invoked the orchestration or notification. In the case of a scheduled object, the user who initiated the schedule start.

- Environment
  This is the environment that the user who invoked the UDO signed in to. In other words, it is the environment in which the UDO was executed. To view the performance of UDOs deployed to a particular environment, see Monitoring UDOs Based on Different EnterpriseOne User, Environment, or Role for more information.

- Product Code
  The product code associated with the UDO when it was created.

### 7.8.4 Reviewing Detailed Exception Information in the List View

Expand an exception row to find additional details about the exception, which include:

- UDO Name
  The ID of the UDO in EnterpriseOne.

- URL
  The URL to the orchestration or notification on the AIS Server.

- Host Address
  The IP address of the requesting host, that is, the IP address of the machine that made the request to the AIS Server. If invoked by the scheduler, this is the IP address of the AIS Server.

- Host Name
  The name of the host requesting the service. If invoked by the scheduler, this is the name of the AIS Server.

- Role
  The role of the user who originated the request to the service (orchestration or notification).

- Processing Time
  The processing time before the exception occurred.

- Schedule Name
  If the service was invoked by the scheduler, the UDO ID of the schedule (for example SCH_1807120003CUST) used by the scheduler is displayed here.

- HTTP Method
  The HTTP method used to invoke the service.

- Step Trace (orchestrations only)
  If step trace is enabled for an orchestration, click this button to view details about each step in the orchestration. See Understanding Orchestration Step Trace Details for more information.

- Input JSON (orchestrations and notifications only)
  Click this button to view the JSON input that was sent to this UDO when invoked.

- Trouble Shooting (schedules only)
Click this button to view additional details to help with troubleshooting the issue.

- Exception Message
  Click this button to view the exception message.

- Application Errors
  If the exception occurred as a result of an EnterpriseOne application error, click this button to view the details of the error, which include errors displayed on the EnterpriseOne form that was executed.

- Group (icon)
  If an exception occurs with an orchestration or notification that calls or is called by another orchestration or notification UDO, click the Group icon to view a list of all associated UDOs that had exceptions during the execution of that request.

- Object Name
  For notifications that call an orchestration, when the failure is due to the orchestration, this field displays the orchestration object name.

### 7.8.5 Understanding Orchestration Step Trace Details

If an orchestration configured for step tracing generates an exception, the Orchestrator Monitor includes step trace details in the exception record.

Step trace details include the orchestration or notification name, inputs and outputs, the start time and end time, duration, the status, and the orchestration response.

For an orchestration exception, the step trace lists all successfully completed steps of an orchestration. The step trace details do not include the step that failed. If the orchestration calls another orchestration, it lists the successfully completed steps in the called orchestration. When you know the last step that completed successfully, you can open the orchestration in the Orchestrator Studio and identify the step that follows to identify where the exception occurred.

---

**Note:** An administrator can access the same step trace details through the AIS Server log file in Server Manager.

---

### 7.8.6 Viewing Exceptions in the Chart View

Use the Chart view to display exception data visually in a chart. The bottom of the page provides options to display the data in a pie chart or a vertical, horizontal, stacked, or unstacked bar chart.

In the Chart view, you can display information by UDO name, exception, or application error. The key to the right of the chart changes to identify the data represented in the chart. Figure 7–1 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.

7.8.7 Changing How Exceptions Are Displayed in the Chart View

In the Chart view, you can select a different date range, change how you want the data grouped by, or display exceptions for a particular UDO type.

To change the exception information displayed in the Chart view:

1. Click the Select Range drop-down list and select a preset range.
   You can also select Custom Range from the list and click the date fields to manually set the date and time for the range.

2. Click Search.

3. Use the two Group By fields to display data with a combination of Name (UDO name), Exception, or Application Error and Day, Month, or Year.

4. In addition to the date range, you can enter a value in the "additional search criteria" field to display exception data based on these exception record details: Name, Exception, UDO Name, Host Address, or Host Name.

5. To display data based on a particular UDO type, click the Type drop-down list and select Orchestration, Notification, or Schedule. Or select All to view data for all UDOs.

   The chart automatically refreshes based on your selection.
7.8.8 Accessing Detailed Information from the Chart

Click any bar or section in a chart to access more detailed information about that selection. For example, if you click a bar representing the number of exceptions for a particular UDO in one month, the Orchestrator Monitor displays details about exceptions that occurred for that UDO in that month. Descriptions of the details provided can be found in the Reviewing Detailed Exception Information in the List View section.

7.9 Managing Orchestrator Health and Exception Records in EnterpriseOne

Important: Before managing health and exception records in EnterpriseOne, make sure an administrator has enabled access to the Orchestrator Health and Orchestrator Exceptions program (P980060). See 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.

7.9.1 Managing Health Records in EnterpriseOne

To view health records in EnterpriseOne:

1. From the EnterpriseOne Navigator, select EnterpriseOne Life Cycle Tools, Orchestrator Management, Orchestrator Health (P980060_W980060B).
2. Click Find to load all records.
3. You can also search for health records based on values that you enter in the following fields:
7.9.2 Managing Exception Records in EnterpriseOne

To view exception records, you must have UDO view security enabled for the orchestration, notification, and schedule UDOs that you want to view.

To view exception records in EnterpriseOne:

1. From the EnterpriseOne Navigator, select EnterpriseOne Life Cycle Tools, Orchestrator Management, Orchestrator Exceptions (P980060_W980060A).
2. Click Find to load.

The Orchestrator Exceptions form displays exception records only for the UDOs to which you have UDO view security access.

3. You can also search for exception records based on values that you enter in the following fields:
   
   - **Object Type**
     To find records based on a particular UDO type, enter ORCH (for orchestrations), or NTF (for notifications). EnterpriseOne does not monitor or store health information for schedules.
   
   - **UDO Name**
     This is the object ID of the UDO in EnterpriseOne. Before filtering results, load all records and then look in the "UDO Name" column to identify this value.
   
   - **Product Code**
     Enter a product code to search for UDOs associated with a particular product code.

To delete exception records:

1. In the Orchestrator Exceptions form, click the check box next to the records you want to delete.
2. Click Delete.
8.1 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

8.1.1 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.

8.1.2 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.

8.1.3 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.

For detailed information about creating an orchestration, see Creating Orchestrations.
Creating Custom Java for Orchestrations

This chapter contains the following topics:

- Section 9.1, "Understanding Custom Java for Orchestrations"
- Section 9.2, "Creating Custom Java"
- Section 9.3, "Deploying Custom Java"

9.1 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.

9.2 Creating Custom Java

When creating custom Java classes, you must reference these JAR files, which are dependencies:

- OrchestratorCustomJava.jar. This contains the definition of the interfaces.
- AIS_Client.jar 1.1.0 or higher. This contains the loginEnvironment attribute and enables AIS calls to JD Edwards EnterpriseOne.

It is not necessary to include these JARs with the deployment as they are already deployed as part of the AIS Server deployment. After creating a custom Java class, you deploy the Java class or Java classes to a JAR file.

A custom rule should implement the CustomRuleInterface class included with the OrchestratorCustomJava.jar. The interface requires a loginEnvironment variable of type com.oracle.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.
9.2.1 Using Custom Java for the Orchestration Service Request

Custom service request Java classes should implement the com.oracle.e1.rest.orchestrator.customjava.CustomServiceRequestInterface which requires the following methods:

- `setLoginEnvironment(com.oracle.e1.aisclient.LoginEnvironment loginEnvironment)`. The method used to perform AIS calls.
- `process()`. The method that returns javax.ws.rs.core.Response which is called automatically after all the attributes are set.

9.2.2 Using Custom Java for the Orchestration Rule

Custom rule Java classes should implement the com.oracle.e1.rest.orchestrator.customjava.CustomRuleInterface which requires the following methods:

- `setLoginEnvironment(com.oracle.e1.aisclient.LoginEnvironment loginEnvironment)`. The method used to perform AIS calls.
- `evaluate()`. The method that returns Boolean and is called automatically after all the attributes are set.

9.3 Deploying Custom Java

You must deploy the JAR file that contains the custom Java to the AIS Server. Follow the instructions accordingly depending on the server on which the AIS Server is installed:

- Deploying Custom Java on AIS Server on Oracle WebLogic Server
- Deploying Custom Java on AIS Server on IBM WebSphere Application Server

9.3.1 Deploying Custom Java on AIS Server on Oracle WebLogic Server

To deploy the custom Java JAR file to an AIS Server on Oracle WebLogic Server:

1. Deploy the JAR as a shared library on the same WebLogic Server on which the AIS Server (otherwise referred to as the JDERestProxy) is deployed.
2. Restart the AIS Server using Server Manager.
3. Edit the weblogic.xml inside the JDERestProxy.war/WEB-INF to reference the custom java shared library, for example:

```xml
<?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.

### 9.3.2 Deploying Custom Java on AIS Server on IBM WebSphere Application Server

To deploy the custom Java JAR file to the AIS Server on Websphere Application Server:

1. Add the JAR as a shared library:
   a. On WebSphere, expand Environment and select **Shared Libraries**.
   b. In ClassPath, add the path to the JAR location on the server.

2. Associate the shared library with the JDERestProxy application:
   a. In the left pane, expand **Applications**, **Application Types**, and then select **WebSphere enterprise applications**.
   b. Select the appropriate AIS deployment.
   c. Under References, select **Shared library references**.
d. Select the JDERestProxy check box and then select the Reference shared libraries button.

e. Use the directional arrow to move the JAR file to the Selected group.

f. Click OK.

3. Synchronize Server Manager with the deployed application:

   Saving the configuration in Websphere will redeploy the application, but you must synchronize Server Manager to recognize the deployed application.

   a. In Server Manager, locate the AIS Server and update a setting in the Configuration section. This is required so that Server Manager detects a change in the AIS Server when you click the Synchronize Configuration button.

   b. Apply the changes and then return to the AIS Server home page.

   c. Click the Synchronize Configuration button to restart the AIS Server.
This chapter contains the following topics:

- Section 10.1, "Understanding Groovy for Orchestration Components"
- Section 10.2, "Groovy Template for a Custom Service Request"
- Section 10.3, "Groovy Template for a Custom Rule"
- Section 10.4, "Groovy Template for Manipulating Output from a REST Connector Response"
- Section 10.5, "Groovy Template for Manipulating Output from an Orchestration Response"
- Section 10.6, "Additional Attributes and Methods Available in the Groovy Script Templates"

10.1 Understanding Groovy for Orchestration Components

In the Orchestration Studio, programmers can use Apache Groovy, a scripting language for the Java platform, to extend the functionality of orchestrations. In the Orchestration Studio, you can use Groovy to:

- Create a custom service request that uses complex logic to perform transactions that cannot be configured in a standard service request.

- Create a custom rule with complex conditions that cannot be configured in a standard rule.

- Manipulate output from a REST connector response. For example, if an orchestration with a REST connector service request returns information in XML format, you can convert it to JSON for output mapping.

- Manipulate the output from an orchestration response.

Using Groovy scripts has two advantages over using custom Java. First, the Orchestration Studio provides an editing window for creating the Groovy script, so you do not have to use an external editor to create and test your custom program. The editing window contains a Groovy template that you can use to help get you started. Second, the Orchestration includes the Groovy script natively as part of the orchestration, so you do not have to deploy a custom program like you do with custom Java. The Groovy script simply executes as part of the orchestration.
10.2 Groovy Template for a Custom Service Request

In the Orchestrator Studio, the Custom Service Request design page contains a Groovy template that you can use to create a custom service request.

Figure 10–1 Groovy Code Template for a Custom Service Request

```groovy
import groovy.transform.CompileStatic;
import com.oracle.ei.common.OrchestrationAttributes;
import java.text.SimpleDateFormat;

HashMap<String, Object> main(OrchestrationAttributes orchAttr, HashMap inputMap)
{
    HashMap<String, Object> returnMap = new HashMap<String, Object>();
    // Add logic here
    // String stringVal = (String)inputMap.get("inputStringVal");
    // BigDecimal numVal = new BigDecimal((String)inputMap.get("inputNumVal");
    // SimpleDateFormat format = new SimpleDateFormat(orchAttr.getSimpleDateFormat());
    // Date dateVal = format.parse((String)inputMap.get("inputDateVal");
    // orchAttr.writeWarn("custom log entry - warning");
    // orchAttr.writeDebug("custom log entry - debug");
    returnMap.put("Limit Status", (String)inputMap.get("InputStringVal");
}
return returnMap;
```

The following list describes the highlighted sections in the preceding code:

1. Add or remove import lines as necessary.
2. Do not modify the function definition or script definition.
3. Copy and paste the code in the commented lines and use them in your script. The code lines include:
   - Parameters for defining a string, numeric value, date format, and date value.
   - Parameters for including a warning or log statements in the AIS Server log, which can be used for debugging script issues.
   - Parameter for populating the returnMap for outputs.

The Groovy template also includes an orchAttr that enables you to include additional information in the Groovy script. See Additional Attributes and Methods Available in the Groovy Script Templates.

10.3 Groovy Template for a Custom Rule

In the Orchestrator Studio, the Custom Rule Request design page provides the following code template to help you create a custom rule using Groovy:
10.4 Groovy Template for Manipulating Output from a REST Connector Response

In the Orchestrator Studio, the Connector design page provides a Groovy code template that you can use to manipulate the output from a REST connector response.

The response can contain output in XML, which you can convert to JSON using Groovy. For example, your orchestration might include a step with a REST connector to a third-party system, and its response may return in XML. You could write a Groovy script to reformat that response in JSON.
Figure 10–3  Groovy Code Template for Manipulating REST Connector Output

```java
import groovy.transform.CompileStatic;
import groovy.json.JsonSlurper;
import groovy.json.JsonBuilder;
import com.oracle.json.el.CommonOrchestrationAttributes;

@CompileStatic
String main(OrchestrationAttributes orchAttr, String input) {
    def jsonIn = new JsonSlurper().parseText(input);
    // modify jsonIn
    def jsonOut = new JsonBuilder(jsonIn).toString();
    // orchAttr.writeWarn("custom log entry - warning");
    // orchAttr.writeDebug("custom log entry - debug");
    return jsonOut;
}
```

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.

10.5 Groovy Template for Manipulating Output from an Orchestration Response

The Orchestration Outputs design page provides a Groovy template that you can use
to:
- Refine the outputs of an orchestration response as required by the parameters in
  the consuming device or program.
- Add static text to the response such as details about a company or customer.
- Delete information from the response so it cannot be read by the consuming
  program.

Figure 10–4 shows the Groovy template that you can use to manipulate orchestration output:
### Figure 10–4 Groovy Code Template for Manipulating Orchestration Output

```java
import groovy.transform.CompileStatic;
import groovy.json.JsonSlurper;
import groovy.json.JsonBuilder;
import com.oracle.el.common.OrchestrationAttributes;
@CompileStatic

String main(OrchestrationAttributes orchAttr, String input) {
    def jsonIn = new JsonSlurper().parseText(input);
    // modify jsonIn
    def jsonOut = new JsonBuilder(jsonIn).toString();
    // orchAttr.writeWarn("custom log entry - warning");
    // orchAttr.writeDebug("custom log entry - debug");
    return jsonOut;
}
```

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.

### 10.6 Additional Attributes and Methods Available in the Groovy Script Templates

All Groovy scripts are passed an OrchestrationAttributes instance called orchAttr that contains additional information that can be used in the Groovy script. Table 10–1 provides a description of the attributes. Table 10–2 describes the methods available in the OrchestrationAttributes class.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>orchestrationName</td>
<td>String</td>
<td>The name of the currently running orchestration.</td>
</tr>
<tr>
<td>token</td>
<td>String</td>
<td>The token for the current session.</td>
</tr>
<tr>
<td>langpref</td>
<td>String</td>
<td>The language of the execution user.</td>
</tr>
<tr>
<td>locale</td>
<td>String</td>
<td>The locale of the execution user.</td>
</tr>
<tr>
<td>dateFormat</td>
<td>String</td>
<td>The date format of the execution user.</td>
</tr>
<tr>
<td>dateSeparator</td>
<td>String</td>
<td>The date separator of the execution user.</td>
</tr>
<tr>
<td>simpleDateFormat</td>
<td>String</td>
<td>The Java simple date format of the execution user.</td>
</tr>
<tr>
<td>decimalFormat</td>
<td>String</td>
<td>The decimal format of the execution user.</td>
</tr>
</tbody>
</table>
Table 10–1 (Cont.) Attributes in the OrchestrationAttributes Class

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>addressNumber</td>
<td>Integer</td>
<td>The address number of the execution user.</td>
</tr>
<tr>
<td>alphaName</td>
<td>String</td>
<td>The name of the execution user.</td>
</tr>
<tr>
<td>appsRelease</td>
<td>String</td>
<td>The current application release.</td>
</tr>
<tr>
<td>country</td>
<td>String</td>
<td>The country code of the execution user.</td>
</tr>
<tr>
<td>username</td>
<td>String</td>
<td>The user name of the execution user.</td>
</tr>
<tr>
<td>environment</td>
<td>String</td>
<td>The current environment of the execution user.</td>
</tr>
<tr>
<td>tempDir (Orchestrator Studio 6.1.0)</td>
<td>String</td>
<td>The temp directory configured in the rest.ini.</td>
</tr>
</tbody>
</table>

Table 10–2 Methods in the OrchestrationAttributes Class

<table>
<thead>
<tr>
<th>Method</th>
<th>Response Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>writeWarn(String message)</td>
<td>String</td>
<td>Write a message as a warning to the log.</td>
</tr>
<tr>
<td>writeDebug(String message)</td>
<td>String</td>
<td>Write a message as a debug to the log.</td>
</tr>
<tr>
<td>writeWarn(String message, Exception e)</td>
<td>String</td>
<td>Write a message as a warning to the log with an exception.</td>
</tr>
<tr>
<td>writeDebug(String message, Exception e)</td>
<td>String</td>
<td>Write a message as a debug to the log with an exception.</td>
</tr>
<tr>
<td>getOrchestrationName()</td>
<td>String</td>
<td>Get the name of the currently running orchestration.</td>
</tr>
<tr>
<td>getToken()</td>
<td>String</td>
<td>Get the token for the current session.</td>
</tr>
<tr>
<td>getLangPref()</td>
<td>String</td>
<td>Get the language preference for the execution user.</td>
</tr>
<tr>
<td>getLocale()</td>
<td>String</td>
<td>Get the locale of the execution user.</td>
</tr>
<tr>
<td>getDateFormat</td>
<td>String</td>
<td>Get the date format of the execution user.</td>
</tr>
<tr>
<td>getDateSeparator()</td>
<td>String</td>
<td>Get the date separator.</td>
</tr>
<tr>
<td>getSimpleDateFormat()</td>
<td>String</td>
<td>Get the Java simple date format of the execution user.</td>
</tr>
<tr>
<td>getDecimalFormat()</td>
<td>String</td>
<td>Get the decimal format of the execution user.</td>
</tr>
<tr>
<td>getAddressNumber()</td>
<td>Integer</td>
<td>Get the address number of the execution user.</td>
</tr>
<tr>
<td>getAlphaName()</td>
<td>String</td>
<td>Get the name of the execution user.</td>
</tr>
<tr>
<td>getAppsRelease()</td>
<td>String</td>
<td>Get the current application release.</td>
</tr>
<tr>
<td>getCountry()</td>
<td>String</td>
<td>Get the country code for the execution user.</td>
</tr>
<tr>
<td>getUsername()</td>
<td>String</td>
<td>Get the username for the execution user.</td>
</tr>
<tr>
<td>getEnvironment() (Orchestrator Studio 6.1.0)</td>
<td>String</td>
<td>Get the current execution environment.</td>
</tr>
<tr>
<td>getTempDir() (Orchestrator Studio 6.1.0)</td>
<td>String</td>
<td>Get the temp directory configured in the rest.ini.</td>
</tr>
</tbody>
</table>
### Table 10–2  (Cont.) Methods in the OrchestrationAttributes Class

<table>
<thead>
<tr>
<th>Method</th>
<th>Response Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>getTempFileName(String fileName)</code></td>
<td>String</td>
<td>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.</td>
</tr>
<tr>
<td>(Orchestrator Studio 6.1.0)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>getUniqueTempFileName(String baseFileName)</code></td>
<td>String</td>
<td>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.</td>
</tr>
<tr>
<td>(Orchestrator Studio 6.1.0)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>getMachineKey()</code></td>
<td>String</td>
<td>Get the machine key (first 10 characters) of the AIS server. (Release 9.2.4.2)</td>
</tr>
<tr>
<td>(Orchestrator Studio 9.2.4)</td>
<td></td>
<td>The maximum length for this field is ten characters. If the string is longer than ten characters, it will be truncated.</td>
</tr>
<tr>
<td><code>getMachineName()</code></td>
<td>String</td>
<td>Get the machine name of the AIS server. (Release 9.2.4.2)</td>
</tr>
<tr>
<td>(Orchestrator Studio 9.2.4)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>getLongUserId()</code></td>
<td>String</td>
<td>Get the long User Id for the currently logged in user. (Release 9.2.4.2)</td>
</tr>
<tr>
<td>(Orchestrator Studio 9.2.4)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>toString()</code></td>
<td>String</td>
<td>Outputs a JSON string representing all values in this class.</td>
</tr>
</tbody>
</table>
This chapter contains the following topics:

- Section 11.1, "Understanding Run Orchestrations"
- Section 11.2, "Testing an Orchestration"

11.1 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.

### 11.1.1 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.
11.2 Testing an Orchestration

To test 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 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.

   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:

   - application/json
   - 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:

   - application/json
   - application/xml

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.

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.
12
Administering the Orchestrator Studio and Orchestrations

This chapter contains the following topics:

- Section 12.1, "Understanding Orchestration Life Cycle Management"
- Section 12.2, "Managing Orchestrator Studio Security"
- Section 12.3, "Setting Up UDO Security for Orchestrator Studio Users"
- Section 12.4, "Clearing Orchestration Cache on the AIS Server"
- Section 12.5, "Setting Up Orchestrator Health and Exception Monitoring"

12.1 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.

12.2 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:

<table>
<thead>
<tr>
<th>Form Name</th>
<th>Form ID</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Studio</td>
<td>W98I0000A</td>
<td>Log in to the Orchestrator Studio.</td>
</tr>
<tr>
<td>Scheduler</td>
<td>W98I0000B</td>
<td>Run the Scheduler services or use the Scheduler page in the Orchestrator Studio.</td>
</tr>
<tr>
<td>Admin Services</td>
<td>W98I0000C</td>
<td>Run the Admin services or clear caches using the Refresh Cache button on the Run Orchestrations page.</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Permission to</th>
<th>Studio (Scheduler)</th>
<th>Studio (Clear Cache)</th>
<th>Scheduler (REST Service)</th>
<th>Admin (REST Service)</th>
</tr>
</thead>
<tbody>
<tr>
<td>P98I0000 (All)</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>N</td>
<td>N</td>
<td>Not allowed</td>
<td>Not allowed</td>
<td>Not allowed</td>
</tr>
<tr>
<td>N</td>
<td>N</td>
<td>Not allowed</td>
<td>Not allowed</td>
<td>Not allowed</td>
</tr>
<tr>
<td>N</td>
<td>Not allowed</td>
<td>Not allowed</td>
<td>Not allowed</td>
<td>Not allowed</td>
</tr>
</tbody>
</table>
12.3 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.

### 12.3.1 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.


<table>
<thead>
<tr>
<th>Permission to</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y  N  N  Y</td>
<td>Not allowed</td>
</tr>
<tr>
<td>Y  N  Y  N</td>
<td>Not allowed</td>
</tr>
<tr>
<td>Y  N  Y  Y</td>
<td>Not allowed</td>
</tr>
<tr>
<td>Y  Y  N  N</td>
<td>Allowed</td>
</tr>
<tr>
<td>Y  Y  N  Y</td>
<td>Allowed</td>
</tr>
<tr>
<td>Y  Y  Y  N</td>
<td>Allowed</td>
</tr>
<tr>
<td>Y  Y  Y  Y</td>
<td>Allowed</td>
</tr>
</tbody>
</table>
12.3.2 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.


12.3.3 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.


12.4 Clearing Orchestration Cache on the AIS Server

The AIS Server caches all orchestration files processed by the Orchestrator. If Orchestrator Studio users modify orchestration components that are currently in use, an administrator must clear the AIS Server cache for the modifications to take effect. Clearing the cache forces the AIS Server to reload files from disk to cache.

**Note:** Clearing the cache is not necessary to run new orchestration files.
You can clear the cache using either of the following methods:

- Use the AIS Administration Service to clear the AIS Server cache (**recommended**). The AIS Administration Service is a REST service on the AIS Server that is exposed like any other service on the AIS Server.
- Restart the AIS Server. This method is not recommended because:
  - If the Orchestrator is currently running an orchestration, it will result in invalid processing of that orchestration.
  - It results in server downtime, which might impact other applications that use the AIS Server.

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.

### 12.4.1 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**.

### 12.4.2 Using the AIS Administration Service to Clear Cache

Run the AIS Administration Service by placing a URL in a browser with valid EnterpriseOne credentials. The AIS Administration Service takes an EnterpriseOne username and password as input. You can invoke the service using either a GET or a POST HTTP method.

The URI is:

```
http://<ais_server>:<port)//jderest/adminservice
```

The AIS Administrator Service is able to take credentials in several forms.

For the GET call, you can either use basic authorization or you can provide the username and password as URL parameters, for example:

```
http://<ais_server>:<port>/jderest/adminservice?username=<userid>&password=<pwd>
```

For the POST operation you can use basic authorization, provide username and password as parameters, or provide username and password as JSON input, for example:

```
{
  "username":"user",
  "password":"pwd"
}
```

If the service succeeds, the response looks like this:

```
{"message":"All XML File Caches and X-Ref/Whitelist Caches have been cleared and refreshed.","timeStamp":"2015-04-30:14.26.58"}
```

If the service fails, these are the possible reasons:
12.5 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

12.5.1 Downloading and Installing the Orchestrator Monitor

To access the Orchestrator Monitor download, use the Update Center or Change Assistant to search on the bug number for your release:

- 28186193 - ORCHESTRATOR MONITOR ENHANCEMENT 9.2
- 28389735 - ORCHESTRATOR MONITOR ENHANCEMENT 9.1

The download contains the following files:

- Orchestrator Monitor.
  This is delivered as an EnterpriseOne page which is referenced by an external form (P980060X | W980060XB). For the system to properly render the Orchestrator Monitor application, you must publish this page.

- Health Summary (optional).
  This provides another way to view Orchestrator health details outside of the Orchestrator Monitor; it does not contain exception information. It can be published as a composed page in EnterpriseOne, but can also be made available within the My Work List composed page (recommended). For more information, see "My Work List" in the JD Edwards EnterpriseOne Tools Foundation Guide.

- Exceptions Since Yesterday watchlist (optional).
  Implement this watchlist and the following query so users can be alerted to Orchestrator exceptions in EnterpriseOne. For more information, see Managing Orchestrator Health and Exception Records in EnterpriseOne.
12.5.2 Enabling Orchestrator Health and Exception Tracking in Server Manager

An administrator must enable performance and exception tracking in the AIS Server configuration settings in Server Manager. When enabled, the system stores health and exception records in the Health (F980061) and Exception (F980060) tables in EnterpriseOne. Data from these records is used to display health and exception details in the Orchestrator Monitor (P980060X).

The AIS Server contains an additional setting to enable management of health and exception records stored in the aforementioned tables through the EnterpriseOne Orchestrator Health and Exceptions program (P980060). Access to these records in P980060 requires providing the credentials of an EnterpriseOne user who has read-write access to these tables.

In addition, in the AIS Server configuration settings, you can specify how the system processes errors and warnings returned from form requests within orchestrations.

---

**Note:** If debugging is enabled on the AIS Server, the AIS Server log will indicate that an exception was saved to the database each time the Orchestrator generates an exception.

---

**Caution:** Based on the number of sessions, exception records in the database have the potential to grow rapidly.

---

To enable exception tracking:

1. In Server Manager, under the Advanced configuration settings for the AIS Server, locate the General Settings section.

2. In the General settings, click the **Save Orchestration Exceptions** check box to enable Orchestrator health and exception tracking.

3. In the "Full path to exceptions directory" field, enter a path to a directory on the AIS Server where exceptions are saved if the system temporarily goes offline.

   When the system resumes, exceptions in the file system are automatically written to the F980060 table.

   If you do not complete this field, files will be saved to a temporary directory on the AIS Server.

4. To enable access to health and exception records in P980060, enter the credentials of an EnterpriseOne user who has read-write access to F980061 and F980060 tables:
   - **Exception/Health Monitor User**
   - **Exception/Health Monitor Password**

   This makes these records accessible through the EnterpriseOne Orchestrator Health and Orchestrator Exceptions program (P980060).

5. In the Exception Scenario field, enter either or both of the following values if you want errors and warnings returned from form requests to be processed as failures:
   - **FSR_ERROR**
Enter if you want **errors** returned by a form request to be processed as a failure.

- **FSR_WARNING**

Enter if you want **warnings** returned by a form request to be processed as a failure. This level of error handling works only for form requests configured with the “Stop on Warnings” setting in the Orchestrator Studio.

### 12.5.3 Set Up User Access to the Orchestrator Monitor and Orchestrator Health and Exceptions Program

An administrator must use EnterpriseOne application security to enable user access to the Orchestrator Monitor (P980060X) and the Orchestrator Health and Exceptions program (P980060). See “Managing Application Security” in the *JD Edwards EnterpriseOne Tools Security Administration Guide*.

### 12.5.4 Enable UDO View Security to Monitor Orchestrations, Notifications, and Schedules

Users must have UDO view security access to the orchestrations, notifications, and schedules that they want to monitor in the Orchestrator Monitor. For more information about UDO view security, see “Managing UDO View Security” in the *JD Edwards EnterpriseOne Tools Security Administration Guide*. 
Understanding the AIS Server Discovery Service

The discovery service is a service on the AIS Server that enables any AIS client or consumer of orchestrations, including the 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 13–1 provides an example of the orchestration details in a JSON response.

**Example 13–1 Orchestration Details in the JSON Response from the Discovery Service**

```json
{
    "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 Measurement 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"
  }
]
Open Standards for Orchestrations

This chapter contains the following topics:

■ Understanding Open Standards for Orchestrations
■ Discovering Orchestrations Using Open Standards

14.1 Understanding Open Standards for Orchestrations

Open standards for orchestrations allows 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 gives 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.

14.2 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:


---

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.

### 14.2.1 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:

```xml
<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.