JD Edwards EnterpriseOne Tools

Application Interface Services Server Reference Guide

9.2

JD Edwards EnterpriseOne Tools
Application Interface Services Server Reference Guide

9.2

Part Number: E61545-23

Contents

Preface

Welcome to the JD Edwards EnterpriseOne documentation.

Documentation Accessibility

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

Access to Oracle Support

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

Related Information

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

`http://learnjde.com`

Conventions

The following text conventions are used in this document:

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

1 Understanding the JD Edwards EnterpriseOne Application Interface Services (AIS) Server

Understanding This Guide

This guide is organized into the following parts:

- Part I, **Application Interface Services (AIS) Server Deployment and Management**provides an overview of the AIS Server, information about deploying an AIS Server through Server Manager, and an overview of AIS Server security and management.
  This part contains the following chapters
  Chapter 1, "Understanding the JD Edwards EnterpriseOne Application Interface Services (AIS) Server"
  Chapter 2, "Certifications"
  Chapter 3, "Configuring the AIS Server"
  Chapter 4, "Understanding AIS Authentication"
  Chapter 5, "Managing the AIS Server Through Server Manager"
  Chapter 6, "Configuring Scheduler Resilience (Release 9.2.2.4)"
- Part II, **AIS Client Development Resources** describes how to use the tools that aid in the development of AIS clients. It also describes the AIS Server capabilities and endpoints that enable AIS client applications to interact with EnterpriseOne business processes and tasks.
  Except where noted, the features described in Part II can be used in all AIS client applications. Refer to the information in this part as a companion to the following guides that describe how to create and customize the different types of AIS clients:
  - JD Edwards EnterpriseOne Tools UX One Deployment and Development Guide
  - JD Edwards EnterpriseOne Tools Developing and Customizing Mobile Enterprise Applications Guide
  - JD Edwards EnterpriseOne Tools Deploying and Developing Oracle Application Development Framework (ADF) Applications for EnterpriseOne
  - JD Edwards EnterpriseOne Application Interface Services Client Java API Developer's Guide
  - JD Edwards EnterpriseOne Tools REST API for the Application Interface Services Server Guide

  Part II contains the following chapters:
  Chapter 7, "Using the AIS Client Class Generator"
  Chapter 8, "Remapping Fields for Customized EnterpriseOne Forms"
  Chapter 9, "AIS Server Capabilities and Services"

Chapter 10, "Caching Service Request Responses (Release 9.2.1.1)"

Understanding the JD Edwards EnterpriseOne Application Interface Services (AIS) Server

The JD Edwards EnterpriseOne Application Interface Services (AIS) Server is the communication interface between JD Edwards EnterpriseOne and various AIS Server clients. The AIS Server provides a JSON over REST interface (HTTP), a light-weight interface that enables AIS clients to interact with EnterpriseOne applications and forms. Any client or software language that uses JSON over REST can interface with the AIS Server.

Some of the clients that use the AIS Server to interact with EnterpriseOne include:

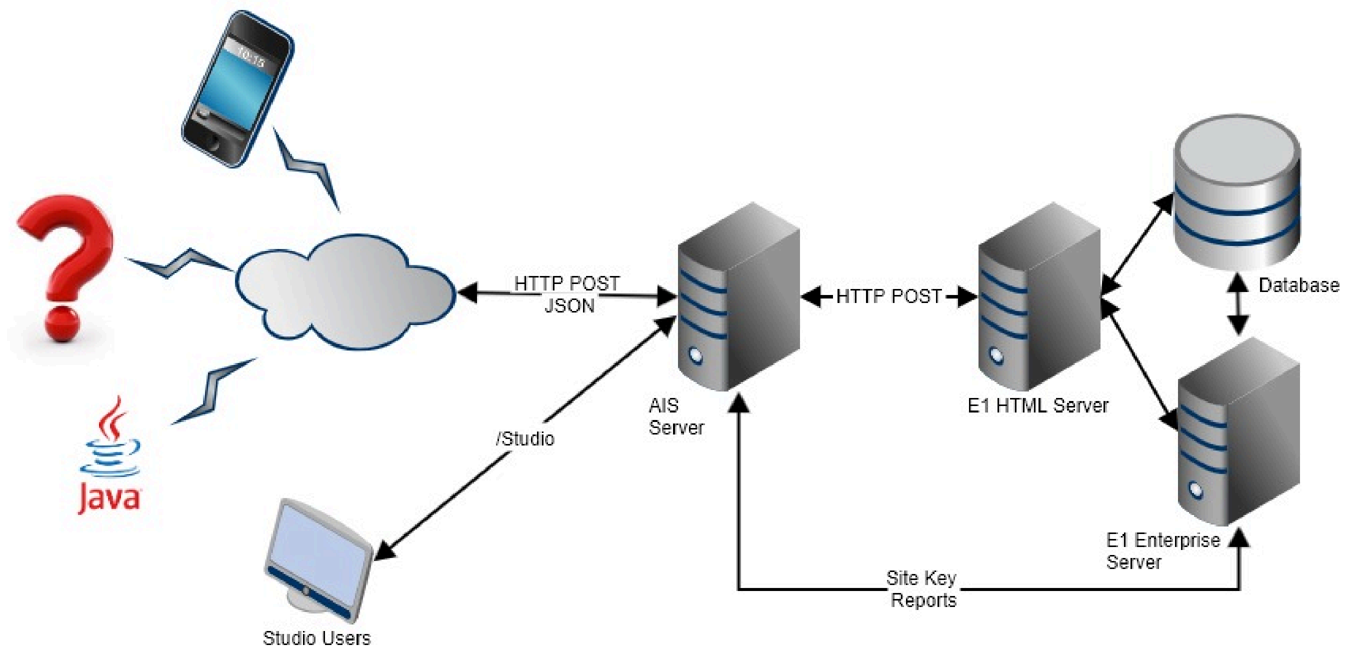
- EnterpriseOne applications created with Oracle Application Development Framework (ADF), otherwise referred to as EnterpriseOne ADF applications.
- Applications created with the AIS Client Java API.

  The Application Interface Services (AIS) Client Java API enables developers to use any development tool that works with Java APIs to create custom applications that interact with EnterpriseOne. For example, you can create a simplified kiosk application for your warehouse, an application that combines features from multiple EnterpriseOne applications into a single purpose-built interface, or an application for the latest wearable device.
- Internet of Things (IoT) devices.

  The AIS Server enables the transfer of data from third-party IoT devices to EnterpriseOne. The EnterpriseOne IoT Orchestrator processes orchestrations on the AIS Server that contain instructions for transferring third party data to actionable business processes in EnterpriseOne.
- Components created using Oracle Java Extension Toolkit (JET) that can run inside EnterpriseOne UX One pages.
- EnterpriseOne mobile enterprise applications created with Oracle Mobile Application Framework (MAF).
- Starting with EnterpriseOne Tools 9.2.3, developers can configure EnterpriseOne to automatically launch an orchestration or notification on the AIS Server from an EnterpriseOne interactive or batch application.

The Orchestrator Studio 9.2.4.0 is deployed along with the AIS Server 9.2.4.0 and can be accessed by using the AIS server URL. The Orchestrator Studio 9.2.4 does not require additional installation and maintenance of discrete WebLogic Server or ADF environment. The JD Edwards EnterpriseOne Orchestrator Studio 9.2.4.0 is installed on AIS server instance. (Release 9.2.4.0)

The interaction with EnterpriseOne requires an AIS Server configuration with the EnterpriseOne HTML Server. The following graphic on EnterpriseOne Architecture with an EnterpriseOne AIS Server illustrates how the AIS Server functions as the interface between AIS clients and JD Edwards EnterpriseOne.



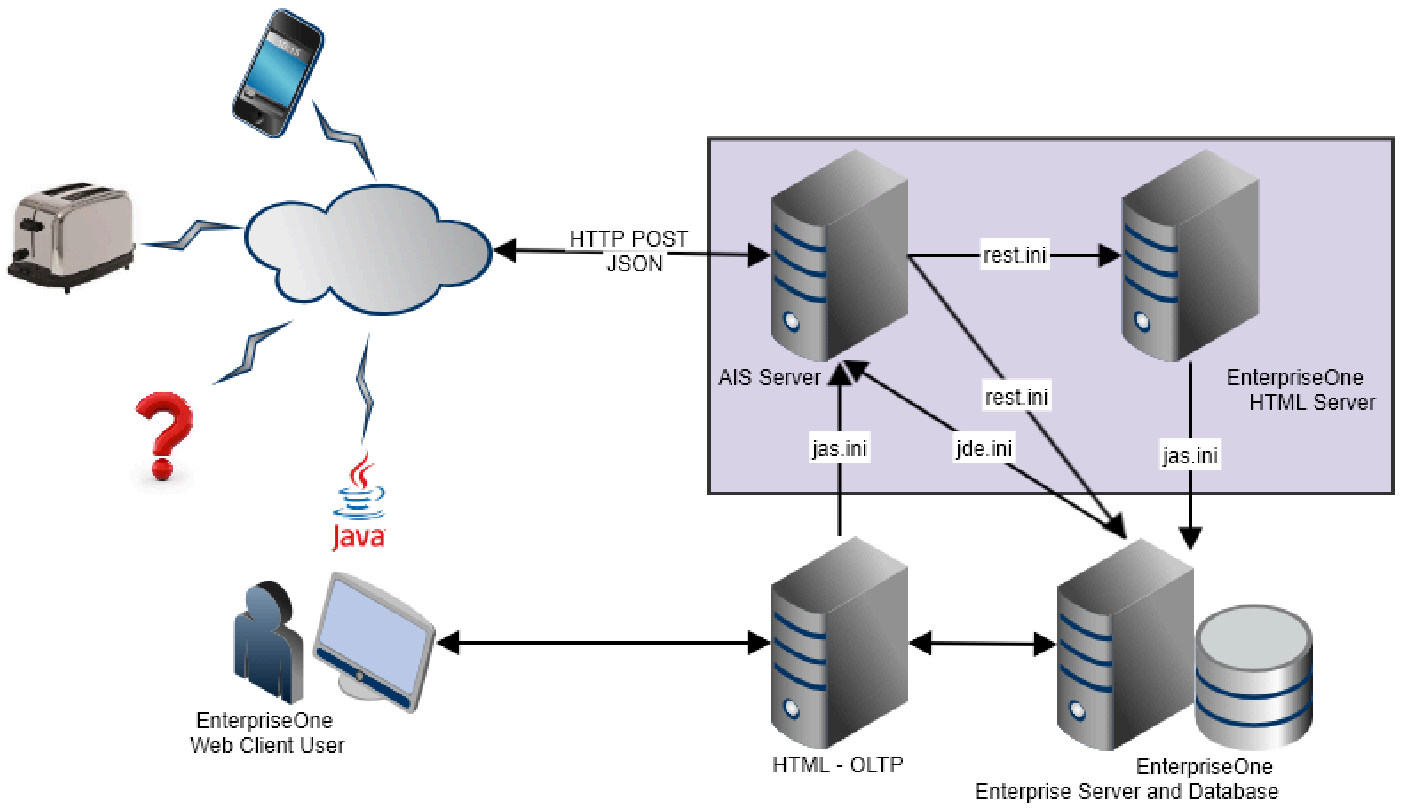
Note: EnterpriseOne ADF applications and the EnterpriseOne Orchestrator Studio 8 and earlier vesions (which is built with Oracle ADF) also require an Oracle WebLogic Server with ADF runtime (not depicted in illustration).

Additional EnterpriseOne HTML Server Instance for Processing AIS Requests

You can 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 used by EnterpriseOne web client users is not impacted by AIS Server requests.

With this configuration, make sure that you select the "Enable AIS Watchlist Execution" check box in the Form Service section of the HTML Server Web Runtime settings in Server Manager. This option applies only to an EnterpriseOne configuration with an AIS Server. If you select this check box, you must configure the settings for the AIS Server in the same Form Service section.

This graphic shows the EnterpriseOne architecture with an additional, dedicated HTML Server instance for processing AIS Server requests. The illustration also depicts the configuration files that enable the communication between servers.


HTTP POST
JSON
rest.ini
AIS Server
EnterpriseOne HTML Server
rest.ini
jas.ini
jde.ini
jas.ini
Java
EnterpriseOne Web Client User
HTML - OLTP
EnterpriseOne Enterprise Server and Database

Note: Starting with Tools Release 9.2.5, you can use the Server Manager Console to configure the **AIS Login On Demand** setting for the HTML server. This setting ensures that the AIS session used by the HTML server is not established until it is required. In the earlier releases, the AIS session was established when the user logged in to the HTML server.

2 Certifications

Certifications

Customers must conform to the supported platforms for the AIS Server, which can be found in the Certifications tab on My Oracle Support: *`https://support.oracle.com`*.

On the Certifications Search tab, search on JD Edwards EnterpriseOne Application Interface Services Server for a list of certifications (otherwise known as minimum technical requirements).

3 Configuring the AIS Server

Deploying the AIS Server through Server Manager

Deploy the AIS Server as a managed instance through Server Manager. You can deploy the AIS Server on Oracle WebLogic Server or IBM WebSphere Application Server.

If you are using Oracle WebLogic Server 12.2.1 to host the AIS Server, you must upgrade the AIS Server and Server Manager to EnterpriseOne Tools 9.2.0.5 or higher. You cannot run AIS Server releases 9.2.0.4 and lower on Oracle WebLogic Server 12.2.1.

See *"Create an Application Interface Services (AIS) Server as a New Managed Instance" in the JD Edwards EnterpriseOne Tools Server Manager Guide* for instructions on how to deploy the AIS Server as a managed instance.

Note: When you deploy the AIS Server, remember the server name and port number. These values make up the URL that you need to provide users of mobile enterprise applications for login: http://<*ais_servername*>:<*portnumber*> The first time users open a mobile application, they are prompted to enter this URL to connect to the AIS Server. However, if you deploy mobile application archives with an integration with Oracle Mobile Cloud Service, mobile applications users would enter the URL to the backend Mobile Cloud Service instance for login. See "Integration with Oracle Mobile Cloud Service" in the *JD Edwards EnterpriseOne Tools Developing and Customizing Mobile Enterprise Applications Guide* for details.

Additional Required AIS Server Configurations

After deploying the AIS Server through Server Manager, perform the following tasks to complete the configuration:

- *Configuring the Allowed Hosts Setting for the EnterpriseOne HTML Server*
- *Configuring the AIS Server with an EnterpriseOne Enterprise Server (Release 9.2.2.2)*
- *Verifying the JAVA Argument for AIS Server (Oracle WebLogic Server Only)*
- *Configuring the Keep JAS Session Open Setting for the AIS Server*
- *Configuring SSL/TLS for the AIS Server (Release 9.2.1)*
- *Configuring the Allow PS Token Login Setting for EnterpriseOne ADF Applications (Release 9.2.0.5)*
- *Configuring Oracle WebLogic Server Domain for HTTP Basic Authentication*
- *Configuring the AIS Server as a Cluster*

Configuring the Allowed Hosts Setting for the EnterpriseOne HTML Server

In Server Manager, configure the Allowed Hosts setting for the EnterpriseOne HTML Server to specify the AIS Server host from which the EnterpriseOne HTML Server will receive requests.

1. In Server Manager, access the EnterpriseOne HTML Server managed instance.

2. Expand the Web Runtime area (Advanced View), and in the Allowed Hosts field, enter the IP address of the AIS Server.
3. Restart the server for the changes to take effect.

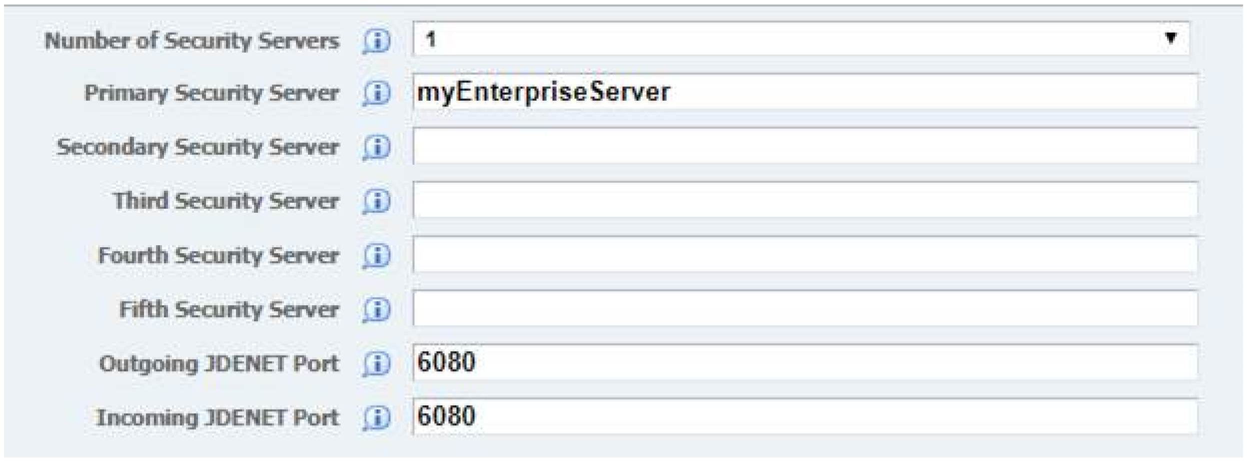
Note: If the AIS client application fails to connect to the server, verify that the IP Address of the AIS Server has been entered correctly in the Allowed Hosts field. If the IP Address is correct and the connection still fails, then enter an * (asterisk) in the Allowed Hosts setting, which enables the EnterpriseOne HTML Server to accept requests from any host.

Configuring the AIS Server with an EnterpriseOne Enterprise Server (Release 9.2.2.2)

Starting with EnterpriseOne Tools 9.2.2.2, an AIS Server requires a configuration with the EnterpriseOne Enterprise Server (Security Server). This configuration ensures that login requests to the AIS Server use the site key on the Enterprise Server for encryption. If not configured, all login requests to the AIS Server will fail.

Also, starting with EnterpriseOne Tools 9.2.3, the AIS Server supports calls from the Enterprise Server when B98ORCH is executed on the Enterprise Server to call an orchestration or notification on the AIS Server.

In Server Manager, access the AIS Server Security Information settings and define a valid Security Server and port for the AIS Server. For a description of each field, access the field-level help within Server Manager.



Number of Security Servers	1
Primary Security Server	myEnterpriseServer
Secondary Security Server	
Third Security Server	
Fourth Security Server	
Fifth Security Server	
Outgoing JDENET Port	6080
Incoming JDENET Port	6080

Configuring Multiple AIS Servers (Release 9.2.5)

You can deploy a single enterprise server for all environments and associate that enterprise server with AIS servers in multiple environments. This configurability facilitates the segregation of AIS servers across environments, such as development, test, and production, while making it possible for a single enterprise server to serve all the environments.

You use the Web Service Soft Coding Records program (P954000) to configure web service soft coding records based on web service soft coding templates. The administrator can create a soft coding record with the **AIS_CONNECTION** soft coding key.See*Working With Soft Coding.*

You can also set up these connections using the Orchestrator Studio. See *Creating a Soft Coding Record for an Enterprise Server AIS Connection section in the orchestrator guide* for more information.

Before Release 9.2.5, the AIS endpoint string was built using the values in AISProtocol, AISHost, and AISPort that are available in the JDE.INI file as shown in the following example:

```
[FORMSERVICE]

    AISProtocol=https
     AISHost=MyAISHost
     AISPort=8001
     AISMaxConcurrentCalls=1
```

As of Release 9.2.5, the endpoint string is obtained from a table and these three JDE.INI settings are used as a fallback if the data does not exist in the table.

The endpoint string value is read from the F954001 table, column when the value of WSTPNAME column is **AIS_CONNECTION**. The row used in this process depends on the user/ role and environment of the signed-in user.

This is the hierarchy rule of the soft coding application:

1. If the user and environment exist in the F954001 table (column UGRP or ENVH, respectively), the system uses the endpoint value in the WSTPVAL column (Template Value field) for that record. If the user and environment does not exist, the system performs step 2.
2. If the user is signed in for the role *ALL, then the system goes through all the roles in *ALL in a hierarchical order. The system will use the endpoint value in the WSTPVAL column for the first role or environment that exists in the F954001 table (column UGRP or ENVH, respectively). If the user is not signed in to the *ALL role, the system performs step 3.
3. If the user is signed in using a specific role, and if that role and environment exists in the F954001 table (column UGRP or ENVH, respectively), the system uses the endpoint value in the WSTPVAL column (Template Value field) for that record. If not, the system performs step 4.
4. If *PUBLIC and environment exists in the F954001 table (column UGRP or ENVH, respectively), the system uses the value in the WSTPVAL column for that record. If *PUBLIC and environment does not exist in the F954001 table, the system looks into [FORMSERVICE] in the JDE.INI file.

Verifying the JAVA Argument for AIS Server (Oracle WebLogic Server Only)

If the AIS Server is deployed on Oracle WebLogic Server, you must make sure that the server configuration includes a JAVA argument for starting the server. To do so:

1. In the WebLogic Admin Console, locate the AIS Server instance.
2. Click the **Server Start** tab.
3. Verify that the following argument is in the Arguments field:

   ```
   -DUseSunHttpHandler=true

   -Dweblogic.http.headers.enableHSTS=true
   ```

Note: If you plan to implement HTTP Strict Transport Security, refer to this Oracle document: *Command Reference for Oracle WebLogic Server 14c* in the section entitled: *HTTP Strict Transport Security*.

Configuring the Keep JAS Session Open Setting for the AIS Server

In Server Manager, ensure that the Keep JAS Session Open setting for the AIS Server is set to True/Checked.

Configuring SSL/TLS for the AIS Server (Release 9.2.1)

The availability of the AIS Server port number can enable AIS client access to the AIS Server without the use of a VPN. To protect information sent between an AIS client and the AIS Server, you should use SSL when configuring the AIS Server. If you enable the AIS Server for SSL/TLS (HTTPS), you must use a valid certificate. The encryption algorithm for the certificate must be this type:

- Signature.SHA1withRSA

Configuring the Allow PS Token Login Setting for EnterpriseOne ADF Applications (Release 9.2.0.5)

If you are running EnterpriseOne ADF applications, make sure that the Allow PS Token Login check box is selected in the AIS Server Security Settings section in Server Manager. EnterpriseOne ADF applications running in JD Edwards EnterpriseOne use the PS Token to establish a session with the AIS Server. If this setting is not checked, the ADF integration will fail.

Configuring Oracle WebLogic Server Domain for HTTP Basic Authentication

REST services on the JD Edwards EnterpriseOne AIS Server can use HTTP Basic Authentication for access. Support for HTTP Basic Authentication is enabled out of the box and is required to run the EnterpriseOne Orchestrator Client, create custom Java calls from orchestrations, and use the AIS client Java API (versions 1.2.1.x and higher).

If the AIS Server is deployed on Oracle WebLogic Server, Oracle WebLogic Server may require the following additional configuration depending on how you manage users:

- If you are maintaining a user registry in Oracle WebLogic Server that matches the user registry in EnterpriseOne, with identical sets of user names and passwords in each system, you do NOT need to modify your configuration.

- If you are NOT maintaining identical sets of users in Oracle WebLogic Server and EnterpriseOne, then you need to perform the following steps to modify your Oracle WebLogic Server configuration. This ensures that Oracle WebLogic Server will not intercept HTTP Basic Authentication credentials passed from the REST service.
    a. In the WebLogic Server domain for your AIS Server, in the Config directory, find the config.xml file.
    b. Add this configuration as the last line within the `<security-configuration>` element, just before the `</security-configuration>` tag:

       ```
       <enforce-valid-basic-auth-credentials>false</enforce-valid-basic-auth-credentials>
       ```
    c. Restart the AIS Server for the changes to take effect.

The following is an example of this configuration in the `<security-configuration>` element:

```
<node-manager-password-encrypted>{AES}tzAokzTHACTNNmkuutLPQEpP8bfk7Ble24vmoycooic=</node-
manager-password-encrypted>
    <enforce-valid-basic-auth-credentials>false</enforce-valid-basic-auth-credentials>
  </security-configuration>
  <server>
```

Configuring the AIS Server as a Cluster

If using an AIS cluster configuration for notifications, the AIS Server needs to be tied to a single EnterpriseOne HTML Server, not a cluster. You can still cluster the AIS Server, but you must have every AIS node point to its own HTML Server.

Testing the AIS Server Configuration

In Server Manager, use the "Test Configuration" button to test the AIS Server setup. This uses the Defaultconfig service on the AIS Server to test communication between the AIS Server and the EnterpriseOne HTML Server.

Using a REST client testing tool, you can perform an additional test by performing a POST to the following URL:

http://<host>:<port>/jderest/formservice

Make sure to include Basic Authorization credentials in the request and include the following JSON in the body:

```
{
   "formName":"P01012_W01012B"
}
```

If you encounter any issues with the AIS Server configuration, see *AIS Troubleshooting* in this guide.

4 Understanding AIS Authentication

Overview

The AIS Server uses EnterpriseOne authentication to authenticate AIS clients. All AIS sessions are established with requests to the EnterpriseOne HTML Server to establish a corresponding HTML Server (JAS) session. The AIS Server can maintain open sessions linked to open JAS sessions. It can also execute stateless calls where sessions are temporarily established only for the time of the call and thereafter terminated.

You can configure the AIS Server to use SSL so that all communication is over HTTPS. It can also be configured to communicate over HTTPS with the EnterpriseOne HTML Server.

The AIS Server supports the following authentication methods or login types:

- Username and Password

  This login type is used in JD Edwards EnterpriseOne mobile enterprise applications for authentication.
- Basic Authentication

  This login type is used for authenticating Internet of Things (IoT) devices calling orchestrations on the AIS Server. It is also used by the EnterpriseOne Orchestrator Client to test running orchestrations on the AIS Server.
- PS Token

  This login type is used in EnterpriseOne ADF applications. It is also used by EnterpriseOne Pages designed to call AIS services through the e1pagehelper.js API.
- JSON Web Token (Release 9.2.0.5)

  This login type can be used in a JD Edwards EnterpriseOne mobile application integration with Oracle Mobile Cloud Service. You can also use this login type to employ OAuth 2.0 authentication for third-party AIS clients, including clients developed using the AIS Client Java API to call AIS services and orchestrations on the AIS Server.

  Note: You can use OAuth 2.0 if you have an EnterpriseOne configuration with Oracle Access Manager (OAM), where OAM is the OAuth provider.

  Note: Starting with Tools Release 9.2.8.2, you can also use a JWT Assertion in a JD Edwards EnterpriseOne mobile application integration with OCI Identity Access Management (IAM) with Microsoft Entra ID as an External IdP for SSO to access AIS REST APIs.

Starting with Tools Release 9.2.4, you have an option to use Header based authentication. You can use the Header based authentication method for requesting an AIS token and using an AIS token.

Requesting a Token

The /tokenrequest service has three HTTP headers that can be used in conjunction with Basic Authorization or JWT token request to indicate the environment, role and device that the session is established with. You can include the following authentication credentials in the Request Header instead of passing these values in the response body:

```
jde-AIS-Auth-Environment

jde-AIS-Auth-Role

jde-AIS-Auth-Device
```

Example

POST to /tokenrequest they would look like this:

```
POST /jderest/v2/tokenrequest HTTP/1.1

Accept: application/json

Content-Type: application/json

jde-AIS-Auth-Environment: JDV920

jde-AIS-Auth-Role: *ALL

jde-AIS-Auth-Device: Postman

Authorization: Basic S09VOktPVQ==
```

Note: When AIS Server Clients are configured with OCI IAM and Microsoft EntraID as an External IdP, AIS REST APIs can be accessed by requesting a token as a JWT Bearer Token generated using the public and private certificates of the clients.

Using a Token

To use a token, all the AIS services allow two HTTP headers for passing the token and device name. You can include the following authentication in the Request Header instead of passing these values in the response body. Device name is optional. However, if the token was requested with device name, the authentication must be used with device name.

```
jde-AIS-Auth

jde-AIS-Auth-Device
```

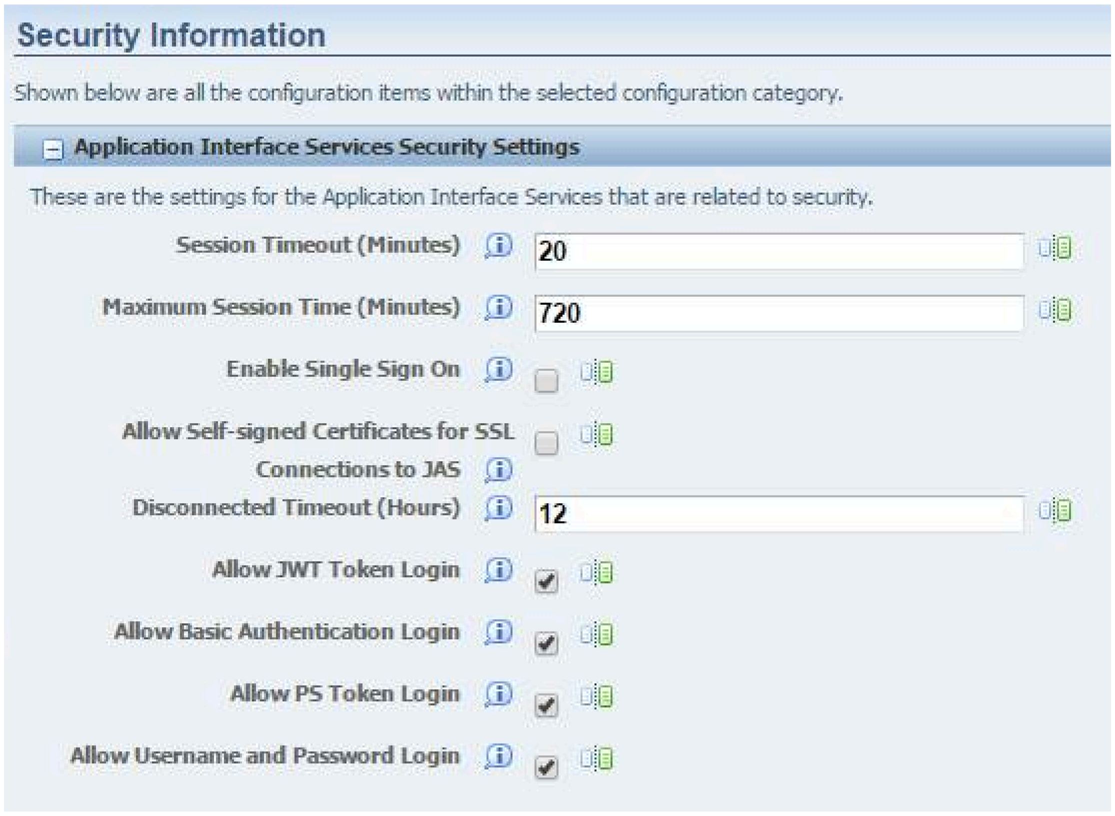
Example

POST to /formservice they would look like this:

```
POST /jderest/v2/formservice HTTP/1.1

Accept: application/json

Content-Type: application/json

jde-AIS-Auth: 0449gCaVHmzYCg3/+3qobSsCukOavk5Xvrn7E8c/
VNsP4I=MDE5MDEzMTMwMjQ0NzY0NDQ1MTkwNTY0MU5pY29sZVBvc3RtYW4xNTYxNDgyMzE5Nzgy

jde-AIS-Auth-Device: Postman
```

You must make sure that the authentication method or login type used by an AIS client is enabled in the Application Interface Services Security Settings section in Server Manager. This graphic **Allowed Login Type Settings for the AIS Server** shows the AIS Server login type settings in Server Manager, which include:

- Allow JWT Token Login
- Allow Basic Authentication Login
- Allow PS Token Login
- Allow Username and Password Login


Security Information

Shown below are all the configuration items within the selected configuration category.

Application Interface Services Security Settings

These are the settings for the Application Interface Services that are related to security.

Session Timeout (Minutes)	20
Maximum Session Time (Minutes)	720
Enable Single Sign On	☐
Allow Self-signed Certificates for SSL Connections to JAS	☐
Disconnected Timeout (Hours)	12
Allow JWT Token Login	✓
Allow Basic Authentication Login	✓
Allow PS Token Login	✓
Allow Username and Password Login	✓

Session Management

After a token request is sent to the AIS Server with successful authentication, the AIS Server generates a token and maintains a session for the user session according to the time out and time-to-live settings in Server Manager (rest.ini). A corresponding user session is also maintained on the EnterpriseOne HTML Server. You can view the AIS sessions in Server Manager, which displays "AIS Server" in the Display Mode for active AIS sessions. The AIS token is the key to the user session and must be passed on to all subsequent calls that use that AIS session.

For stateless AIS requests, credentials are supplied (not AIS tokens). Requests are given a temporary session that is removed after a request completes.

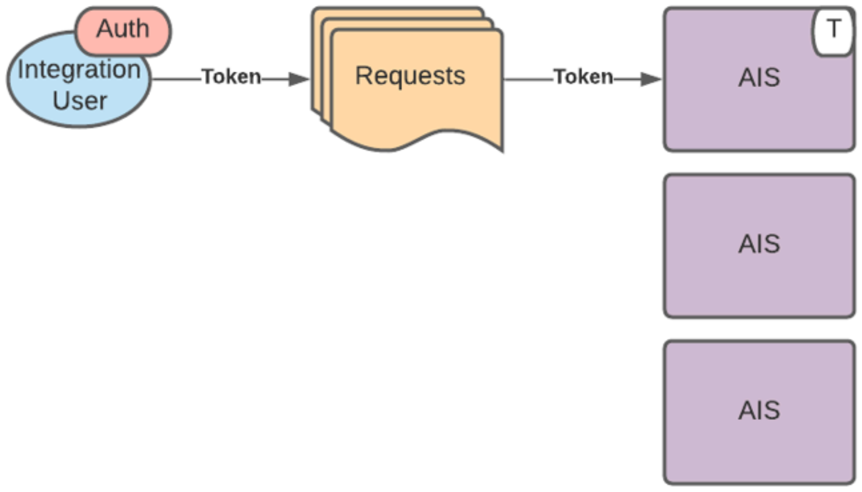
The original security model for mobile applications still applies, even for non-mobile clients. The deviceName (or Device ID) is not required. If Device ID is not passed, the requesting IP address is used. Thus a token requested from one device or IP address cannot be used by another device or IP address. Validation is performed every time the token is used.

Understanding Stateless Load Balancing for the AIS Server (Tools Release 9.2.5.2)

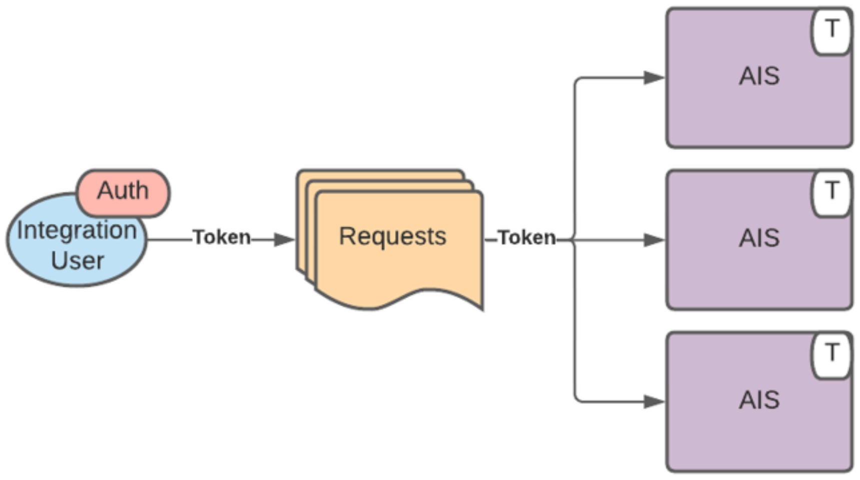
Overview

The Secure Session Balancing across AIS Servers feature (9.2.5.2) provides stateless load-balancing, which enables AIS servers to share sessions across multiple load-balanced AIS server instances.

Without Stateless Load Balancing, an established session can be used only by the AIS Server on which the session was established. The load balancer relies on configured session affinity to route requests to the same server every time when an established session is used. Sessions are retrieved using the AIS token and without Stateless Load Balancing, as you can see in the following image a single session is retrieved, which is represented here by “T”:



When Stateless Load Balancing is applied, a session is spread across all the participating nodes in the cluster. The same AIS token is used to retrieve the session on any of the participating nodes. Cookies or other mechanisms that provide session affinity must not be used or configured to ensure that the load balancer can balance requests across all the available nodes. As the session is already established and managed, user authentication is not required for each request, and this simplification improves the performance of each call compared with a traditional stateless call for which credentials have to be entered every time.



Implementing Stateless Load Balancing

To enable the use of a single token across multiple AIS Servers, the participating AIS Servers communicate token and session information with each other using the SSL sockets. Note that all participating AIS Servers must have network access to each other and the ports used for socket communication must be open.

The same JWT-based keystore used for the Notification and Scheduler applications is used to store and manage the SSL certificates required for SSL socket communication. The trusted node configuration that is shared with the Notification and Scheduler applications is also required for SSL socket communication.

See *Configuring EnterpriseOne HTML Server for JSON Web Token (JWT) (Release 9.2.0.5)* in the *JD Edwards EnterpriseOne Tools Security Administration Guide*

Enabling Stateless Load Balancing

It is recommended that Stateless Load Balancing be configured only for scenarios in which a single session will serve a very high volume of requests, possibly over an extended period of time. For example, in an integration scenario a third-party system or Cloud service might use a single session to continuously invoke orchestrations. Without Stateless Load Balancing, the single session is authenticated to only one AIS server, and fail-over or high-volume transactions are not easily supported. However, with Stateless Load Balancing, the single session is authenticated to a group of AIS (and HTML) servers, allowing them to scale out for high volume and fault tolerance.

Configuring Stateless Load Balancing Using Server Manager

The following three AIS Server settings are used to configure Stateless Load Balancing. These settings must be the same in all the AIS Servers that are participating in the load balancing process.

The **Security Information** section of the AIS Server in Server Manager Console contains a new section called **Stateless Load Balancing** that has these three settings:

- **Enable Stateless Load Balancing**

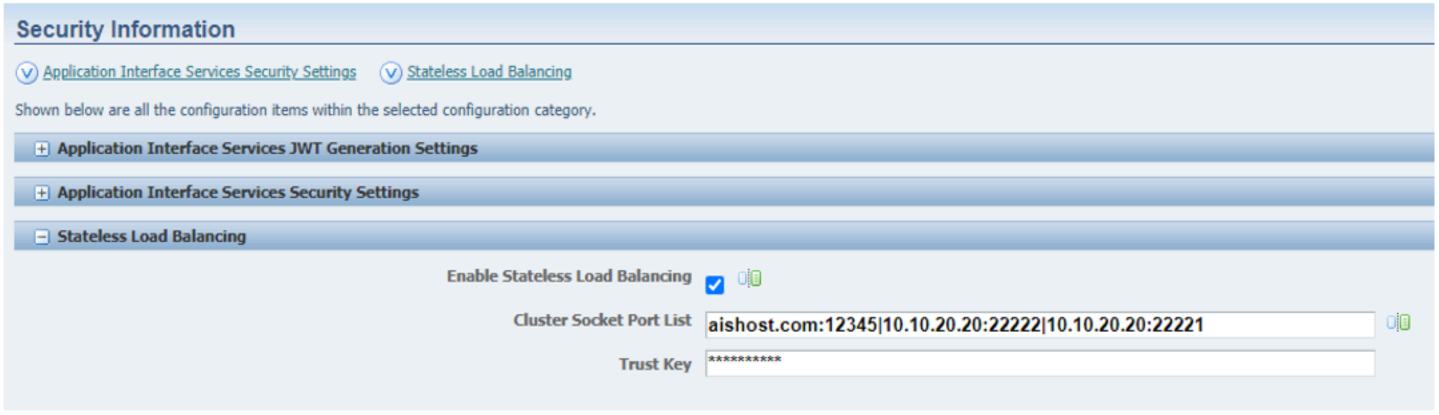
  Select this option to turn on Stateless Load Balancing.
- **Cluster Socket Port List**

  Enter a bar-delimited list of all the AIS Servers participating (including the server on which you are logged in) in the load balancing (with ports) for socket communication. For example, enter `aishost.com:12345| 10.10.20.20:22222|10.10.20.20:22221`

  > **Note:** The port number here is not the AIS HTTP port number. This number represents a unique port that is used for socket communication.
- **Trust Key**

  Enter a value that is identical for all the servers in the list enabling them to trust each other's tokens.

  This key is encrypted with the site key for additional security.



Security Information

Application Interface Services Security Settings   Stateless Load Balancing

Shown below are all the configuration items within the selected configuration category.

Application Interface Services JWT Generation Settings

Application Interface Services Security Settings

Stateless Load Balancing

Enable Stateless Load Balancing	✓
Cluster Socket Port List	aishost.com:12345|10.10.20.20:22222|10.10.20.20:22221
Trust Key	**********

Using Stateless Load Balancing

When you are coding for Stateless Load Balancing, ensure that session affinity does not exist for your request. You can avoid session affinity by not sending any cookies that may be used for session affinity (JSESSION and so on).

Generate a request for a token to the AIS Server by using either the `/v2/tokenrequest` endpoint or the `/orchestrator/jde-login` endpoint. After the token is returned, you can use it on any of the participating servers.

Use the established token to execute orchestrations (/orchestrator endpoint) or any AIS APIs on any of the AIS Servers participating in the load balancing. If you are using a load balancer, send the orchestration request to the load balancer url:port without cookies or other affinity mechanisms so that the orchestrator request can be routed to any of the participating AIS Servers.

You can also use the token to process the validation requests (`/v2/tokenrequest/validate` or `orchestrator/jde-validate-session`) on any of the participating AIS Servers. Include validation with touch to keep the session active.

To log out from the session on all the participating nodes, simply call logout (`/v2/tokenrequest/logout or /orchestrator/jde-logout`) once. This call will propagate the logout to all the nodes.

In some cases, specifically when load balancers are used, the device name may be required in the token request and in the subsequent requests. You can pass the device name in the body as `"deviceName"` or use the `jde-AIS-Auth-Device` URL parameter.

In the AIS log file, the mismatch warnings such as `Get Session Device Name Mismatch 10.111.122.122!=10.222.33.44` indicate that the AIS Server cannot match the device name based on the requests from the load balancer and therefore you must use the device name in the requests to share the token across instances.

Introducing Servers into the Cluster

If you introduce a new server into the cluster in the middle of processing, the system will automatically assign an active session to the new server. This functionality enables you to introduce additional servers as and when required to manage increased load or maintenance requirements.

If you want to introduce additional servers, ensure that the server and the socket are added in the rest.ini files for all AIS servers that are configured for load balancing.

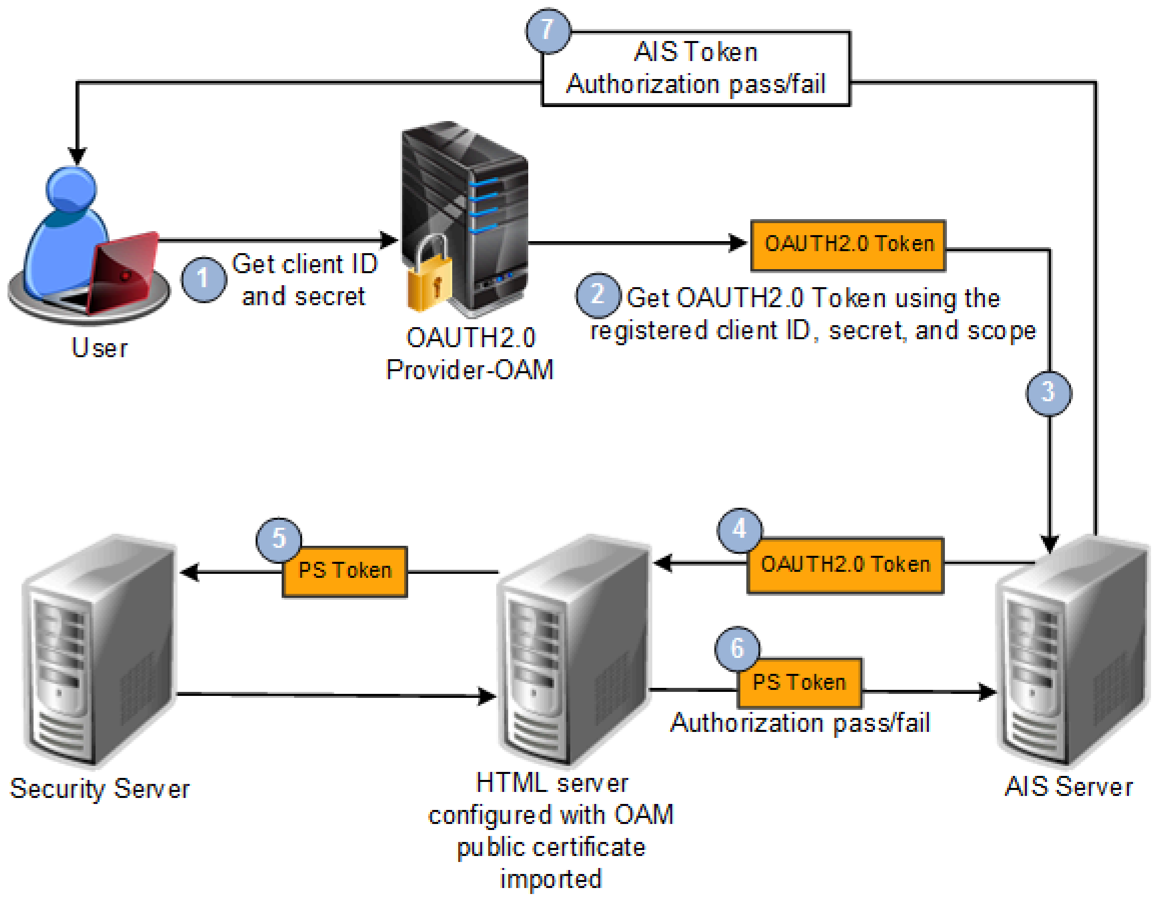
Configuring OAuth 2.0 Resource Servers with OAM 11g

You can use an OAuth 2.0 resource server to handle the authentication requests from AIS clients. This type of authentication allows access to AIS services, as well as orchestrations created using the Orchestration Studio. In this authentication process, an OAuth token is requested from an authentication provider and then passed to the AIS token request (stateful) or AIS service directly (stateless). If the AIS token request (stateful) is used, the AIS token is used for subsequent AIS calls.

To use OAuth 2.0, the format of the OAuth 2.0 token must be JWT and you must configure the steps required for JWT configuration.

See *"Configuring EnterpriseOne HTML Server for JSON Web Token (JWT) (Release 9.2.3.2)"* or *"Configuring EnterpriseOne HTML Server for JSON Web Token (JWT) (Release 9.2.0.5)"* in the *JD Edwards EnterpriseOne Tools Security Administration Guide*.

This graphic shows OAuth 2.0 authentication flow for stateful scenario.



The following steps describe the configuration and authentication flow:

1. Create an OAuth 2.0 resource server.

   See *Creating an OAuth 2.0 Resource Server* for more information.

   See *Creating an OAuth 2.0 Client* for more information.
2. Get an OAuth 2.0 token using the registered client ID, secret, domain, and scope.

   The following is an example of the parameters that you would provide to generate an OAuth 2.0 token. In this example, the value passed for Authorization parameter 'SkRFOnBsNEhBYzg5' is a base64 encoded value of client ID and secret that is separated by a colon.

   curl -i -H 'Authorization: Basic <SkRFOnBsNEhBYzg5'> -H "Content-Type: application/ x-www-form-urlencoded;charset=UTF-8" -H "X-OAUTH-IDENTITY-DOMAIN-MAME:<JDEIdentityDomain>" --request POST http://mycompany.com:14100/oauth2/rest/token -d 'grant_type=client_credentials&scope=<JDEResourceServer>'.

Parameter	Value	Description
Authorization Header	Basic <base34_clientid_secret>	This parameter contains base64 encoded values of the client ID and client secret separated by colon. The parameter is used as access token by the client.

Parameter	Value	Description
Client ID	<client_id>	This parameter is the unique API key that is generated when you register your application with OAM.
Client Secret	<client_secret>	This parameter is the private key that is generated when you register your application with OAM.
Access Token URL	ms_oauth/ oauth2/endpoints/ oauthservice/tokens	This parameter is an endpoint that is used to obtain an access token from OAM.
Grant Type	client_credentials	This parameter indicates that the REST API to be invoked is owned by the client application.
Scope	JDEResourceServer	This parameter returns all the grants provided to your application.
Domain	JDEIdentityDomain	This parameter is the Identity Domain name under which all clients and resource servers are created.

3. An OAuth 2.0 token is generated with the client ID, secret, and scope and sent in the Bearer header of an AIS token request. (Stateless requests are also supported.)
4. The AIS Server forwards the OAuth 2.0 token to the EnterpriseOne HTML Server and is configured to allow a JWT. The AIS Server forwards the OAuth 2.0 token in the Bearer if login is required.
5. The Security Server checks the PS Token with node trust, and returns an authorization response to the EnterpriseOne HTML Server. You must import the OAM certificate into the HTML Server to validate OAuth 2.0 token locally in the HTML Server. For more information, see *"Adding an Existing Certificate to a New Keystore" in the JD Edwards EnterpriseOne Tools Security Administration Guide* .
6. The EnterpriseOne HTML Server returns the authorization response to the AIS Server. The PS Token is included in the response.
7. The AIS Server returns the authorization response to the AIS client (third-party). If passed, for a token request the response includes an AIS token.

   In stateless scenario, if passed, the actual resources are provided to the user.

Creating an OAuth 2.0 Resource Server

If you have an EnterpriseOne configuration through Oracle Access Manager (OAM), then you can follow the following steps to set up an OAuth 2.0 resource server with OAM:

1. Log in to the Oracle Access Management console.

   The Launch Pad opens.
2. Click Mobile OAuth Services.
3. Select the default domain or create a domain.

4. Click the Resource Servers tab.
5. On the Custom Resource Server Configuration page, complete the following fields:

   Name

   The name of this resource server (or resource service).

   Description

   (Optional) A short description to help you or another administrator identify this resource server in the future.

   Allow Token Attributes Retrieval

   Select this option to allow custom attributes (both attribute names and values) to be shared with clients and the resource owner.

   For more information, see *"Configuring OAuth Service Profiles" in the Oracle Fusion Middleware Administrator's Guide for Oracle Access Management for All Platforms* .

   Authorization & Consent Service Plug-in

   From the menu, choose an authorization plug-in for the resource server. This plug-in type defines security policy around interactions where authorization and user consent are granted. It can influence claims in a generated token as well.

   Resource Server ID

   The unique ID created for this resource server during registration.

   Scope

   Click Add to add a new row to the scopes table.

   > **Note:** JD Edwards EnterpriseOne supports only single scope and all orchestrations are validated for single scope.

   Name

   Type a scope definition. Use dot notation, for example: photo.read.

   Description

   Type a short note that describes the scope.

   For more information, see *"Understanding the OAuth Resource Servers Configuration Page" in the Oracle Fusion Middleware Administrator's Guide for Oracle Access Management for All Platforms* .
6. Click Save.

Creating an OAuth 2.0 Client

Complete the following steps to create an OAuth 2.0 client:

1. Log in to the Oracle Access Management console.
   The Launch Pad opens.
2. Click Mobile OAuth Services.
   The OAuth Identity Domains page opens.
3. Select the default domain or create a domain.
4. Click the OAuth Clients tab.
5. To create an OAuth Web client, click the Create button located directly under the OAuth Web Clients heading and complete the following fields:

Name
The name of this OAuth client.

Description
(Optional) A short description to help you or another administrator identify this OAuth Web client in the future.

Client ID
The unique ID that the authorization server created for this client during registration. The client ID must be a valid JD Edwards EnterpriseOne user.

Allow Token Attributes Retrieval
Select this option to allow custom attributes (both attribute names and values) to be shared with resource servers and the resource owner.

Client Secret
A secret value known to the OAuth authorization service and the client. The authorization service checks the client secret and the client ID when it receives token endpoint requests from the client.

Privileges
Bypass User Consent
If selected, the client will not ask for the user's explicit authorization to access the user's protected resources.

This option must be selected for the OAuth 2.0 to work as JD Edwards EnterpriseOne supports only two legged OAuth authentication for AIS services and orchestrations.

Allowed Scopes
Lists the range of access the client has to the requested resources. To grant additional access, click Add to add a row to the table, then choose from the drop-down menu the scope to be added.

Grant Types
Client Credentials
The client requests an access token using only its client credentials.

For more information, see *"Understanding the OAuth Web Clients Configuration Page" in the Oracle Fusion Middleware Administrator's Guide for Oracle Access Management for All Platforms* .

6. Click Save.

Configuring OAuth 2.0 Resource Servers with OAM 12c

You can use an OAuth 2.0 resource server to handle the authentication requests from AIS clients. This type of authentication allows access to AIS services, as well as orchestrations created using the Orchestration Studio. In this authentication process, an OAuth token is requested from an authentication provider and then passed to the AIS token request (stateful) or AIS service directly (stateless). If the AIS token request (stateful) is used, the AIS token is used for subsequent AIS calls.

To use OAuth 2.0, the format of the OAuth 2.0 token must be JWT and you must configure the steps required for JWT configuration.

See *"Configuring EnterpriseOne HTML Server for JSON Web Token (JWT) (Release 9.2.3.2)"* or *"Configuring EnterpriseOne HTML Server for JSON Web Token (JWT) (Release 9.2.0.5)"* in the *JD Edwards EnterpriseOne Tools Security Administration Guide*.

The following steps describe the configuration and authentication flow:

1. Create an OAuth 2.0 resources.

   See *Creating an OAuth 2.0 Identity Domain* for more information.

   See*Creating an OAuth 2.0 Resource Server* for more information.

   See *Creating an OAuth 2.0 Client* for more information.

   See *Fetching Identity Domain Certificate* for more information.
2. Get an OAuth 2.0 token using the registered client ID, secret, domain, and scope.

   The following is an example of the parameters that you would provide to generate an OAuth 2.0 token. In this example, the value passed for Authorization parameter 'SkRFOnBsNEhBYzg5' is a base64 encoded value of client ID and secret that is separated by a colon.

   curl -i -H 'Authorization: Basic <SkRFOnBsNEhBYzg5'> -H "Content-Type: application/ x-www-form-urlencoded;charset=UTF-8" -H "X-OAUTH-IDENTITY-DOMAIN-MAME:<JDEIdentityDomain>" --request POST http://mycompany.com:14100/oauth2/rest/token -d 'grant_type=client_credentials&scope=<JDEResourceServer>'.

Parameter	**Value**	**Description**
Authorization Header	Basic <base34_clientid_secret>	This parameter contains base64 encoded values of the client ID and client secret separated by colon. The parameter is used as access token by the client.
Client ID	<client_id>	This parameter is the unique API key that is generated when you register your application with OAM.
Client Secret	<client_secret>	This parameter is the private key that is generated when you register your application with OAM.
Access Token URL	/oauth2/rest/token	This parameter is an endpoint that is used to obtain an access token from OAM.
Grant Type	client_credentials	This parameter indicates that the REST API to be invoked is owned by the client application.
Scope	JDEResourceServer	This parameter returns all the grants provided to your application.
Domain	JDEIdentityDomain	This parameter is the Identity Domain name under which all clients and resource servers are created.

3. An OAuth 2.0 token is generated with the client ID, secret, and scope and sent in the Bearer header of an AIS token request. (Stateless requests are also supported.)
4. The AIS Server forwards the OAuth 2.0 token to the EnterpriseOne HTML Server and is configured to allow a JWT. The AIS Server forwards the OAuth 2.0 token in the Bearer if login is required.
5. The Security Server checks the PS Token with node trust, and returns an authorization response to the EnterpriseOne HTML Server. You must import the OAM certificate into the HTML Server to validate OAuth 2.0 token locally in the HTML Server. For more information, see *"Adding an Existing Certificate to a New Keystore" in the JD Edwards EnterpriseOne Tools Security Administration Guide* .
6. The EnterpriseOne HTML Server returns the authorization response to the AIS Server. The PS Token is included in the response.
7. The AIS Server returns the authorization response to the AIS client (third-party). If passed, for a token request the response includes an AIS token.

   In stateless scenario, if passed, the actual resources are provided to the user.

Note: Before you create resources, you need to enable OAuth and OpenIDConnect Service from the OAM admin console. You can enable OAuth service from the Configuration tab under Available Services in the OAM console. If OpenIDConnect Service is not available in OAM, install the latest available bundle patch for OAM 12c.

Creating an OAuth 2.0 Identity Domain

An identity domain corresponds to the notion of a tenant. All clients and resource servers are created under an identity domain. If you have an EnterpriseOne configuration through Oracle Access Manager (OAM 12c), follow these steps stated in the Administrating Oracle Access Management to create the identity domain with OAM 12c - `Creating an Identity Domain` .

Creating an OAuth 2.0 Resource Server

If you have an EnterpriseOne configuration through Oracle Access Manager (OAM), then you can follow these steps stated in the Administrating Oracle Access Management to set up an OAuth 2.0 resource server with OAM - `Creating a Resource` .

Creating an OAuth 2.0 Client

Follow these steps stated in the Administrating Oracle Access Management to create an OAuth 2.0 client - `Creating a Client` .

Fetching Identity Domain Certificate

You need to fetch identity domain certificate when the domain is created. This is done using REST API:

1. `http://{{host}}:{{mgdport}}/oauth2/rest/security?identityDomainName={{domainname}}`

2. Extract the content from JSON response against the key 'x5c'.

   a. Create cert files with 2 signing keys.
   b. Create cert file for each certificate part from above response by placing it between -----BEGIN CERTIFICATE----- and -----END CERTIFICATE-----.
   c. Save these files.

   Note: Ensure that there are no extra spaces in the files to import successfully.

Configuring OAuth 2.0 Services with OCI Identity Access Management

You can use OAuth 2.0 services configured with OCI Identity Access Management (IAM) to handle the authentication requests from AIS clients. This type of authentication allows access to AIS services, as well as orchestrations created using the Orchestration Studio. In this authentication process, an OAuth token is requested from an authentication provider and then passed to the AIS token request (stateful) or AIS service directly (stateless). If the AIS token request (stateful) is used, the AIS token is used for subsequent AIS calls. It is important to note that before using OAuth2.0 with AIS requests the HTML server that AIS users must have OCI IAM Single Sign-On configured.

Use this procedure to set up OAuth 2.0 services with OCI IAM:

1. Configure OAuth Services for OCI IAM.

   See *Configuring SSO Support for EnterpriseOne AIS Server Clients* for more information.
2. Configure OCI IAM as an External Identity Provider for AIS Server Clients

   For more information refer the following document on *LearnJDE*:

   See *Configure OCI IAM with Microsoft Entra ID as an External IdP for SSO* for more information.

5 Managing the AIS Server Through Server Manager

Managing the AIS Server Through Server Manager

Server Manager provides settings and features that enable you to manage and monitor the AIS Server. These include:

- Configuration group settings that determine whether the Environment and Role fields and the Single Sign On option are displayed or hidden on the login screen in an EnterpriseOne mobile enterprise application. The configuration group settings also include settings for controlling session timeouts, display options, AIS Server and HTML Server communication options, security options, and logging options.

  For details of each configuration setting for the AIS Server, refer to the Server Manager internal help for each setting.
- Admin Service setting to enable administrators to clear orchestration cache in an Internet of Things environment. See *"Clearing Orchestration Cache on the AIS Server" in the JD Edwards EnterpriseOne Tools Orchestrator Guide* .
- Settings to enable caching of service request responses on the AIS Server.

  See *Caching Service Request Responses (Release 9.2.1.1)* in this guide for more information.
- Runtime metrics that show various user and server-related information, which enable you to view a list of users connected to the AIS Server and monitor user activity.

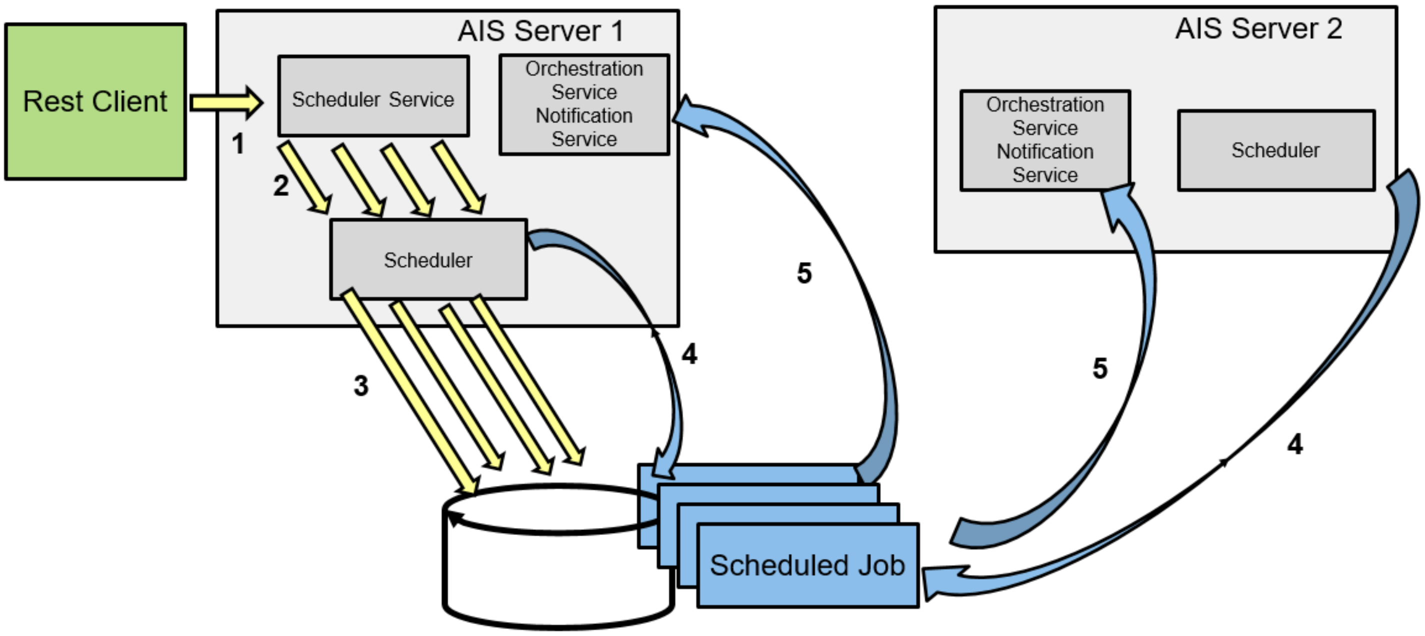
  See *"Application Interface Services Server Runtime Metrics" in the JD Edwards EnterpriseOne Tools Server Manager Guide* for more information.

6 Configuring Scheduler Resilience (Release 9.2.2.4)

Understanding Scheduler Resilience

The scheduler is used to run notifications and orchestrations at designated intervals defined by the schedule attached to each. The scheduler runs as a process on one or more AIS Servers. As notifications and orchestrations are incorporated into critical business processes, the scheduling and execution of those notifications and orchestrations becomes equally critical. The scheduler must be able to tolerate failures, restart with minimum human intervention, and load balance in that if one scheduler instance begins to fall behind, another can pick up the overflow.

Starting a notification or orchestration on a scheduler to run at some interval creates a scheduled job with a set of properties. To make the scheduler resilient, you can store the scheduled job properties for notifications and orchestrations centrally in the database. This means that if you are running a single instance of the scheduler, the scheduler can restart from a failure and continue processing from the queue of scheduled jobs without an administrator having to resubmit all jobs manually. Or if you are running multiple instances of the scheduler, the schedulers can each take scheduled jobs from the queue. If any single instance of a scheduler fails, the other schedulers can proceed independently for continuous operations.



In this graphic, the following steps occur:

1. The user requests notifications and/or orchestrations to run on their schedule using the scheduler service.
2. The scheduler service creates a job for each of the notifications and/or orchestrations.
3. The job definitions are stored in a database.
4. The scheduler on one of the AIS Servers finds a job to run when that job's scheduled interval is reached.
5. The job calls a notification or orchestration service.

Creating Tables in the Database

In order to store scheduled jobs in the database, you need to create the Quartz tables and indexes where the jobs will be stored. First, obtain the appropriate create script for Quartz-2.2.3 and the database you are using. Modify the script as necessary to create the tables in a database accessible from your AIS Server. Although these are not standard JD Edwards tables, they can be added to the JD Edwards EnterpriseOne database.

For more information on obtaining the scripts and creating these tables, see the "Setting up Quartz Scheduler Resilience to be used with Notifications and Orchestrations" document on My Oracle Support at: `https://support.oracle.com/epmos/faces/DocumentDisplay?_afrLoop=555331132725298&id=2368608.1`

Installing the JDBC Driver

The AIS Servers containing the scheduler will access the Quartz runtime database through JDBC. If your AIS Server is deployed to WebLogic and your database is Oracle, the JDBC driver will already have been installed and you can skip this step.

For more information on obtaining, uploading, and installing JDBC drivers, see the *Manage JDBC Drivers* chapter in the *JD Edwards EnterpriseOne Tools Server Manager Guide*.

Configuring AIS Server Manager Settings

You can configure the following settings for the Scheduler. You can access and change these in Server Manager under Configuration, Advanced, Miscellaneous, Scheduler Configuration. The settings are:

- **This is a Scheduler Server**. Select this option to designate this AIS server as a Scheduler server. The Scheduler server will automatically start when the associated AIS server is started. As a result, the jobs that are designated as auto-start jobs will be automatically started when the Scheduler server is started. See, Working with Scheduler (Release 9.2.4)

  The auto-start process requires the following information to ping the server until it is up before continuing. Selecting the "This is a Scheduler Server" option required you to enter the following details:

  - **Scheduler Bootstrap User**. User name used when retrieving the Scheduler auto-start jobs.
  - **Scheduler Bootstrap Password**. Password used when retrieving the Scheduler auto-start jobs. The password is encrypted in the Server Manager.
  - **Scheduler Bootstrap Environment**. Environment used when retrieving the Scheduler auto-start jobs.
  - **AIS Scheduler Instance URL**. Enter the URL for this AIS Scheduler server instance used for Scheduler auto-start processing. (Protocol://Server:Port) Protocol is http or https.
- **Scheduler Resilience**. Enable this option in order to store the runtime data in a database.
- **Number of Threads**. The number of threads allocated to the scheduler, from 1 through 100. The default value is 10.
- **Thread Priority**. The priority of scheduler threads as compared to the priority of other threaded processes on the server. Valid values are between 1 and 10. The default value is 5.
- **JDBC Max Connections**. The maximum number of connections to the scheduler database.
- **JDBC Driver Class Name**. The java class name of the database driver for the database that contains scheduler runtime data.

  Examples of driver class names are:

  - **MS SQL Server**: com.microsoft.sqlserver.jdbc.SQLServerDriver

   - **Oracle**: oracle.jdbc.driver.OracleDriver
   - **IBMi**: com.ibm.as400.access.AS400JDBCDriver

- **JDBC URL**. The URL where the database storing the Quartz tables is deployed.

  Examples of URLs are:

   - **MS SQL Server**: jdbc:sqlserver://<HOST>:<PORT>;databaseName=<DATABASE_NAME>
   - **Oracle**: jdbc:oracle:thin:@<HOST>:<PORT>:<SID> **OR** jdbc:oracle:thin:@<HOST>:<PORT>/<SERVICENAME>
   - **IBMi**: jdbc:as400://<HOST>/<SCHEMA>;libraries=<*LIBL>

- **JDBC UserName**. The user name used when creating the Quartz tables.
- **JDBC Password**. The password used when creating the Quartz tables.

The Server Manager interface provides more details on the usage of each setting. Click the "i" (Information) icon next to each in Server Manager for more information.

Scheduler Administration

Starting with Tools 9.2.4, you can start and stop the scheduled notification and orchestration jobs using the Scheduler user interface page in the Orchestrator Studio 9.2.4.0. For information on starting and stopping the notification and orchestration jobs using the Orchestrator Studio, see "Working with Scheduler" in the *JD Edwards EnterpriseOne Orchestrator Guide*.

Alternatively, you can also use a set of REST APIs for starting, stopping, and managing scheduled jobs.

Schedule REST APIs respect UDO security and feature security. When you start a notification or orchestration job, the user starting the job is authenticated to determine what is started. Only notifications and orchestrations to which that user has access are started.

There is also a REST API you can use for troubleshooting. The List API is similar to a log file in that it contains information such as number of errors, percentage of errors, exception messages, and time to execute for jobs that are currently executing. It does not contain all history like an actual log file would. It only shows the last exception message that occurred.

These are the services you can use to administer the scheduler:

Endpoint	Behavior
<URL>/jderest/v2/scheduler/start	Starts all notifications and orchestrations for which the service caller has permissions.
<URL>/jderest/v2/scheduler/stop	Stops all notifications and orchestrations. They are stopped permanently and will need to be restarted.
<URL>/jderest/v2/scheduler/startjob/ORC_1707250005TOOLS	Starts a single notification. URL ends with /<OMW NAME> (short or long).
<URL>/jderest/v2/scheduler/stopjob/ORC_1707250005TOOLS	Stops a single notification. URL ends with /<OMW NAME> (short or long).

Endpoint	Behavior
<URL>/jderest/v2/scheduler/list	Use to view runtime information about what is currently running.
<URL>/jderest/v2/scheduler/listJobs (Release 9.2.3.4)	Use to view information about the collection of Notifications and Orchestrations that are scheduled to run, are currently running, or have no schedule attached.
<URL>/jderest/v2/scheduler/ listExecuting (Release 9.2.3.4)	Use to view the attributes of all the Notifications and Orchestrations that are currently executing on the server.
<URL>/jderest/v2/scheduler/view (Release 9.2.3.4)	Use to view information about all the Notifications and Orchestrations. You can add an optional parameter of filterScheduled=true which will limit the returned data to Notifications or Orchestrations that have an attached schedule.
<URL>/jderest/v2/scheduler/startJobs (Release 9.2.3.4)	Starts a collection of Notifications and Orchestrations in the scheduler.
<URL>/jderest/v2/scheduler/stopJobs (Release 9.2.3.4)	Stops a collection of Notifications and Orchestrations in the scheduler.

Starting with Tools 9.2.4, the format of "jobname" for an orchestration and a notification is unique with the combination of:

- Notification ID or Orchestration ID
- User Name
- Environment
- Role

This enables you to execute multiple instances of a job based on the combination of Notification ID or Orchestration ID, User Name, Environment, and Role.

As a result, the services like start, startjob, startjobs, stop, stopjob, stopjobs, view, list, and listjobs will return specific jobs for each unique combination of Notification ID or Orchestration ID, User Name, Environment, and Role.

Example: For the startJob service, endpoint of the URL can be the Notification or Orchestration ID (long or short). However, the job that the service starts is unique to the Notification or Orchestration ID along with the authenticated user, environment, and role.

When you enter the endpoint of the URL as the Notification or Orchestration ID for the stopJob service, all the instances of the jobs with the specified Notification or Orchestration ID are stopped. A job name can be passed as NTF/ ORCH_ID.USER.ENV.ROLE (such as: NTF_1903210002JDE.ABC.JDV920.*ALL) to stop a specific job.

An error message is returned when you start a job that is already running. You have to manually stop the job before the job with the same Notification or Orchestration ID, user, environment, and role can be started.

A resilient scheduler can be started and stopped while not affecting the jobs that have previously been scheduled to run.

Starting and stopping scheduler instances are done with these services:

Endpoint	Behavior
<URL>/jderest/v2/scheduler/ startSchedulers	Start a collection of schedulers, one per AIS host.
<URL>/jderest/v2/scheduler/ startScheduler	Start a scheduler - the AIS server where the service is running is the scheduler host.
<URL>/jderest/v2/scheduler/ stopSchedulers	Stop a collection of schedulers, one per AIS host.
<URL>/jderest/v2/scheduler/ stopScheduler	Stops a scheduler. The AIS server where the service is running is the scheduler host.
<URL>/jderest/v2/scheduler/ pingSchedulers	Check whether a collection of schedulers are up, one per AIS host.
<URL>/jderest/v2/scheduler/ pingScheduler	Checks whether the scheduler is up. The AIS server where the service is running is the scheduler host.

All of these services use the same service body. A single schedulerHost is valid.

Service Body

```
{
             "username": "<user>",
             "password": "<password>",
             "environment": "<environment>",
             "role": "<role>",
             "scheduleIntervalMinutes": 1,
             "schedulerHosts": [{
                       "protocol": "<protocol>",
                       "host": "<AIS host>",
                       "port": "<port>"
                             },
                   {
                       "protocol": "<protocol>",
                       "host": "<AIS host 2>",
                       "port": "<port>"
                             }

                     ]
       }
```

Note: If you have not configured scheduler resilience, jobs are run on a single instance and are volatile. Stopping the AIS Server or the scheduler causes the jobs to be cleared. Without resilience, multiple schedulers should not be started because they will duplicate the notification and orchestration executions from the other scheduler instances.

For more information on REST APIs used for starting, stopping, and managing the scheduler, see *"Scheduler Service"* in the *JD Edwards EnterpriseOne Tools REST API for the Application Interface Services Server Guide*.

Troubleshooting Misfires

The AIS log displays if notifications and orchestrations are backing up. In other words, some may be running long and causing others to be placed in an exception queue where the scheduler will try to pick them up later.

Note: A job misfire is logged only if it is five minutes overdue.

Here is an example of what the message might look like:

```
13 Feb 2018 17:53:18,007 [SEVERE] - [AIS]             Scheduler Trigger for Name:
 NTF_1712060007JDE, and Group SCH_1802080001JDE (Notification/Orchestration:
 description , Schedule: description) MISFIRED. This condition is caused when
 notifications or orchestrations are not starting close to their scheduled start
 time. This may occur if other notifications or orchestrations have not yet completed,
 indicating that more jobs are scheduled than the system can complete within the schedule
 time. This can be fixed by scheduling the jobs to run less frequently, staggering the
 schedules, or by increasing the number of threads in the Scheduler, which may require
 additional system resources.
```

Some possible solutions for this issue include:

- **Decrease Frequency**. There may be jobs scheduled to run every 5 minutes that can run every 30 minutes.
- **Staggering**. Do not schedule too many jobs to run at the same time, or in multiples of the same number. Schedules that are set to run every 5, 10, 15, and 30 minutes will all attempt to run all of the notifications and orchestrations at the same time at the 30-minute mark. Using numbers such as 7, 17, 31, and 39 will balance the load better.
- **More Threads**. The scheduler uses one thread each time a job is triggered to run on its schedule. If more jobs are triggered to run than there are available threads, a group of jobs will be picked up, one for each thread while the other jobs wait for their own thread. These other jobs will wait for a thread to be returned by a job that has completed execution. Increasing the threads will allow more jobs to run at one time, but might tax system resources with the added activity.
- **Clustering**. Additional schedulers can pick up jobs that might otherwise become misfires. If the first scheduler has 10 threads, misfires could result when more than 10 jobs are scheduled to run at one time. Adding a second scheduler with 10 threads doubles the number of simultaneous jobs that can be running. There is no explicit load balancing between the two schedulers. The first one to examine scheduled jobs runs as many as it can. Only the remaining scheduled jobs are picked up by the next scheduler. In the preceding example of two schedulers with 10 threads each, if there are 12 jobs that are running on the same schedule, the first server to acquire a set of jobs might acquire 10, leaving the remaining 2 for the next scheduler. Reducing the threads on the first server to 8 might mean that 4 jobs are picked up by the next scheduler. Multiple schedulers running at the same time do not know about each other, so attempting to load balance across schedulers is an administrative task. Also, if the two AIS Servers point to the same HTML Server, little or no benefit will be gained.

Testing Your Configuration

Testing the configuration requires tools such as Postman, SoapUI, or cURL to run RESTful services. This guide does not provide training in running the web services.

After the configuration is complete, the next step is to test that the scheduler is running. There is a RESTful web service that submits a job and validates that it is being picked up from the database.

Start Cluster Unit Test Service

The End Point for this example is:

```
<URL>/jderest/v2/scheduler/startClusterUnitTest
```

In this example, <URL> is the AIS Server where the scheduler runs.

In the following example, replace the values that look like <xxxx> with the appropriate values.

Service Body

```
{
          "username": "<user>",
          "password": "<password>",
          "environment": "<environment>",
          "role": "<role>",
          "scheduleIntervalMinutes": 1,
          "schedulerHosts": [{
                "protocol": "<protocol>",
               "host": "<AIS host>",
               "port": "<port>",
                    }
          ]
}
```

For more information on REST APIs used for starting, stopping, and managing the scheduler, see *"Scheduler Service"* in the *JD Edwards EnterpriseOne Tools REST API for the Application Interface Services Server Guide* guide.

List Service

This job will be picked up each minute by the scheduler running on the AIS Server. Using the list service, you can see the job running.

The End Point for this example is:

```
<URL>/jderest/v2/scheduler/list
```

In this example, <URL> is the AIS Server where the scheduler runs.

Service Body

```
{
```

```
curl -X POST -H "Content-Type:application/json" http://ais_server_url/jderest/v2/
scheduler/list -d {
 "token": "044QF2SLgaM6vZX081eq8KsVi6XcJiiFL5un5ACH
+eBGUg=MDE5MDEyMTY4NzY4NjcwMjI2NzExNzcyNDEwLjE1OS45OS43MzE0NzkxNDU4NDM4ODU="
}
```

Return Body

```
{
     "jobs": [
        {
            "jobLastExecutionBaseUrl": "http://an_ais_host:7101",
            "jobEndpointRequested": "UnitTest_a",
            "serviceShortEndpoint": "UnitTest_a",
            "serviceLongEndpoint": "Unit Test a",
            "omwServiceDescription": "Job For Unit Test ID = a",
            "omwScheduleDescription": "Run every 1 minutes",
            "scheduleIntervalMinutes": 1,
            "jobname": "a",
            "jobgroup": "DEFAULT"
        }
    ],
"scheduler": {
     "isResilient": false,
     "isStarted": true,
     "runningSinceUTC": "2019-03-28T18:08:35.305Z",
     "runningSinceUserLocale": "03/28/2019 12:08:35 PM",
     "schedulerInstanceJobsExecuted": 13
  }
}
```

Note: Irrelevant values have been removed from the return body.

List Jobs Service (Release 9.2.3.4)

This service lists the status of one or more notifications and orchestrations that are scheduled to run.

The End Point for this example is:

```
<URL>/jderest/v2/scheduler/listJobs
```

In this example, <URL> is the AIS Server where the scheduler runs.

Service Body

```
{
    "username": "{{username}}",
    "password": "{{password}}",
    "environment": "{{environment}}",
    "role": "{{role}}",
    "schedulerServices":
    [
        {
            "serviceEndpoint" : "{{JOBA}}"
        },
        {
            "serviceEndpoint" : "{{JOBB}}"
        }
    ]
}
```

Return Body

```
{
    "scheduledInSchedulerCount": 2,
    "notScheduledInSchedulerCount": 0,
    "scheduledInScheduler": [
        {
            "jobname": "NTF_1903210002JDE.USER.JDV920.*ALL",
            "jobgroup": "NTF_1903210002JDE",
            "jobVersion": "V2",
            "jobStartTimeUserLocale": "06/17/2019 11:20:41 AM",
            "jobStartTimeUTC": "2019-06-17T17:20:41.124Z",
            "jobNextFireTimeUserLocale": "06/17/2019 11:20:41 AM",
            "jobNextFireTimeUTC": "2019-06-17T17:20:41.124Z",
            "jobNextFireTime": "2019-06-17 11:20:41 124",
            "jobLastExecutionTimeMillis": 0,
            "jobTotalExecutions": 0,
            "jobTotalCompletions": 0,
            "jobTotalErrors": 0,
            "jobConsecutiveErrors": 0,
            "jobPercentageErrors": 0,
            "udoServiceType": "NOTIFICATION",
            "jobEndpointRequested": "NTF_1903210002JDE",
            "omwServiceDescription": "\"A\" Search Type LI AB LI WL 1 and 2",
            "jobServicePath": [
                "jderest",
                "v2",
                "notification"
            ],
            "serviceShortEndpoint": "NTF_1903210002JDE",
            "serviceLongEndpoint": "Search Type LI AB LI WL  1 and 2",
            "jobLastExecutionBaseUrl": "http://127.0.0.1:7101",
            "jobBaseUrl": "http://127.0.0.1:7101",
            "omwScheduleDescription": "LISTS JOBS",
            "scheduleCronString": "0 0/1 * 1/1 * ? *",
            "scheduleIntervalMinutes": -1,
            "userName": "USER",
            "environment": "JDV920",
            "role": "*ALL"
        },
        {
            "jobname": "NTF_1903210003JDE.USER.JDV920.*ALL",
            "jobgroup": "NTF_1903210003JDE",
            "jobVersion": "V2",
            "jobStartTimeUserLocale": "06/17/2019 11:20:41 AM",
            "jobStartTimeUTC": "2019-06-17T17:20:41.214Z",
            "jobNextFireTimeUserLocale": "06/17/2019 11:20:41 AM",
            "jobNextFireTimeUTC": "2019-06-17T17:20:41.214Z",
            "jobNextFireTime": "2019-06-17 11:20:41 214",
            "jobLastExecutionTimeMillis": 0,
            "jobTotalExecutions": 0,
            "jobTotalCompletions": 0,
            "jobTotalErrors": 0,
            "jobConsecutiveErrors": 0,
            "jobPercentageErrors": 0,
            "udoServiceType": "NOTIFICATION",
            "jobEndpointRequested": "NTF_1903210003JDE",
            "omwServiceDescription": "\"B\" Search Type LI AB LI WL 2 and 3",
            "jobServicePath": [
                "jderest",
                "v2",
                "notification"
            ],
            "serviceShortEndpoint": "NTF_1903210003JDE",
            "serviceLongEndpoint": "Search Type LI AB LI WL 2 and 3",
            "jobLastExecutionBaseUrl": "http://127.0.0.1:7101",
            "jobBaseUrl": "http://127.0.0.1:7101",
            "omwScheduleDescription": "LISTS JOBS",
            "scheduleCronString": "0 0/1 * 1/1 * ? *",
            "scheduleIntervalMinutes": -1,
            "userName": "USER",
            "environment": "JDV920",
            "role": "*ALL"
        }
    ],
    "notScheduledInScheduler": [],
    "scheduler": {
        "isResilient": false,
        "isStarted": false,
        "schedulerInstanceJobsExecuted": 0
    }
}
```

List Executing Service (Release 9.2.3.4)

This service displays the attributes of all the notifications and orchestrations that are currently executing on the server.

The End Point for this example is:

```
<URL>/jderest/v2/scheduler/listExecuting
```

In this example, <URL> is the AIS Server where the scheduler runs.

Service Body

```
{
    "username": "{{username}}",
    "password": "{{password}}",
    "environment": "{{environment}}",
    "role": "{{role}}",
    "schedulerServices":
    [
        {
            "serviceEndpoint" : "{{JOBA}}"
        },
        {
            "serviceEndpoint" : "{{JOBB}}"
        }
    ]

}
```

Return Body

```
{
    "executingJobs": [
        {
            "serviceShortEndpoint": "NTF_1903210001JDE",
            "serviceLongEndpoint": "Search Type LI AB LI WL 2 and 4",
            "omwServiceDescription": "\"C\" Search Type LI AB LI WL 2 and 4",
            "startTime": "2019-06-17T17:20:41.273Z",
            "fireTime": "2019-06-17 11:38:14 220",
            "fireTimeUTC": "2019-06-17T17:38:14.220Z",
            "fireTimeUserLocale": "06/17/2019 11:38:14 AM",
            "executionMillis": 9409,
            "exectionTime": "00:00:09.409"
        },
        {
            "serviceShortEndpoint": "NTF_1903210003JDE",
            "serviceLongEndpoint": "Search Type LI AB LI WL 2 and 3",
            "omwServiceDescription": "\"B\" Search Type LI AB LI WL 2 and 3",
            "startTime": "2019-06-17T17:20:41.214Z",
            "fireTime": "2019-06-17 11:38:14 220",
            "fireTimeUTC": "2019-06-17T17:38:14.220Z",
            "fireTimeUserLocale": "06/17/2019 11:38:14 AM",
            "executionMillis": 9410,
            "exectionTime": "00:00:09.410"
        },
        {
            "serviceShortEndpoint": "NTF_1903210002JDE",
            "serviceLongEndpoint": "Search Type LI AB LI WL  1 and 2",
            "omwServiceDescription": "\"A\" Search Type LI AB LI WL 1 and 2",
            "startTime": "2019-06-17T17:20:41.124Z",
            "fireTime": "2019-06-17 11:38:14 198",
            "fireTimeUTC": "2019-06-17T17:38:14.198Z",
            "fireTimeUserLocale": "06/17/2019 11:38:14 AM",
            "executionMillis": 9433,
            "exectionTime": "00:00:09.433"
        }
    ]
}
```

View Service (Release 9.2.3.4)

This service displays all the notifications and orchestrations that you have UDO permissions to. This enables you to determine whether a job is scheduled and also to find out whether a job is currently running. Using filterScheduled=true parameter, you can view only the notifications and orchestration that have a schedule attached.

The End Point for this example is:

```
<URL>/jderest/v2/scheduler/view
```

In this example, <URL> is the AIS Server where the scheduler runs.

Service Body

```
curl -X POST -H "Content-Type:application/json" http://ais_server_url/jderest/v2/
scheduler/view filterScheduled=true -d
{
```

```
"token": "044QF2SLgaM6vZX081eq8KsVi6XcJiiFL5un5ACH
+eBGUg=MDE5MDEyMTY4NzY4NjcwMjI2NzExNzcyNDEwLjE1OS45OS43MzE0NzkxNDU4NDM4ODU="
}
```

Return Body

```
{
"scheduler": {
"isResilient": true,
"isStarted": true,
"runningSinceUTC": "2019-03-04T20:23:37.613Z",
"schedulerInstanceJobsExecuted": 11
},
"services": [{
"environment": "JDV920",
"role": "*ALL",
"deviceName": "127.0.0.1",
"ssoEnabled": false,
"username": "USER",
"jobEndpointRequested": "NTF_1808170001TOOLS",
"scheduleIntervalMinutes": -1,
"jobServicePath": [
"jderest",
"v2",
"notification"
],
"scheduleCronString": "0 0/1 * 1/1 * ? *",
"omwScheduleDescription": "0 0/1 * 1/1 * ? *",
"omwScheduleObjectName": "SCH_1808170001TOOLS",
"omwScheduleLongName": "Cron Minutely",
"omwScheduleOwner": "USER",
"schedule": {
"name": "Cron Minutely",
"description": "0 0/1 * 1/1 * ? *",
"omwObjectName": "SCH_1808170001TOOLS",
"udoOwner": "USER",
"cronString": "0 0/1 * 1/1 * ? *",
"intervalType": "cronString"
},
"omwServiceObjectName": "NTF_1808170001TOOLS",
"omwServiceOwner": "USER",
"omwServiceDescription": "UserMailingType",
"serviceShortEndpoint": "NTF_1808170001TOOLS",
"serviceLongEndpoint": "UserMailingType",
"udoServiceType": "NOTIFICATION",
"serviceScheduledInScheduler": true,
"jobInScheduler": {
"jobname": "NTF_1808170001TOOLS.USER.JDV920.*ALL",
"jobgroup": "NTF_1808170001TOOLS",
"jobVersion": "V2",
                "jobStartTimeUserLocale": "03/26/2019 02:40:50 PM",
                "jobStartTimeUTC": "2019-03-26T20:40:50.843Z",
                "jobNextFireTimeUserLocale": "03/26/2019 02:42:00 PM",
                "jobNextFireTimeUTC": "2019-03-26T20:42:00.000Z",
                "jobNextFireTime": "2019-03-26 14:42:00 0",
"jobTotalExecutions": 86,
"jobTotalCompletions": 86,
"jobTotalErrors": 0,
"jobConsecutiveErrors": 0,
"jobPercentageErrors": 0,
"udoServiceType": "NOTIFICATION",
"jobEndpointRequested": "NTF_1808170001TOOLS",
"omwServiceDescription": "UserMailingType",
"jobServicePath": [
"jderest",
"v2",
"notification"
],
"serviceShortEndpoint": "NTF_1808170001TOOLS",
"serviceLongEndpoint": "UserMailingType",
"jobLastExecutionBaseUrl": "http://127.0.0.1:7101",
"jobBaseUrl": "http://127.0.0.1:7101",
"omwScheduleDescription": "0 0/1 * 1/1 * ? *",
"scheduleCronString": "0 0/1 * 1/1 * ? *",
"scheduleIntervalMinutes": -1,
"userName": "USER",
"environment": "JDV920",
"role": "*ALL"
},
"executingInSchedulerInstance": {
"serviceShortEndpoint": "NTF_1808170001TOOLS",
"serviceLongEndpoint": "UserMailingType",
"omwServiceDescription": "UserMailingType",
"startTimeUTC": "2019-03-04T16:12:29.124Z",
"fireTime": "2019-03-04 13:32:08 434",
"fireTimeUTC": "2019-03-04T20:32:08.434Z",
"exectionTime": "00:00:48.740",
"executionMillis": 48740
}
]
}
```

Analyze Unit Test State Service

To view the results, verifying a successful configuration, run the analyzeUnitTestState RESTful web service.

The End Point for this example is:

```
<URL>/jderest/v2/scheduler/analyzeUnitTestState
```

In this example, <URL> is the AIS Server where the scheduler runs.

In the service body, replace the values that look like <xxxx> with the appropriate values.

Service Body

```
{
          "username": "<user>",
          "password": "<password>",
          "environment": "<environment>",
          "role": "<role>",
}
```

Return Body

```
{
    "unitTestResults": {
        "testSucceeded": true,
        "errorEventIndex": [],
        "errorEvent": [],
        "unitTestEvents": [
            {
                "environment": "JDV920",
                "role": "*ALL",
                "jasserver": null,
                "token": null,
                "deviceName": null,
                "ssoEnabled": false,
                "ssoUniqueId": null,
                "psToken": null,
                "username": "jde",
                "jasProcessingInstruction": null,
                "unitTestJobBehavior": {
                    "unitTestStartupHost": {
                        "protocol": "http",
                        "host": "an_ais_host_1",
                        "port": "7101"
                    },
                    "id": "a",
                    "unitTestHostActions": [],
                    "unitTestHostCount": 1,
                    "runningOnSingleServer": true
                },
                "executionHost": "http:// an_ais_host_1:7101"
            },
            {
                "environment": "JDV920",
                "role": "*ALL",
                "jasserver": null,
                "token": null,
                "deviceName": null,
                "ssoEnabled": false,
                "ssoUniqueId": null,
                "psToken": null,
                "username": "jde",
                "jasProcessingInstruction": null,
                "unitTestJobBehavior": {
                    "unitTestStartupHost": {
                        "protocol": "http",
                        "host": " an_ais_host_2",
                        "port": "7101"
                    },
                    "id": "a",
                    "unitTestHostActions": [],
                    "unitTestHostCount": 1,
                    "runningOnSingleServer": true
                },
                "executionHost": "http:// an_ais_host_1:7101"
            }
        ],
        "jobsStopped": null
    }
}
```

The most important value in this return body is the "testSucceeded": "true" pair. Everything else describes details of the successful executions of the unit test by the scheduler. In this case, there are two "unitTestEvents," meaning that the unit test ran twice. Because it is successful, your scheduler resilience is properly configured. If a different payload is returned or the testSucceeded is false, check the logs for that AIS Server.

When you have completed your testing, stop the server using the stop RESTful web service described previously.

Configuring More Scheduler Instances

To configure more scheduler instances, follow the instructions for configuring the first instance on a separate AIS host machine.

Start the new scheduler using the REST APIs mentioned in *Scheduler Administration*.

Follow the same steps for testing a single instance, but change the list of hosts to include the new host. All the server clocks should be synchronized.

The End Point for this example is:

```
<URL>/jderest/v2/scheduler/startClusterUnitTest
```

In this example, <URL> is the AIS Server where the scheduler runs.

In the following example, replace the values that look like <xxxx> with the appropriate values.

Service Body

```
{
          "username": "<user>",
          "password": "<password>",
          "environment": "<environment>",
          "role": "<role>",
          "scheduleIntervalMinutes": 1,
          "schedulerHosts": [{
                "protocol": "<protocol>",
               "host": "<AIS host>",
               "port": "<port>",
                     },
{
                "protocol": "<protocol>",
               "host": "<AIS host 2>",
               "port": "<port>",
                     }
          ]
}
```

The results should be the same as with a single instance, except the "ID" values will iterate through the alphabet -- one sequential letter for each scheduler instance, i.e. unitTestEvent ID "a", then "b", then "c" for three hosts. Make sure that all previous unit test jobs have been stopped using the /stop service or the /stopjob service mentioned previously.

7 Using the AIS Client Class Generator

Understanding Generating Objects with the AIS Client Class Generator

The AIS Client Class Generator generates Java classes for use in Java-based AIS clients. The AIS Client Class Generator is an extension to JDeveloper and must be installed before using it.

A form service call to the AIS Server, also referred to as a form service request, results in a response that contains a string in JSON format. In JDeveloper, you can access and use the AIS Client Class Generator to transform the response into object form because objects are easier to work with than strings. The AIS Client Class Generator generates classes matching the form. The generator is also able to generate classes for data service responses.

If you are building a mobile application using Oracle MAF, in the AIS Client Class Generator, choose the option to generate classes for a mobile application.

Installing the AIS Client Class Generator

The AIS Client Class Generator is JDeveloper extension. To install the extension:

1. In JDeveloper, select the **Help** menu, **Check for Updates**.
2. Click **Next**.
3. Select **Install From Local File**, and then enter the location of the AISCGE 12c_v.x.x.zip file.

   This file is included with the download package for the AIS client that you are using. If you have not downloaded this package yet, see the "Developer's" guide for the AIS client you are using for information on how to obtain this download.
4. Click **Next**, and then click **Finish**.

   JDeveloper closes automatically.

Configuring the AIS Client Class Generator

To configure the AIS Client Class Generator:

1. In JDeveloper, access Preferences:

   On Microsoft Windows, select the **Tools** menu, **Preferences**.

   On Apple MacOS, select the **JDeveloper** menu, **Preferences**.
2. Select **AIS Client Class Generator**.

3. On Preferences, complete the following fields to specify the AIS Server location and AIS Server information:
   - **AIS Server URL**. This is a fully qualified URL to the AIS Server that includes the protocol, server, and port number. For example: http://myaisserver.com:8474)
   - **JAS Server URL**. (Optional) This is the URL to the EnterpriseOne HTML Server. Enter a URL only if you want to override the JAS Server URL configured on the AIS Server.)
   - **Username**. Enter a JD Edwards EnterpriseOne user name.
   - **Password**. Enter a JD Edwards EnterpriseOne user password.
   - **Environment**. (Optional) Enter a value only if you want to override the value configured on the AIS Server.
   - **Role**. (Optional). Enter a value only if you want to override the value configured on the AIS Server.
   - **JSON Files Folder**. The directory for storing the JSON files. The default location is the AISClientClassGenerator\input directory.)
   - **Default Java Classes Folder**. The directory where the generated form classes are stored, if an active project is not selected.

     The AIS Client Class Generator uses this folder only when it is run without a project open in JDeveloper. When a project is open in JDeveloper, the generator stores the Java files in the source directory for the project at the defined package path or the default package path which is `com.oracle.e1.formservicetypes.`
   - **Java Package**. Enter the name of the Java package assigned to the generated form classes. This will also determine the folder structure for the Java classes in the project src folder.
4. Click **OK**.

Generating Data Classes Based on a Form

Use the AIS Client Class Generator to generate data classes for an EnterpriseOne form. In the AIS Client Class Generator, you supply the service request information.

Note: The AIS Client Class Generator supports form interconnects only; it does not support form events.

To use the AIS Client Class Generator to generate data classes for a form service request:

1. In JDeveloper, select the ApplicationController project. For ADF application development, select the **Model** project in your ADF application workspace.

   JDeveloper will save the classes generated by the AIS Client Class Generator in this location.
2. Select the **Tools** menu and then select **AIS Client Class Generator**.
3. Click the Form Service radio button. (Available in AIS Client Class Generator v1.6.2.)
4. On AIS Client Class Generator, complete the following fields to supply the service request information:
   - **Username**. This contains the default value entered in the preferences.
   - **Password**. This contains the default value entered in the preferences.
   - **Environment**. This contains the default value entered in the preferences.
   - **Role**. This contains the default value entered in the preferences.
   - **Application Name**. Enter the name of the EnterpriseOne application.
   - **Form Name**. Enter the name of the EnterpriseOne application form.

- **Version**. (Optional) Enter the version name. If you leave it blank, the generator will use ZJDE0001 by default.
- **MaxPageSize**. (Optional)
- **ReturnControlIDs**. (Optional) Use this field to specify the exact fields on the form that you want generated. The return control IDs can specify hidden fields or a subset of fields.
- **FormInputs**. (Optional)
- **FormServiceAction**. Enter the action to be performed. Valid values include: Create, Read, Update, Delete.
- **FindOnEntry**. (Optional)
- **DemoMode**. (Optional, but recommended) This ensures at least one grid row is present, so grid classes are generated even if there is no data in the database.
- **Generate for Mobile Application**. Select this check box for mobile applications only. For ADF and AIS client applications, make sure that this check box is cleared or not checked.
- **Output Version**. Select Version 1, Version 2, or Grid Data to generate classes based on the output type. The Grid Data option is available starting with AIS Client Class Generator v2.2.0.

  For more information about output types, see *"Additional Supported Output Types for Form Service and Data Service" in the JD Edwards EnterpriseOne Application Interface Services Client Java API Developer's Guide* .

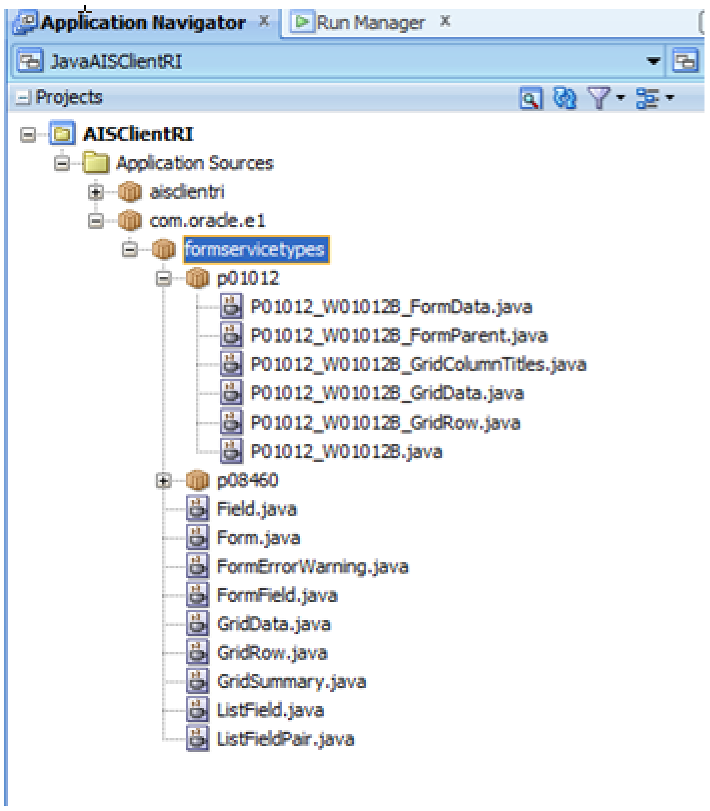
5. If you want to preview and keep the JSON files, select the **Preview JSON Data** and **Keep JSON Files** check boxes.
6. Click the **Generate** button to generate the JSON, and then verify that it has the fields and records you need.
7. Click **Continue** to generate the Java files.

   If successful, a confirmation message appears that shows the location of the JSON files and Java class files.
8. Highlight the Application Controller project (or Model project for ADF applications) and then click the "**refresh**" button to display the new files.

   The AIS Client Class Generator displays a dialog box that shows where the classes are saved.

Example of Classes Generated from the AIS Client Class Generator

This example shows the generated classes for form W01012B in application P01012.



Notice that the structure of the generated classes in JDeveloper represent the EnterpriseOne form. The form fields are in the class P01012_W01012B_FormData; the grid data is also within that form. Inside the P01012_W01012B_GridData class is a rowset that contains the grid records. Each row in the rowset is from the P01012_W01012B_GridRow class, which is where all the columns are listed.

JSON string responses for the P01012_W01012B form can now be de-serialized into these classes.

Additional Grid Columns Added to the _GridRow Class

If you need any fields on the form that have not been generated, you can add them manually. The code in this example shows additional hidden grid columns added to the _GridRow class.

```
Field sAddressLine1_40  = new Field();
Field sCity_44  = new Field();
Field SPrefix_81  = new Field();
Field sPhoneNumber_46  = new Field();

public void setsAddressLine1_40(Field sAddressLine1_40)
{
this.sAddressLine1_40 = sAddressLine1_40;
}
public Field getSAddressLine1_40()
{
return sAddressLine1_40;
}
public void setsCity_44(Field sCity_44)
{
```

```
this.sCity_44 = sCity_44;
}
public Field getSCity_44()
{
return sCity_44;
}
public void setsPrefix_81(Field SPrefix_81)
{
this.SPrefix_81 = SPrefix_81;
}
public Field getSPrefix_81()
{
return SPrefix_81;
}
public void setsPhoneNumber_46(Field sPhoneNumber_46)
{
this.sPhoneNumber_46 = sPhoneNumber_46;
}
public Field getSPhoneNumber_46()
{
return sPhoneNumber_46;
}
```

Generating Data Classes Based on a Data Request (AIS Client Class Generator v1.6.2)

The dataservice endpoint on the AIS Server enables AIS clients to receive responses from EnterpriseOne that contain either a count or a list of records matching a query of a table or view. You can use the AIS Client Class Generator to generate data classes based on the data request.

To use the AIS Client Class Generator to generate data classes based on the data request:

1. In JDeveloper, select the **ApplicationController** project.

   JDeveloper will save the classes generated by the AIS Client Class Generator in this location.
2. Select the **Tools** menu, **AIS Client Class Generator**.
3. Click the **Data Service** radio button. (Available in AIS Client Class Generator v1.6.2.)
4. On AIS Client Class Generator, complete the following fields to supply the service request information:
   - **Username**. This contains the default value entered in the preferences.
   - **Password**. This contains the default value entered in the preferences.
   - **Environment**. This contains the default value entered in the preferences.
   - **Role**. This contains the default value entered in the preferences.
   - **Target Type**. This is the table or view based on the object on which the query is performed.
   - **Target Name**. The object name to be queried, for example `F0101` or `V0101A`.
   - **ReturnControlIDs**. (Optional) Use this field to specify the exact fields on the form that you want generated. Specify fields by *Table.Column*, for example F0101.AN8 and F0101.ALPH.
   - **MaxPageSize**. (Optional)
   - **FindOnEntry**. (Optional)

- **DemoMode**. (Optional, but recommended) This ensures at least one grid row is present, so grid classes are generated even if there is no data in the database.
- **Output Version**. Select Version 1, Version 2, or Grid Data to generate classes based on the output type. The Grid Data option is available starting with AIS Client Class Generator v2.2.0.

  For more information about output types, see *"Additional Supported Output Types for Form Service and Data Service" in the JD Edwards EnterpriseOne Application Interface Services Client Java API Developer's Guide* .

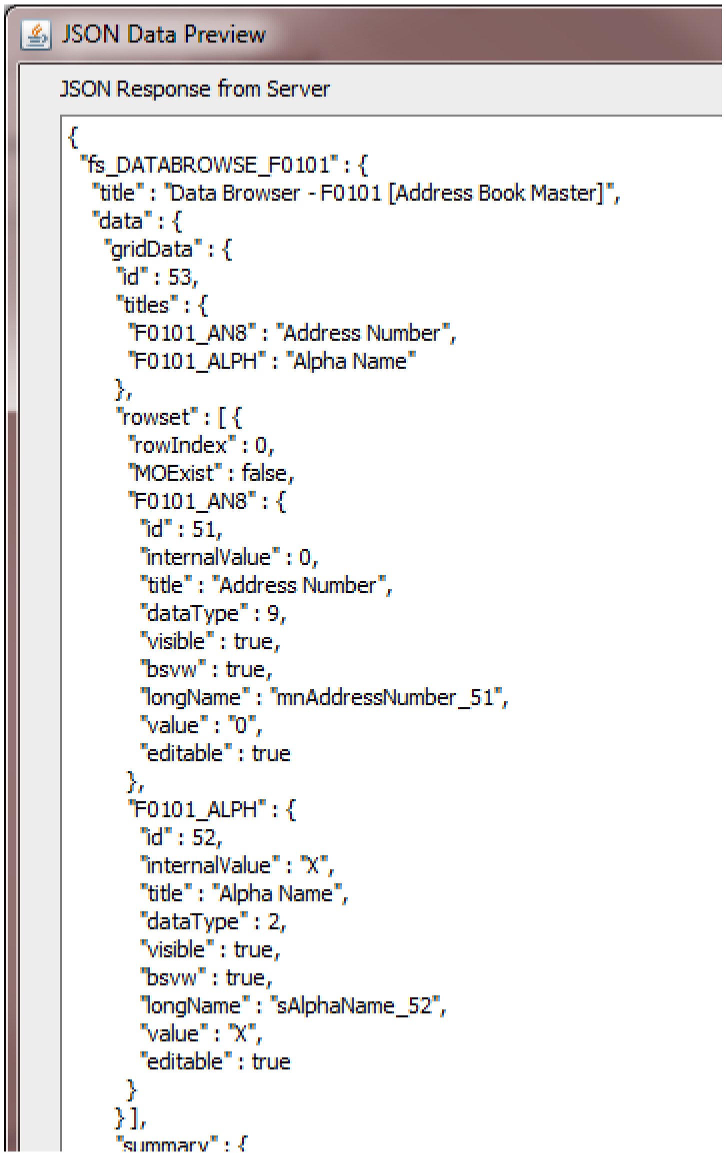
5. Make sure to select the **Preview JSON Data** and **Keep JSON Files** check boxes if you want to preview and keep the JSON files.
6. Click the **Generate** button to generate the JSON, and then in the preview, verify that it has the fields and records you need.
7. Click **Continue** to generate the Java files.

   If successful, a confirmation message appears that shows the location of the JSON and Java class files.
8. Click **OK** and close the generator.
9. Highlight the Application Controller project and then click the "**refresh**" button to display the new files.

   The AIS Client Class Generator displays a dialog box that shows where the classes are saved.

Example of Classes Generated from the AIS Client Class Generator Data Request

This example shows the generated classes for data in the F0101_AN8 and F0101_ALPH columns.



```
{
 "fs_DATABROWSE_F0101" : {
  "title" : "Data Browser - F0101 [Address Book Master]",
  "data" : {
   "gridData" : {
    "id" : 53,
    "titles" : {
     "F0101_AN8" : "Address Number",
     "F0101_ALPH" : "Alpha Name"
    },
    "rowset" : [ {
     "rowIndex" : 0,
     "MOExist" : false,
     "F0101_AN8" : {
      "id" : 51,
      "internalValue" : 0,
      "title" : "Address Number",
      "dataType" : 9,
      "visible" : true,
      "bsvw" : true,
      "longName" : "mnAddressNumber_51",
      "value" : "0",
      "editable" : true
     },
     "F0101_ALPH" : {
      "id" : 52,
      "internalValue" : "X",
      "title" : "Alpha Name",
      "dataType" : 2,
      "visible" : true,
      "bsvw" : true,
      "longName" : "sAlphaName_52",
      "value" : "X",
      "editable" : true
     }
    } ],
    "summary" : {
```

8 Remapping Fields for Customized EnterpriseOne Forms

Overview

If an AIS client invokes an EnterpriseOne form that has been customized, the AIS client might return unexpected data or not function. Customizations that can make an AIS client inoperative include changes to an EnterpriseOne form in Form Design Aid (FDA) or fields modified with global data dictionary overrides—overrides with no specified jargon or language.

To resolve this issue, Oracle provides a comparison utility called the JDE JSON Mapping Tool. This tool enables you to compare and identify mismatched fields between an AIS client and EnterpriseOne, and remap the fields through AIS endPoint mappings.

Note: ADF applications use ALIAS naming which eliminates mismatched fields due to Data Dictionary overrides.

How AIS endPoint Mappings Work

AIS clients are developed based on the JSON output returned from a specific EnterpriseOne environment. For AIS clients to function properly, the client must receive JSON responses from the AIS Server (server JSON) that match the expected responses in the client. This is referred to as the client contract.

To generate modified JSON output, the system uses an AISEndPoint.xml file deployed to the EnterpriseOne AIS Server. The AISEndPoint.xml file contains ID mappings in endPoint elements. The ID mappings map the fields expected in the AIS client to the fields being sent by the AIS Server for each EnterpriseOne form.

Note: The AISEndpoint.xml file is included in the AIS deployment object (war file).

In the AISEndPoint.xml file, the endPoint mapping is organized in a hierarchy that is equal to the hierarchy of an EnterpriseOne application form, which follows this order:

1. Form inputs for the main form or the parent form.
2. Any additional forms, which can be the main form or subforms.
3. Any form data controls, form action controls, or a grid within a form.
4. Any columns within a grid.

The AIS Server uses an endPoint mapping to transform the IDs used for both the input and output of a form service request. For input, the requested form IDs and all the IDs used in any grid or form action are replaced with values the server is expecting. For output, all of the IDs within the JSON response for the form are replaced with values expected by the AIS client. Hence the "server" and "client" attributes in the mapping.

Transformations occur only when the application and form listed in the endPoint "module" parameter in the endPoint mapping match the application and form ID requested in the "formName" parameter of the service call. The module might not match the value in the "appOID" parameter. In this case, if the module matches the requested form name, the AIS Server will execute the form defined in the appOID, calling a different form than the form listed in the "formName" parameter.

The following example shows an endPoint mapping in the AISEndPoint.xml file.

Example of an AIS Endpoint Mapping

```
<processingOptions appIdClient="P87NLPF1" appIdServer="P87NLPF1">
  <option longClient="dtSDFromDate1_34" longServer="dtSDFromDate_3"/>
    <option longClient="nInteger015_55" longServer="nInteger01_5"/>
  </processingOptions>

<endPoint module="P87TEST_W87NLPFA" appOID="P87NLPF_W87NLPFA" type="JAS">
  <mapping>
   <!--Form Inputs - both server and client required -->
     <formInput client="3" server="1"/>
     <formInput client="4" server="2"/>

<!--Forms - both server and client required, 0 is the id of the parent form for power
 forms -->
      <form client="0" server="0">
<!--controlData - both short and long names required with ids -->
            <controlData longClient="txtEnterpriseOneEventPoint01_46"
                             longServer="txtEnterpriseOneEventPoint01_36"/>
            <!--controlAction - both client and server ids required -->
            <controlAction client="95" server="15"/>
<!--Grid columns - both server and client required, both short and long names required -->
<column longClient="sPhoneType_89"
                            longServer="sPhoneType_25"/>
                </grid>
            </form>
<!--Forms - both server and client required, subforms have non-zero ids -->
            <form client="66" server="26">
                <controlAction client="30" server="50"/>
                < controlData longClient="txtEnterpriseOneEventPoint01_99"
                             longServer="txtEnterpriseOneEventPoint01_36"/>
                <grid client="20" server="20">
      <column    longClient="sPhoneType_88"
                            longServer="sPhoneType_25"/>
                </grid>
            </form>
        </mapping>
    </endPoint>
```

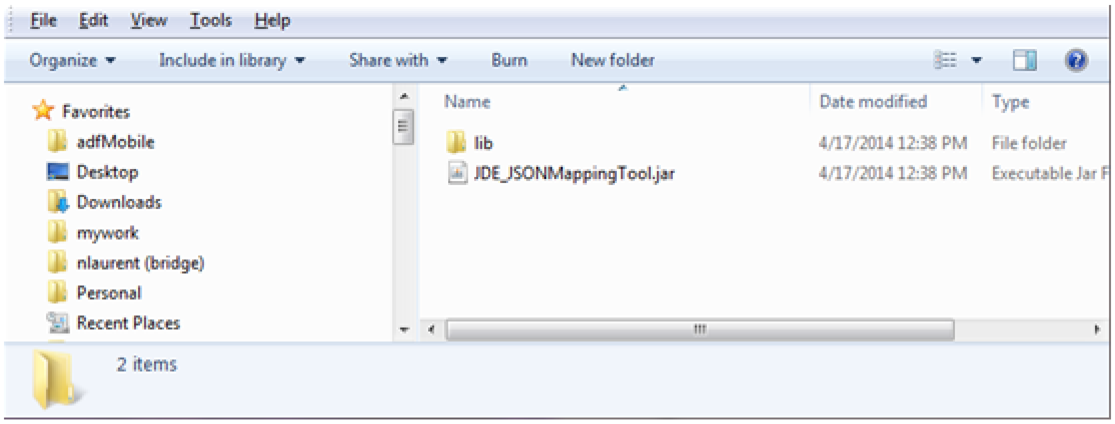
Using the JDE JSON Mapping Tool to Map Mismatched Fields

The JDE JSON Mapping Tool enables you to update the AISEndPoint.xml file with the proper JSON client-server mappings. After updating the AISEndPoint.xml file, you deploy it to the AIS Server to produce the expected JSON responses. The tool enables you to:

- Map mismatched fields to the current fields in the EnterpriseOne form.
- Save the mappings to a file so that later, you can incorporate the mappings into the AISEndPoint.xml file.

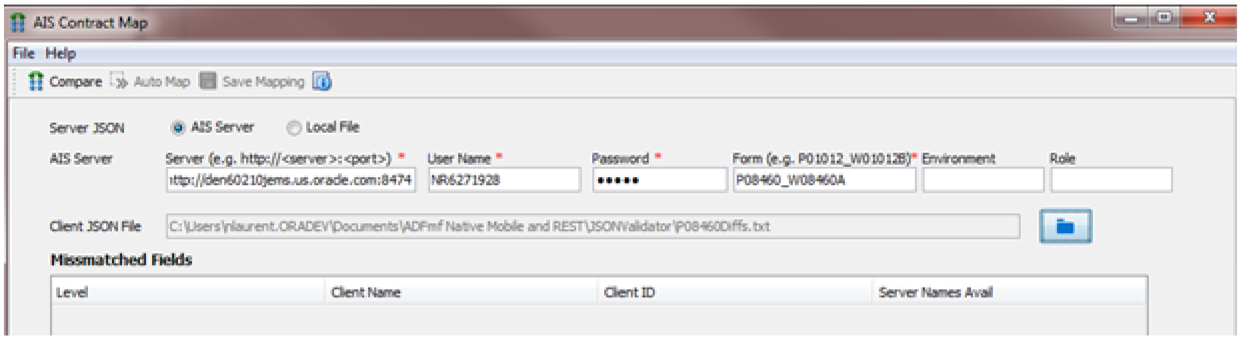
To map mismatched fields:

1. Locate the JDE_JSONMappingTool.zip file and unzip it to your local machine.



Name	Date modified	Type
lib	4/17/2014 12:38 PM	File folder
JDE_JSONMappingTool.jar	4/17/2014 12:38 PM	Executable Jar F

2. Double-click the jar file to launch the application.



Server JSON (● AIS Server ○ Local File)

AIS Server

Server (e.g. http://<server>:<port>) *	User Name *	Password *	Form (e.g. P01012_W01012B)*	Environment	Role
ttp://den60210jems.us.oracle.com:8474	NR6271928	•••••	P08460_W08460A		

Client JSON File: C:\Users\nlaurent.ORADEV\Documents\ADFmf Native Mobile and REST\JSONValidator\P08460Diffs.txt

Missmatched Fields

Level	Client Name	Client ID	Server Names Avail

3. Complete the following fields to identify the location of the server JSON information:

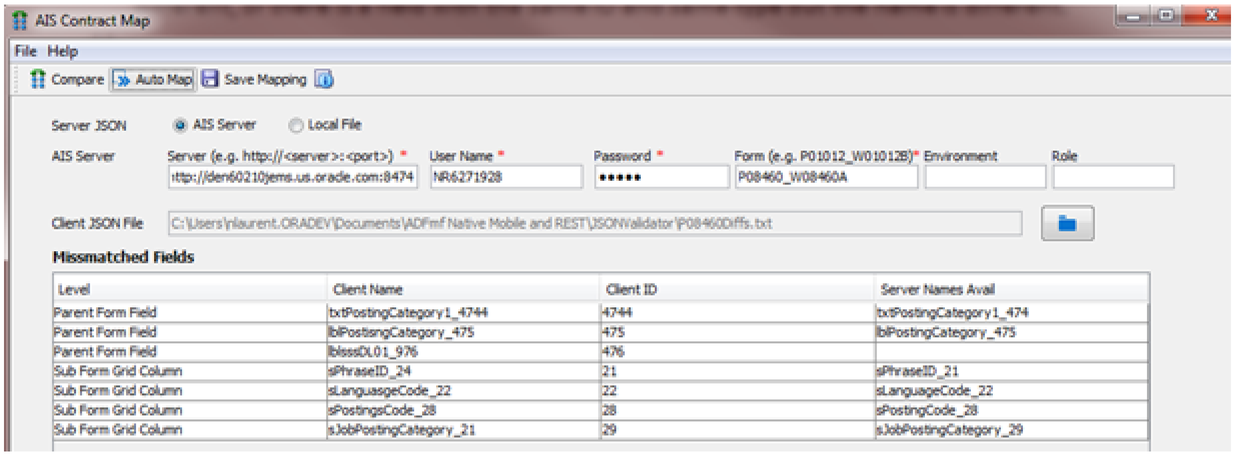
   - **Server**. Enter the path to the server where the AIS Server is deployed, for example: `http://<server>:<port>`
   - **User Name**. Enter a user name for the EnterpriseOne HTML Server.
   - **Password**. Enter a password for the EnterpriseOne HTML Server.
   - **Form**. Use the following syntax to identify the form for which you want to compare form service responses:

     `<application ID>_<form ID>`. For example: `P08460_W08460A`
   - **Environment**.
   - **Role**.

   Note: You can also select the Local File option if you have the file with server JSON information on your local machine.

4. In the Client JSON File field, click the folder button to select the locally stored file with the client contract, which contains the client's expected JSON response for that form.

5. Click the **Compare** button in the menu bar. The tool displays any mismatched fields in the grid.



Server JSON: [✓ AIS Server] Local File

AIS Server

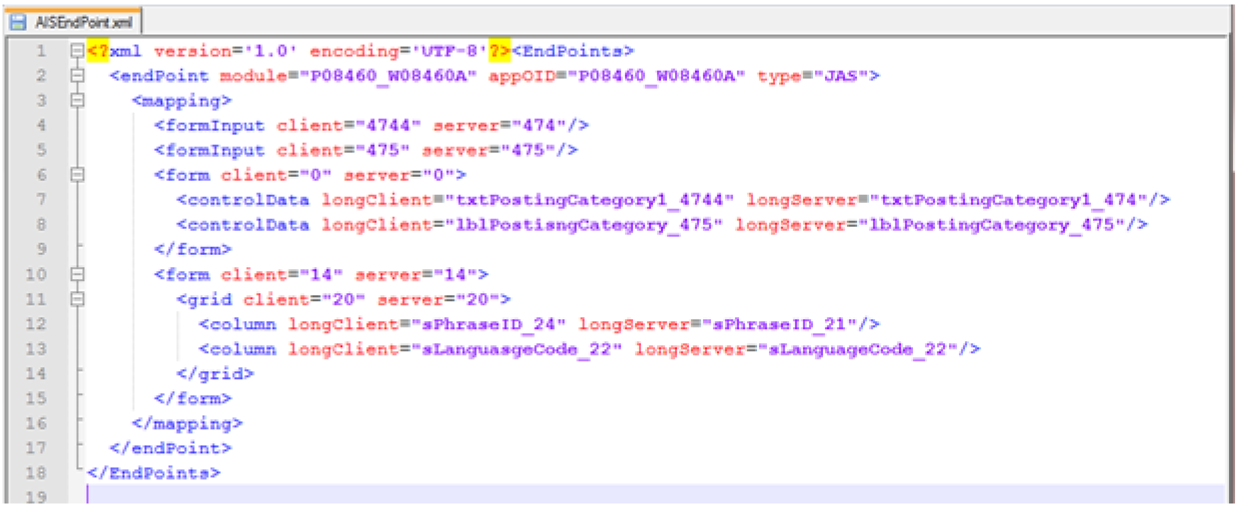
Server (e.g. http://<server>:<port>) *	User Name *	Password *	Form (e.g. P01012_W01012B)*	Environment	Role
ıttp://den60210jems.us.oracle.com:8474	NR6271928	●●●●●	P08460_W08460A		

Client JSON File: C:\Users\nlaurent.ORADEV\Documents\ADFmf Native Mobile and REST\JSONValidator\P08460Diffs.txt

Missmatched Fields

Level	Client Name	Client ID	Server Names Avail
Parent Form Field	txtPostingCategory1_4744	4744	txtPostingCategory1_474
Parent Form Field	lblPostisngCategory_475	475	lblPostingCategory_475
Parent Form Field	lblssDL01_976	476	
Sub Form Grid Column	sPhraseID_24	21	sPhraseID_21
Sub Form Grid Column	sLanguasgeCode_22	22	sLanguageCode_22
Sub Form Grid Column	sPostingsCode_28	28	sPostingCode_28
Sub Form Grid Column	sJobPostingCategory_21	29	sJobPostingCategory_29

6. Click the **Auto Map** button to map the fields automatically.
   Auto Map matches fields where only the ID is different, or it matches fields that have the same ID and type but different names.
7. Review the mappings and map any remaining mismatched fields manually by using the drop-down menu in the Server Names Avail column.
8. Click the **Save Mapping** button and enter a unique name to save the xml file as a new file. Later, you must manually copy the endpoint information from this file into AISEndPoint.xml file.
9. Click **OK** on the confirmation message.
10. Open the xml file to review and verify the output.



```xml
<?xml version='1.0' encoding='UTF-8'?><EndPoints>
  <endPoint module="P08460_W08460A" appOID="P08460_W08460A" type="JAS">
    <mapping>
      <formInput client="4744" server="474"/>
      <formInput client="475" server="475"/>
      <form client="0" server="0">
        <controlData longClient="txtPostingCategory1_4744" longServer="txtPostingCategory1_474"/>
        <controlData longClient="lblPostisngCategory_475" longServer="lblPostingCategory_475"/>
      </form>
      <form client="14" server="14">
        <grid client="20" server="20">
          <column longClient="sPhraseID_24" longServer="sPhraseID_21"/>
          <column longClient="sLanguasgeCode_22" longServer="sLanguageCode_22"/>
        </grid>
      </form>
    </mapping>
  </endPoint>
</EndPoints>
```

11. Follow the instructions in the *Modifying and Deploying AIS Field Mappings* section to deploy the updated mappings.

Modifying and Deploying AIS Field Mappings

Perform the following tasks to modify and deploy AIS field mappings:

- *Locate and Save the AIS Server Component JAR File as a New JAR*

- *Modify the AISEndpoint.xml in the AIS Server Component JAR File*
- *Use Server Manager to Deploy the New JAR File with the Updated AISEndpoint.xml to an AIS Server Instance*

Locate and Save the AIS Server Component JAR File as a New JAR

To do so:

1. Locate the JAR file, which should be named similar to this: `E1_AISServer_9.1.4.6_03-13-2014_12_48.jar`.

   This is the same JAR file used to deploy the AIS Server.
2. Copy the JAR file to your local machine, renaming the file to differentiate it from the original.

Modify the AISEndpoint.xml in the AIS Server Component JAR File

Update the AISEndpoint.xml file in the new JAR file with the endPoint mappings from the mapping file generated from the JDE JSON Mapping Tool. To do so:

1. Use a zip utility to open the archive.

   > **Note:** If you use 7-zip, you can open it (without unzipping it) and drill down to the folder that contains the JAR file, edit the JAR file, and then save it. If you use another zip utility, you may have to unzip the archive, edit the JAR file, and then rezip it.

2. In an editor, open the AISEndPoint.xml, which you can find in the following location:

   ```
   \E1_AISServer_Release_Name_ReleaseDate.jar\JDERestProxy.ear\JDERestProxy.war\WEB-INF\classes
   \Configuration\
   ```

   If you do not have the file path to the XML files defined in the Server Manager configuration settings for AIS, you can directly deploy the AISEndPoint.xml file as always, but in this location in the war file:

   ```
   \E1_AISServer_Release_Name_ReleaseDate.jar\JDERestProxy.ear\JDERestProxy.war\WEB-INF\classes
   \Configuration\
   ```

   If you defined the path to the XML files in Server Manager, you do not need to modify the deployment. Simply place the AISEndPoint.xml file directly in the folder defined in the configuration.

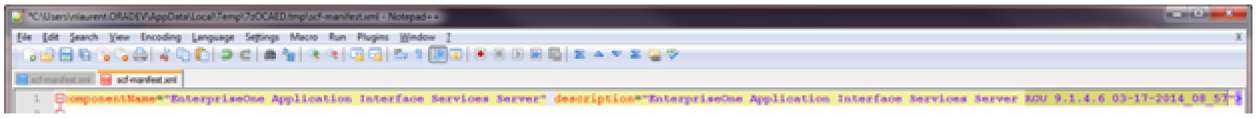
   If you defined the path to the XML files in Server Manager, and you choose to not place the AISEndPoint.xml file in that location, the system will still look for the AISEndPoint.xml file in the deployment JAR file.
3. In an editor, open the mapping file generated by the JDE JSON Mapping Tool.
4. In the mapping file, copy each `<endPoint>` section from the start tag to the end tag, and paste it into the AISEndPoint.xml as a child of the `<EndPoints>` element.
5. After you modify it, save and close it. If a message box appears asking if you want to update the modified file, click **OK**.

Use Server Manager to Deploy the New JAR File with the Updated AISEndpoint.xml to an AIS Server Instance

To do so:

1. In the JAR file, update the date and time.
2. Navigate to the top level of the JAR file. Click **OK** if any dialog boxes appear asking if you want to update files in the archive.
3. Right-click the scf-manifest.xml file and select **Open**.
4. At the top of the file, modify the description attribute to uniquely identify it.

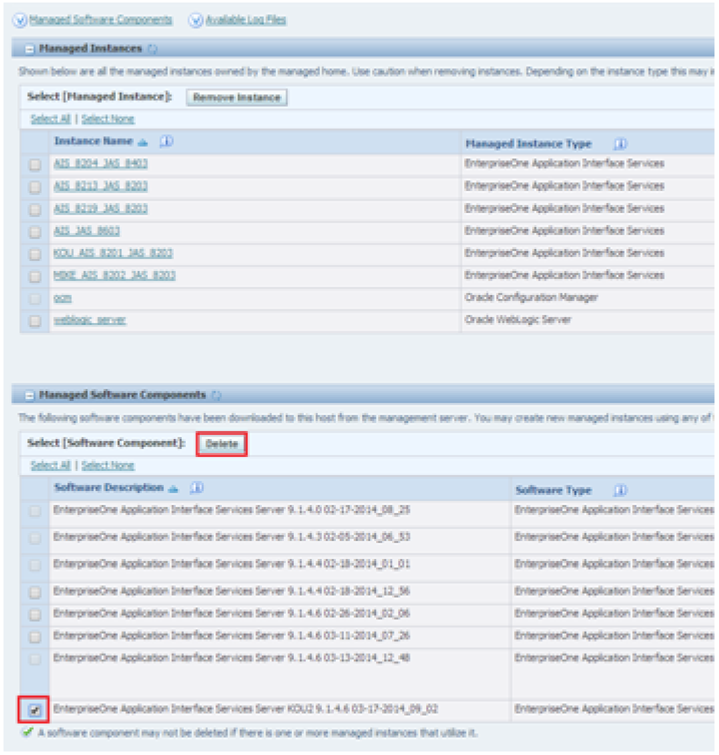
   The following screenshot shows an example of an scf-manifest.xml file with an updated description and time:



```
componentName="EnterpriseOne Application Interface Services Server" description="EnterpriseOne Application Interface Services Server KOU 9.1.4.6 03-17-2014_08_57"
```

5. Save and close the file. Click **OK** if a dialog box appears asking if you want to update the archive.
6. Close the archive or rezip the files into the archive, depending on the zip utility you are using.
7. Access Server Manager.
8. Find the AIS Server managed instance and click the **Change** button to change the software version to the following base version if it is not already set:

   EnterpriseOne Application Interface Services Server 9.1.4.6 03-13-2014_12_48



Managed Instances

Shown below are all the managed instances owned by the managed home. Use caution when removing instances. Depending on the instance type this may i

Select [Managed Instance]:

Instance Name	Managed Instance Type
AIS_8204_JAS_8403	EnterpriseOne Application Interface Services
AIS_8213_JAS_8203	EnterpriseOne Application Interface Services
AIS_8219_JAS_8203	EnterpriseOne Application Interface Services
AIS_JAS_8603	EnterpriseOne Application Interface Services
KOU_AIS_8201_JAS_8203	EnterpriseOne Application Interface Services
MIKE_AIS_8202_JAS_8203	EnterpriseOne Application Interface Services
ocm	Oracle Configuration Manager
weblogic_server	Oracle WebLogic Server

Managed Software Components

The following software components have been downloaded to this host from the management server. You may create new managed instances using any of

Select [Software Component]:

	Software Description	Software Type
	EnterpriseOne Application Interface Services Server 9.1.4.0 02-17-2014_08_25	EnterpriseOne Application Interface Services
	EnterpriseOne Application Interface Services Server 9.1.4.3 02-05-2014_06_53	EnterpriseOne Application Interface Services
	EnterpriseOne Application Interface Services Server 9.1.4.4 02-18-2014_01_01	EnterpriseOne Application Interface Services
	EnterpriseOne Application Interface Services Server 9.1.4.4 02-18-2014_12_56	EnterpriseOne Application Interface Services
	EnterpriseOne Application Interface Services Server 9.1.4.6 02-26-2014_02_06	EnterpriseOne Application Interface Services
	EnterpriseOne Application Interface Services Server 9.1.4.6 03-11-2014_07_26	EnterpriseOne Application Interface Services
	EnterpriseOne Application Interface Services Server 9.1.4.6 03-13-2014_12_48	EnterpriseOne Application Interface Services
✓	EnterpriseOne Application Interface Services Server KOU2 9.1.4.6 03-17-2014_09_02	EnterpriseOne Application Interface Services

A software component may not be deleted if there is one or more managed instances that utilize it.

9. Go to the managed home and delete the previous component that was assigned to your server.

10. In the left pane, select **Manage Software**, and then click the **Choose File** button.
11. Select the file you just modified, and then click the **Upload** button.
12. If you receive the following message, delete any existing uploads that were based on the same JAR. This will not affect any current deployments.

    ```
    "Caution: The uploaded file already exists in the management console. The uploaded file has been
    discarded."
    ```

    After uploading the file, you need to distribute it.
13. To distribute the file, select the managed home with the AIS Server instances. If the check box for the AIS Server instance is not selected, then you need to perform steps b and c.
14. Go to your managed instance and stop it.
15. For the Software Component Version, click the **Change** button, select the new one you just distributed, and then click the **Change Component** button.

    Server Manager automatically restarts the server.
16. Test the updated mappings by running the AIS client. If the application functions properly, you have successfully updated the mappings.

9 AIS Server Capabilities and Services

AIS Server Capabilities

The AIS Server is agnostic of any clients that may call its services. The AIS Server exposes various capabilities that AIS clients may or may not depend on. Clients may have a dependency on specific capabilities of the AIS Server, so the AIS Server exposes the list of capabilities, which enables clients to ensure that the AIS Server they are using will be able to properly respond to their service requests.

The defaultconfig service on the AIS Server enables you to view a list of the capabilities available on the AIS Server. This service also provides various configuration information about the AIS Server instance.

Depending on your EnterpriseOne Tools release, there can be two sets of AIS Server capabilities that you can access:

- Base capabilities that are available with the original "version 1" AIS services. To view the base capabilities available with your EnterpriseOne Tools release, use the following URL to the defaultconfig service:

  `https://<AIS_Server>:<Port>/jderest/defaultconfig`

- Version 2 capabilities available with version 2 AIS services (EnterpriseOne Tools 9.2.1.2). Use the following URL to the defaultconfig service to view a list of version 2 capabilities:

  `https://<AIS_Server>:<Port>/jderest/v2/defaultconfig`

You can enter the URL in a browser to access the list of AIS Server capabilities. Or in an AIS client application, you can perform a GET operation on the defaultconfig URL.

When you access the list of AIS Server capabilities, the capabilities that are displayed are based on the EnterpriseOne Tools release of the AIS Server. See *Base AIS Server Capabilities* and *Version 2 AIS Server Capabilities* for a list of capabilities.

Base AIS Server Capabilities

The following list shows all base AIS Server capabilities and the EnterpriseOne Tools release they were released with:

```
"capabilityList": [
    {
        "name": "grid",
        "shortDescription": "Grid Actions",
        "longDescription": "Ability to update, insert and delete grid
         records.",
        "asOfRelease": "9.1.4.4"
    },
    {
        "name": "editable",
        "shortDescription": "Enabled/Disabled",
        "longDescription": "Ability to indicate if form field or grid cell is
         editable (enabled) or not (disabled).",
        "asOfRelease": "9.1.4.4"
    },
    {
        "name": "log",
        "shortDescription": "Logging",
        "longDescription": "Endpoint exposed for logging to AIS server log
         from client",
        "asOfRelease": "9.1.4.6"
    },
    {
        "name": "processingOption",
        "shortDescription": "Processing Options",
        "longDescription": "Processing Option Service exposed for fetching PO
         values from E1",
        "asOfRelease": "9.1.4.6"
    },
    {
        "name": "ignoreFDAFindOnEntry",
        "shortDescription": "Ignore FDA Find On Entry",
        "longDescription": "Ability to use the IgnoreFDAFindOnEntry flag",
        "asOfRelease": "9.1.4.6"
    }
    {
        "name": "selectAllGridRows",
        "shortDescription": "Select or Unselect All Grid Rows",
        "longDescription": "Ability to use select and unselect all grid rows,
         or unselect a single row in an action event.",
        "asOfRelease": "9.1.5"
    },
    {
        "name": "applicationStack",
        "shortDescription": "Operations on a Stack of E1 Applications",
        "longDescription": "Ability to maintain a stack of open E1
         applications and operate forms that are called",
        "asOfRelease": "9.1.5"        },
    {
        "name": "thumbnailSize",
        "shortDescription": "Specify desired thumbnail size for MO List",
        "longDescription": "Ability to request a specific sized thumbnail
         images in a Media Object List Request",
        "asOfRelease": "9.1.5"
    },
    {
        "name": "gridCellClick",
        "shortDescription": "Click Grid Cell Hyperlink",
        "longDescription": "Ability to use GridCellClick event, to execute
         hyperlink in grid.",
        "asOfRelease": "9.1.5.2"
    },
    {
        "name": "query",
        "shortDescription": "Query",
        "longDescription": "Ability to use Query on forms that support it",
        "asOfRelease": "9.1.5.2"
    },
    {
        "name" : "taskAuthorization",
        "shortDescription" : "Task Authorization",
        "longDescription" : "Ability to receive a list of authorized tasks
         based on a task view id, or task id and parent id with in a task
         view",
        "asOfRelease" : "9.1.5.2"
    },
    {
        "name": "urlMediaObjects",
            "shortDescription": "URL Media Objects",
            "longDescription": "Ability to view, add or delete url type media
             objects",
            "asOfRelease": "9.1.5.2"
        }
        {
            "name" : "jargon",
            "shortDescription" : "Data Item Jargon Service",
            "longDescription" : "Ability to request data item descriptions based
             on users language and jargon (system) code",
            "asOfRelease" : "9.1.5.3"
        },
        {
            "name" : "aliasNaming",
            "shortDescription" : "Alias Naming",
            "longDescription" : "Ability receive form service responses with fields named
 by Data Dictionary alias",
            "asOfRelease" : "9.1.5.3"
        },
        {
            "name" : "orchestrator",
            "shortDescription" : "Orchestrator",
            "longDescription" : "Ability process multiple service requests, rules,
             cross references in a single call based on defined orchestration
             metadata",
            "asOfRelease" : "9.1.5.5"
        },
        {
            "name" : "basicAuth",
            "shortDescription" : "Basic Authorization",
            "longDescription" : "Ability receive basic authorization credentials
             for the token request service",
            "asOfRelease" : "9.1.5.5"
        },
        {
            "name" : "dataservice",
            "shortDescription" : "Data Service",
            "longDescription" : "Ability to execute queries directly against
             tables and views",
           "asOfRelease" : "9.1.5.5"
        }
        ],
        {
            "name" : "actionControls",
            "shortDescription" : "Show Action Controls",
            "longDescription" : "Ability to request that action controls are
             returned in the form service response",
            "asOfRelease" : "9.2.0.2"
        },
        {
            "name" : "aggColumnClick",
            "shortDescription" : "Click Grid Column Aggregate",
            "longDescription" : "Ability to send an event to click the aggregation
             icon on a grid column",
            "asOfRelease" : "9.2.0.2"
        },
        {
              "name" : "dataServiceAggregation",
              "shortDescription" : "Data Service Aggregation",
              "longDescription" : "Ability to request aggregation of the data in a
             data service call",
            "asOfRelease" : "9.2.0.2"
          },
          {              "name" : "outputType",             "shortDescription" : "Output
Type",            "longDescription" : "Ability to request different output format for
             form service or data service responses",            "asOfRelease" :
"9.2.0.2"         },
        {
            "name" : "preferenceService",
            "shortDescription" : "Preference Service",
            "longDescription" : "Ability to save and retrieve preference values by
             user",
            "asOfRelease" : "9.2.0.2"
        },
        {
            "name" : "orchestrationDiscovery",
            "shortDescription" : "Orchestration Discovery",
            "longDescription" : "Ability to discover available orchestrations",
            "asOfRelease" : "9.2.0.2"
        },
        {
            "name" : "queryObjectName",
            "shortDescription" : "Query Object Name",
            "longDescription" : "Ability to request an exsiting query to be used
             for a FormService or DataService request",
            "asOfRelease" : "9.2.0.2"
        }
         {
            "name" : "watchlist",
            "shortDescription" : "Watchlist",
            "longDescription" : "Ability to request an exsiting watchlist value",
            "asOfRelease" : "9.2.0.3"
        },
        {
            "name" : "aggregationCurrencyDecimals",
            "shortDescription" : "Aggregation Currency Decimals",
            "longDescription" : "Ability to request aggregation of currency
             columns and apply currency processing for deciamls on aggregated
             data",
            "asOfRelease" : "9.2.0.3"
        },
        {
            "name" : "dataServiceOrderBy",
            "shortDescription" : "Data Service Browse Order By",
            "longDescription" : "Ability to request columns to order by for browse
      type data service requests",
      "asOfRelease" : "9.2.0.5"
  },
  {
      "name" : "availableQueries",
      "shortDescription" : "List Available Saved Queries",
      "longDescription" : "Ability to request the list of available saved queries for an
form, table or view",
      "asOfRelease" : "9.2.1.0"
  },
  {
      "name" : "complexQuery",
      "shortDescription" : "Complex Query",
      "longDescription" : "Ability to specify a complex ad hoc query, several combined
queries",
      "asOfRelease" : "9.2.1.0"
  },
  {
      "name" : "dateGroupSpecialHandling",
      "shortDescription" : "Date Group By Special Handling",
      "longDescription" : "Ability to specify the date format returned for aggregations
grouped by date columns",
      "asOfRelease" : "9.2.1.0"
  },
  {
      "name" : "asIfCurrency",
      "shortDescription" : "As If Currency",
      "longDescription" : "Ability to specify a single currency for all values returned
in an aggregation with currency decimal processing",
      "asOfRelease" : "9.2.1.0"
  },
  {
      "name" : "queryDetails",
      "shortDescription" : "User Defined Query Details",
      "longDescription" : "Ability to get the details of an individual user defined
query",
      "asOfRelease" : "9.2.1.0"
  },
  {
      "name" : "applicationQueryInDataService",
      "shortDescription" : "Application Query in Data Service",
      "longDescription" : "Ability to request an existing query defined for a form, to be
used in a data service request over the same view as the form",
      "asOfRelease" : "9.2.1.0"
  },
  {
      "name" : "queryCombining",
      "shortDescription" : "Combine Saved and Ad Hoc Query",
      "longDescription" : "Ability to request both a saved query and an ad hoc query be
combined with an AND to filter data for a form service or data service request",
      "asOfRelease" : "9.2.1.0"
  },
  {
      "name" : "queryAggegration",
      "shortDescription" : "Use Saved Query in Aggregation",
      "longDescription" : "Ability to request filtering of data with a saved query for an
aggregation data request",
      "asOfRelease" : "9.2.1.0"
  },
  {
      "name" : "turboMode",
      "shortDescription" : "Turbo Mode",
      "longDescription" : "Ability to request optimized grid processing in form service
request, where only the requested columns are processed",
      "asOfRelease" : "9.2.1.0"
  },
  {
      "name" : "timings",
      "shortDescription" : "Request Timing Information",
      "longDescription" : "Ability to request timing information for form service and
data service processing times",
      "asOfRelease" : "9.2.1.0"
  },
  {
      "name" : "availableWatchlists",
      "shortDescription" : "List Available Watchlists",
      "longDescription" : "Ability to request the list of available watchlists for a
user",
      "asOfRelease" : "9.2.1.0"
  }
  {
   "name" : "version2services",
   "shortDescription" : "Version 2 Service URL",
   "longDescription" : "New capabilities available with V2 Services",
   "asOfRelease" : "9.2.1.2"
  },
  {
   "name" : "nextPage",
   "shortDescription" : "Next Page for App Stack and Data Request",
   "longDescription" : "Ability to request next data set for Application Stack and Data
Request",
   "asOfRelease" : "9.2.1.2"
  },
  {
   "name" : "simpleGetServices",
   "shortDescription" : "Simple GET Data Requests",
   "longDescription" : "Ability to perform Data Requests with GET operation",
   "asOfRelease" : "9.2.1.2"
  },
  {
   "name" : "formServiceOrderBy",
   "shortDescription" : "Order By in Form Service",
   "longDescription" : "Ability to order by columns in a grid with form service",
   "asOfRelease" : "9.2.1.2"
  },
  {
   "name" : "messageService",
   "shortDescription" : "Send Message Service",
   "longDescription" : "Ability to send E-mail or Work Center messages",
   "asOfRelease" : "9.2.1.2"
  } ],
```

Version 2 AIS Server Capabilities

The following list shows all version 2 AIS Server capabilities and the EnterpriseOne Tools release they were released with:

```
"capabilityList": [{
           "name": "version2services",
           "shortDescription": "Version 2 Service URL",
           "longDescription": "New capabilities available with V2 Services",
           "asOfRelease": "9.2.1.2",
           "sinceVersion": "v2"
     },
     {
           "name": "nextPage",
           "shortDescription": "Next Page for App Stack and Data Request",
           "longDescription": "Ability to request next data set for Application Stack and
Data Request",
           "asOfRelease": "9.2.1.2",
           "sinceVersion": "v2"
     },
     {
           "name": "simpleGetServices",
           "shortDescription": "Simple GET Data Requests",
           "longDescription": "Ability to perform Data Requests with GET operation",
           "asOfRelease": "9.2.1.2",
           "sinceVersion": "v2"
     },
     {
           "name": "formServiceOrderBy",
           "shortDescription": "Order By in Form Service",
           "longDescription": "Ability to order by columns in a grid with form service",
           "asOfRelease": "9.2.1.2",
           "sinceVersion": "v2"
     },
     {
           "name": "messageService",
           "shortDescription": "Send Message Service",
           "longDescription": "Ability to send E-mail or Work Center messages",
           "asOfRelease": "9.2.1.2",
           "sinceVersion": "v2"
     }
]
```

Capabilities Not Supported in EnterpriseOne Mobile Enterprise Applications

The following capabilities are NOT supported in the JDE Mobile Application Framework API for mobile enterprise applications: all of the 9.2.1.0 capabilities in the preceding list and queryObjectName, orchestrationDiscovery, outputType, aggColumnClick, actionControls, oAuth, basicAuth, orchestrator, aliasNaming, jargon, and taskAuthorization.

Version 1 and Version 2 AIS Services (Endpoints)

The AIS Server exposes endpoints that:

- Enable access to EnterpriseOne data and applications.
- Produce JSON responses.

Each endpoint provides a particular service, referred to as an AIS service, that AIS clients can use to interact with EnterpriseOne applications.

#unique_76/unique_76_Connect_42_CHDBGCHI describes version 1 AIS services available on the AIS Server. These services are accessed using the following URL path:

`https://<AIS_Server>:<Port>/jderest/<endpoint>`

With EnterpriseOne Tools release 9.2.1.2 comes the availability of version 2 AIS services. All version 1 AIS services are included in version 2 AIS services. *#unique_76/unique_76_Connect_42_CHDIBIAD* describes additional AIS services available in version 2. All paths to version 2 AIS services include **`/v2`** in the URL path, for example:

`https://<AIS_Server>:<Port>/jderest/v2/<endpoint>`

For a detailed description of the methods, operations, and parameters in each AIS service, see the *JD Edwards EnterpriseOne Tools REST API for the Application Interface Services Server Guide* reference documentation.

Endpoint URIs for Version 1 AIS Services

Endpoint URI	HTTP Method	Description of Service
/defaultconfig	GET	The response will include information about the AIS Server including the release level, EnterpriseOne HTML Server configuration, and capabilities list. The capabilities that are available on the AIS Server depend on the EnterpriseOne Tools release applied to the AIS Server. If you are using the latest AIS Client Java API with up-to-date capabilities to develop AIS client applications, you need to make sure the AIS Server is on the latest EnterpriseOne Tools release. See *AIS Server Capabilities* for a list of capabilities available by EnterpriseOne Tools release.
/tokenrequest	POST	Based on the input, the response will contain login information including a login token and user details.
/tokenrequest/logout	POST	Based on the input (AIS token), the response will be a code of 200 if successful and 500 if the logout fails.
/formservice	POST	Based on the input, the response will contain a JSON representation of the form requested.
/batchformservice	POST	Based on the input, the response will contain a JSON representation of all of the forms requested.
/file/gettext	POST	Based on the input, the response will contain the text for the first text media object.
/file/updatetext	POST	Base on the input, the response will contain the status of the text update.
/file/list	POST	Based on the input, the response will contain the list of media objects for the structure and key requested.
/file/upload	POST (Multi-Part Form)	The response will contain the details of the uploaded file, including the media object sequence number.
/file/download	POST	This response will contain a multi-part form including the data for the attachment.
/file/addurl	POST	The response will contain the details of the URL media object, including the URL text and the sequence number.
/file/delete	POST	The response indicates the success or failure to delete the media object for the sequence number passed in.
/appstack	POST	Based on the input, the response will contain the current form open on the stack and any stack related information.

Endpoint URI	HTTP Method	Description of Service
/poservice	POST	Based on the input, the response will contain the processing option values for the requested application and version.
/log	POST	Base on the input, the AIS Server will write a log entry with the information passed to the log service.
/jargonservice*	POST	Based on the input and the logged in users language, the correct item description will be returned for each data item provided.
/dataservice	POST	Based on the input, the response will contain either a count or a list of records matching a query of a table or view.
/orchestrator/<Name>*	POST	Based on the input and the URI, the requested orchestration will run. See *"Configuring an Orchestration" in the JD Edwards EnterpriseOne Tools Orchestrator Guide* for more information about creating orchestrations that use service requests to invoke EnterpriseOne applications.
/preference (Release 9.2.0.2)	POST	Allows management of preference records in the User Overrides Table (F98950).
/discover (Release 9.2.1)*	POST	Request a list of available orchestrations for that user.
/admin (Release 9.2.1)*	POST	Perform administrative operations such as clearing caches. The user must be configured in the Admin List in Server Manager to use this service.
/udomanager/<operation> (Release 9.2.1)*	POST	Request details about UDO objects. Valid operations are WATCHLIST and QUERY.
/watchlist (Release 9.2.1)*	POST	Execute an EnterpriseOne watchlist.

*These endpoints are **NOT** supported in the JDE Mobile Framework API for developing EnterpriseOne mobile enterprise applications.

Endpoint URIs for Version 2 AIS Services

Endpoint URI	HTTP Method(s)	Description of Service
/appstack/next	GET POST	Executes stateful calls to get the next set of records in a grid.
/dataservice/next	GET POST	Executes stateful calls to get the next set of records in a grid.

Endpoint URI	HTTP Method(s)	Description of Service
/dataservice/table/<tableName>	GET	Executes simple queries over EnterpriseOne tables.
/dataservice/view/<viewName>	GET	Executes simple queries over EnterpriseOne business views.
/formservice/<application>/<form>	GET	Executes simple queries in EnterpriseOne forms.
/formservice/<application>/<form>/<version>	GET	Executes a simple query in a form in a particular EnterpriseOne application version.
/message	POST	Sends an Enterpriseone message (PPAT) to external email systems or the EnterpriseOne Work Center email system

10 Caching Service Request Responses (Release 9.2.1.1)

Overview

You can enable the AIS Server to cache responses of service requests that perform Read operations, which include:

- Data service requests

  All data service requests perform a Read operation, therefore all responses from data services requests can be cached.
- Form service requests configured to perform a Read operation, for example `"formServiceActions":"R"`

  If you are using a batch form service request, you need to make sure that each request in the batch request is configured with the preceding parameter.

The information in each service request provides the key to the cache. Only requests with the exact same information in the input parameters as previous requests receive responses from the cache.

When the AIS Server cache is used for the service request, the response to the service request will contain the following parameter:

```
"fromCache": true
```

Enabling Service Request Response Caching

In Server Manager, configure the following settings in the AIS Server configuration group settings to enable AIS Server caching:

- **Read Cache Time To Live (Milliseconds)**. This setting determines the amount of time cache can be used for service request responses before subsequent requests are fetched from the database. To enable the cache, this number must be greater than zero. The default value is 60000 milliseconds.
- **Enable Caching by Default**. Select this check box to enable caching of responses for service requests not coded for caching. If not enabled, you must use caching parameters described in *Service Request Caching Parameters* to enable caching of responses for individual service requests.

For details of each configuration setting, including valid values, refer to the Server Manager internal help for each setting.

Service Request Caching Parameters

The following table describes the parameters that you can use to configure the caching behavior of responses from form service and data services requests.

Parameter	Value	Description
allowCache	Boolean (true, false)	Enable or disable the caching of the service request response. If the "Enable Caching by Default" option is enabled in Server Manager, this parameter is ignored and all responses for Read operations will be cached.
forceUpdate	Boolean (true, false)	Force or do not force the system to fetch the data from the database for the service request. If the "Enable Caching by Default" setting is enabled in Server Manager, you can use this value to override the setting.
cacheTime	Integer (milliseconds)	Enter the amount of time in milliseconds for the service request to use the cache for the response. This setting overrides the value in the "Default Read Cache Time To Live" setting in Server Manager.
setDirtyOnly	Boolean (true,false)	If set to true, this parameter forces the next service request call (that does not contain this flag) to fetch data from the database. If you include this in the request, you will not get any data back. You will only receive a status message.

11 Appendix A - AIS Troubleshooting

Troubleshooting AIS Server Issues

AIS Client Fails to Connect to the Server

If the AIS client application fails to connect to the server, verify that the IP Address of the AIS Server has been entered correctly in the Allowed Hosts field. If the IP Address is correct and the connection still fails, then enter an * (asterisk) in the Allowed Hosts setting, which enables the EnterpriseOne HTML Server to accept requests from any host.