

JD Edwards EnterpriseOne

Application Interface Services Client Java API Developer's
Guide

Release 9.1.5

E62368-05

December 2015

Describes how to work with the Application Interface Services (AIS) Client Java API which provides classes and methods for creating custom applications that work with EnterpriseOne.

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

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

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

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

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

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

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

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

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

Contents

Preface	v
Audience	v
Documentation Accessibility	v
Related Documents	v
Conventions	v
 1 Understanding the AIS Client Java API	
1.1 Overview	1-1
1.2 Accessing AIS Server Endpoints with the AIS Client Java API	1-1
 2 Getting Started	
2.1 Certifications (Formerly Known as Minimum Technical Requirements)	2-1
2.2 Prerequisites	2-1
2.3 Installing the AIS Client Class Generator Extension for JDeveloper	2-2
 3 Configuring the Login Environment	
3.1 Configuring the Login	3-1
3.2 Configuring the Logout	3-2
 4 Using the AIS Client Class Generator	
4.1 Understanding Generating Objects with the AIS Client Class Generator	4-1
4.2 Configuring the AIS Client Class Generator Preferences	4-1
4.3 Generating Data Classes Based on a Form	4-2
4.4 Generating Data Classes Based on a Data Request (Available in AIS Client Class Generator v1.6.2)	4-5
 5 Performing AIS Form Service Calls	
5.1 Understanding AIS Server Capabilities	5-1
5.2 Understanding Form Service Requests	5-4
5.2.1 Overview	5-4
5.2.2 Form Service Request Structure	5-5
5.2.3 Control ID Notation for Return Control IDs	5-5
5.2.4 Reading Data	5-6
5.2.5 Adding Data	5-7

5.2.6	Deleting Data.....	5-8
5.2.7	Placing Events in the Proper Order	5-10
5.2.8	Considering Hidden Filters and Hidden QBE	5-10
5.2.9	Available Actions or Events.....	5-10
5.3	Batch Form Service	5-11
5.4	Application Stack Service (Tools Release 9.1.5).....	5-15
5.5	Media Object Operations	5-17
5.5.1	Get Text	5-17
5.5.2	Update Text	5-18
5.5.3	List.....	5-18
5.5.4	Upload.....	5-21
5.5.5	Download.....	5-22
5.5.6	Add URL (Tools Release 9.1.5.2 and API Version 1.0)	5-23
5.5.7	Delete	5-23
5.6	Processing Option Service	5-24
5.7	Task Authorization Service	5-25
5.8	Logging Service	5-26
5.9	Query (Release 9.1.5.2)	5-27
5.9.1	Query Object Parameters.....	5-28
5.10	Jargon Service (Release 9.1.5.3)	5-30
5.11	Data Service (API Release 1.1.0 and EnterpriseOne Tools Release 9.1.5.5)	5-31
5.12	Orchestration Support (API Release 1.1.0 and EnterpriseOne Tools Release 9.1.5.5)....	5-33

Glossary

Preface

Welcome to the *JD Edwards EnterpriseOne Application Interface Services Client Java API Developer's Guide*. This guide has been updated for JD Edwards EnterpriseOne Tools 9.1 Update 5.2, 9.1 Update 5.3, and 9.1 Update 5.5.

Audience

This guide is intended for application developers who are responsible for creating client applications that use the Application Interface Services (AIS) Server to interact with JD Edwards EnterpriseOne web client applications.

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 Documents

See the following guide for related information:

- *JD Edwards EnterpriseOne Application Interface Services (AIS) Client API Reference*, available alongside this guide in the JD Edwards EnterpriseOne Tools Documentation Library:
http://docs.oracle.com/cd/E24705_01/nav/development.htm
- *JD Edwards EnterpriseOne Tools Internet of Things Orchestrator Guide*.

Conventions

The following text conventions are used in this document:

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

Understanding the AIS Client Java API

This chapter contains the following topics:

- [Section 1.1, "Overview"](#)
- [Section 1.2, "Accessing AIS Server Endpoints with the AIS Client Java API"](#)

1.1 Overview

With the Application Interface Services (AIS) Client Java API, you can use any development tool that works with Java APIs to create custom applications that interact with EnterpriseOne. Whether you need a simplified kiosk application for your warehouse, an application that composites features from multiple EnterpriseOne applications into a single purpose-built interface, or an application for the latest wearable device, the AIS Client Java API enables you to choose the development platform that fits your needs.

The AIS Client Java API enables developers to create applications, referred to as AIS clients, that communicate with the JD Edwards EnterpriseOne AIS Server. The AIS Server is a REST services server that when configured with the EnterpriseOne HTML Server, enables access to EnterpriseOne forms and data. The AIS Client Java API provides classes and methods that enable AIS clients to manage (create, read, update, delete) data in EnterpriseOne through REST services.

Note: The EnterpriseOne HTML Server also executes some Java processing; therefore, it is sometimes referred to as the Java Application Server (JAS). The terms HTML Server and JAS Server are synonymous.

See Also:

- "EnterpriseOne Application Interface Services (AIS) Server" in the *JD Edwards EnterpriseOne Tools System Overview Guide* for an overview and illustration of the AIS Server architecture.

1.2 Accessing AIS Server Endpoints with the AIS Client Java API

The AIS Server exposes endpoints that:

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

Each endpoint provides a particular service that AIS clients can use to interact with EnterpriseOne applications. [Table 1–1](#) describes the services that the AIS Server endpoints provide.

The AIS Client Java API enables easy access to all endpoints; all of the communication is handled for you. You can access AIS Server endpoints using this URL format:
`http://<server>:<port>/jderest/<URI>`

All POST calls expect JSON formatted request payloads.

When you use the API, you work with Java objects, not the JSON strings. But it is still important to understand how the data is transmitted. The following chapters in this guide describe in detail how to use the services the endpoints provide and the Java objects required to use them:

- [Chapter 3, "Configuring the Login Environment"](#)
- [Chapter 5, "Performing AIS Form Service Calls"](#)

Table 1–1 Endpoint URIs for Accessing EnterpriseOne Applications and Data

Endpoint URI	HTTP Method	Description of Service
/defaultconfig	GET	The response will include information about the AIS Server including the release level, JAS Server configuration, and capabilities list. The AIS Server has different capabilities based on the version of the EnterpriseOne Tools release applied to the AIS Server. Therefore, even if the version of the AIS Client Java API has the latest, up-to-date capabilities, the AIS Server may not. See Understanding 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 (Release 9.1.5.2).
/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.

Table 1–1 (Cont.) Endpoint URIs for Accessing EnterpriseOne Applications and Data

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 the JD Edwards EnterpriseOne Tools IoT Orchestrator Guide for more information about creating orchestrations that use form service requests to invoke EnterpriseOne applications.

Getting Started

This chapter contains the following topics:

- Section 2.1, "Certifications (Formerly Known as Minimum Technical Requirements)"
- Section 2.2, "Prerequisites"
- Section 2.3, "Installing the AIS Client Class Generator Extension for JDeveloper"

2.1 Certifications (Formerly Known as Minimum Technical Requirements)

Customers must conform to the supported platforms for the release, which can be found in the Certifications tab on My Oracle Support: <https://support.oracle.com>.

For more information about JD Edwards EnterpriseOne Minimum Technical Requirements, see the following document on My Oracle Support: JD Edwards EnterpriseOne Minimum Technical Requirements Reference (Doc ID 745831.1), which is available here:

<https://support.oracle.com/epmos/faces/DocumentDisplay?id=745831.1>

2.2 Prerequisites

To develop AIS client applications, you must complete the following prerequisites:

- You must be running a minimum of JD Edwards EnterpriseOne Tools release 9.1.5.
- Deploy the Application Interface Service (AIS) Server configured with an EnterpriseOne HTML Server. See "Create an Application Interface Services (AIS) Server as a New Managed Instance" in the *JD Edwards EnterpriseOne Tools Server Manager Guide*.
- Download the latest AIS_Client_Java_API_1.x.x from the JD Edwards Update Center on My Oracle Support (<https://support.oracle.com/>).

To locate the download on the JD Edwards Update Center, use the Type field to search on "EnterpriseOne ADF."

The zip file contains:

- AIS_Client.jar, which contains the AIS Client Java API.

Click the following link to access the *JD Edwards EnterpriseOne Application Interface Services (AIS) Client API Reference* Javadoc, which provides descriptions of the AIS Client Java API classes and methods:

http://docs.oracle.com/cd/E24705_01/nav/development.htm

- Jackson 2.2.4 library, which includes the jackson-databind, jackson-core, and jackson-annotations jar files.
- AISCGE 12c_v1.6.x.zip (AIS Client Class Generator extension for JDeveloper).

The AIS Client Class Generator is compatible only with JDeveloper 12.1.3 and up. After you download it, see [Installing the AIS Client Class Generator Extension for JDeveloper](#).

Important: The AIS client and Jackson jar files must be in the classpath of your AIS client.

2.3 Installing the AIS Client Class Generator Extension for JDeveloper

The AIS Client Class Generator extension for JDeveloper contains the AIS Client Class Generator, a tool that supports the creation of Application Controller foundational classes that are required by EnterpriseOne mobile applications.

For more information about the AIS Client Class Generator, see [Understanding Generating Objects with the AIS Client Class Generator](#) in this guide.

To install the AIS Client Class Generator 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 zip file.
4. Click **Next**, and then click **Finish**.

JDeveloper closes automatically.

Configuring the Login Environment

This chapter contains the following topic:

- [Section 3.1, "Configuring the Login"](#)
- [Section 3.2, "Configuring the Logout"](#)

3.1 Configuring the Login

For an AIS client to call AIS services, the AIS client must first obtain a login environment by passing the following information to the constructor in the `LoginEnvironment` object:

- **EnterpriseOne login credentials.** EnterpriseOne credentials include a user ID, password, environment, and role. The AIS Server configuration uses a default EnterpriseOne environment and role unless you specify a different environment and role here.
- **AIS Server URL and the device name.** The device name is a string that represents the device on which the client is running. The device name serves as a unique identifier for your client.
- **A list of required capabilities.** (Optional) If the AIS client uses AIS Server capabilities, then you have the option to pass a list of required capabilities to the `LoginEnvironment` constructor. The `LoginEnvironment` constructor verifies that the capabilities are available on the AIS Server. If they are available, access to the AIS client is granted. If they are not available, access is denied.

This prevents an AIS client from running if the AIS Server capability that it requires to properly function is not available in the version of the AIS Server. See [Understanding AIS Server Capabilities](#) for a list of AIS Server capabilities available by EnterpriseOne Tools release.

When the client requests a `LoginEnvironment`, the processing within the API uses the `defaultcfg` and `tokenrequest` URI, the endpoints described in [Table 1–1, "Endpoint URIs for Accessing EnterpriseOne Applications and Data"](#).

Example 3–1 Examples for Obtaining a Login Environment

```
//login with minimum required information
final String AIS_SERVER = "http://ais.company.com:7777";
final String USER_NAME = "jde";
final String PASSWORD = "jde";
final String DEVICE = "Java";
LoginEnvironment loginEnv = new LoginEnvironment(AIS_SERVER, USER_NAME, PASSWORD,
DEVICE);
```

```
//login overrides default environment and role
final String ENVIRONMENT = "PROD";
final String ROLE = "PROLE";
LoginEnvironment loginEnv2 = new LoginEnvironment(AIS_SERVER, USER_NAME,
PASSWORD, ENVIRONMENT, ROLE, DEVICE);

//login with required capabilities
//A CapabilityException will be thrown if AIS doesn't have those in the list
final String REQ_CAPABILITIES = "grid, processingOption";
LoginEnvironment loginEnv3 = new LoginEnvironment(AIS_SERVER, USER_NAME,
PASSWORD, DEVICE, REQ_CAPABILITIES);

//login with token
String PS_TOKEN = "a ps token string";
LoginEnvironment loginEnv4 = new LoginEnvironment(AIS_SERVER, USER_NAME, null,
null, null, DEVICE, null, null, PS_TOKEN)
```

All calls to the AIS Server include the `LoginEnvironment` object. From this point forward in this guide, references to the `loginEnv` variable assume that this step has been performed and that the variable is available.

3.2 Configuring the Logout

When finished making calls to the AIS Server, you must include the following logout call to end the user session:

```
AISClientUtilities.logout(loginEnv);
```

Using the AIS Client Class Generator

This chapter contains the following topic:

- [Section 4.1, "Understanding Generating Objects with the AIS Client Class Generator"](#)
- [Section 4.2, "Configuring the AIS Client Class Generator Preferences"](#)
- [Section 4.3, "Generating Data Classes Based on a Form"](#)
- [Section 4.4, "Generating Data Classes Based on a Data Request \(Available in AIS Client Class Generator v1.6.2\)"](#)

Important: If you have not yet installed the AIS Client Class Generator, see [Installing the AIS Client Class Generator Extension for JDeveloper](#) in this guide.

4.1 Understanding Generating Objects with the AIS Client Class Generator

A form service call to the AIS Server, otherwise 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.

4.2 Configuring the AIS Client Class Generator Preferences

The AIS Client Class Generator is available as a JDeveloper extension. You must install the extension before you can configure the preferences. See [Chapter 2, "Getting Started"](#) in this guide for instructions on how to install the extension.

To configure the AIS Client Class Generator:

1. In JDeveloper, access Preferences:
On Microsoft Windows, select the **Tools** menu, **Preferences**.
On Mac, 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 environment configured on the AIS Server.
- **Role.** (Optional) Enter a value only if you want to override the role configured on the AIS Server.
- **JSON Files Folder.** The folder for storing the JSON files. The default location is the AISClientClassGenerator\input directory.
- **Default Java Classes Folder.** The folder for storing generated Java files.
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.** The Java package name for the generated classes. The default is `com.oracle.e1.formservicetypes`.

4. Click **OK**.

4.3 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.

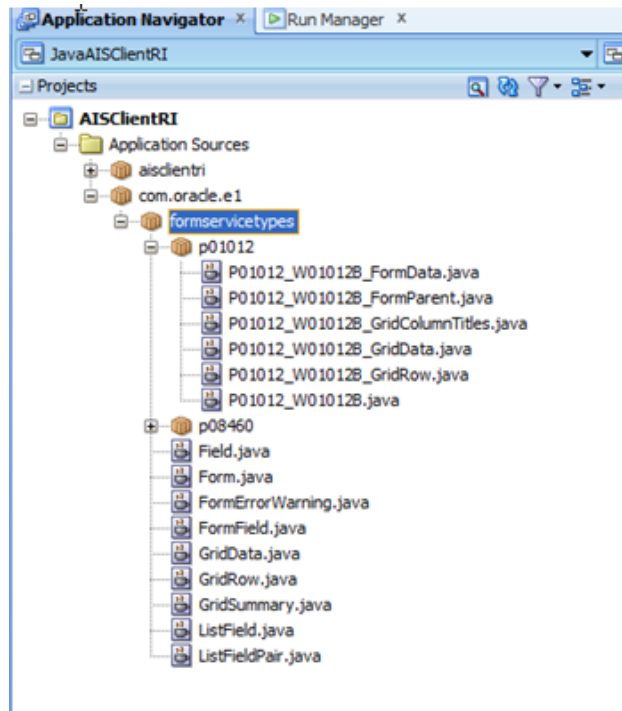
To use the AIS Client Class Generator to generate data classes:

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 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.
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 4–1 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 JDDeveloper 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.

If you need any fields on the form that have not been generated, you can add them manually. The code in [Example 4-2](#) shows an example of additional hidden grid columns added to the _GridRow class.

Example 4-2 Additional 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 setAddressLine1_40(Field sAddressLine1_40)
{
    this.sAddressLine1_40 = sAddressLine1_40;
}
public Field getSAddressLine1_40()
{
    return sAddressLine1_40;
}
public void setCity_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;
}

```

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

Starting with EnterpriseOne Tools 9.1.5.5, the dataservice endpoint is available to enable an AIS client 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.

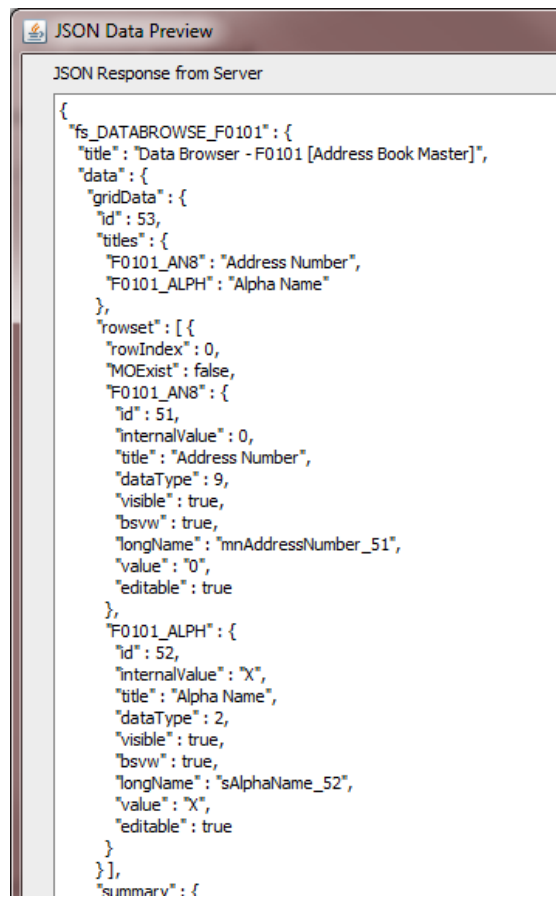
To use the AIS Client Class Generator to generate data classes based on the data request:

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.
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 4–3 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.



```

JSON Data Preview
JSON Response from Server
{
  "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": {

```

Performing AIS Form Service Calls

This chapter contains the following topics:

- [Section 5.1, "Understanding AIS Server Capabilities"](#)
- [Section 5.2, "Understanding Form Service Requests"](#)
- [Section 5.3, "Batch Form Service"](#)
- [Section 5.4, "Application Stack Service \(Tools Release 9.1.5\)"](#)
- [Section 5.5, "Media Object Operations"](#)
- [Section 5.6, "Processing Option Service"](#)
- [Section 5.7, "Task Authorization Service"](#)
- [Section 5.8, "Logging Service"](#)
- [Section 5.9, "Query \(Release 9.1.5.2\)"](#)
- [Section 5.10, "Jargon Service \(Release 9.1.5.3\)"](#)
- [Section 5.11, "Data Service \(API Release 1.1.0 and EnterpriseOne Tools Release 9.1.5.5\)"](#)
- [Section 5.12, "Orchestration Support \(API Release 1.1.0 and EnterpriseOne Tools Release 9.1.5.5\)"](#)

5.1 Understanding AIS Server Capabilities

The AIS Server exposes various capabilities that AIS client applications may or may not depend on. If your application requires a certain capability, you must include it in the list of required capabilities in the `LoginEnvironment` constructor.

If you included a capability in the list, the Login module verifies that capability is available when the application launches. If the capability is not available, the application returns an error message. If the capability is available, the application continues to the login screen. See [Chapter 3, "Configuring the Login Environment"](#) for more information.

You can access the AIS Server capabilities using the following URL:

`http://<AIS Server>:<Port>/jderest/defaultconfig`

The following code shows the available capabilities along with a description of each capability:

```
"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" : "oAuth",
  "shortDescription" : "OAM OAuth 2-Legged",
  "longDescription" : "Ability support 2-Legged OAM oAuth flow and
validate OAM Access Tokens supplied by a client",
  "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"  
    }  
},  
],
```

The code in [Example 5–1](#) shows the grid and editable capabilities listed in the `LoginEnvironment` constructor.

Example 5–1 Capabilities in LoginEnvironment Constructor

```
final String REQUIRED_CAP_LIST = "grid,query";  
th.loginEnv = new LoginEnvironment(AIS_SERVER, USER_NAME, PASSWORD,  
ENVIRONMENT, ROLE, DEVICE, REQUIRED_CAP_LIST, JAS_SERVER);
```

If the list includes a capability that is not available on the AIS Server, it throws a `CapabilityException`, as shown in [Example 5–2](#).

Example 5–2 Capability Exception

```
com.oracle.e1.aisclient.CapabilityException: Required Capabilities [grid,  
somethingelse] Available Capabilities: [grid, editable, log, processingOption,  
ignoreFDAFindOnEntry, selectAllGridRows, applicationStack, thumbnailSize,  
gridCellClick, query, taskAuthorization, urlMediaObjects, jargon, aliasNaming]
```

5.2 Understanding Form Service Requests

This section contains the following topics:

- [Section 5.2.1, "Overview"](#)
- [Section 5.2.2, "Form Service Request Structure"](#)
- [Section 5.2.3, "Control ID Notation for Return Control IDs"](#)
- [Section 5.2.4, "Reading Data"](#)
- [Section 5.2.5, "Adding Data"](#)
- [Section 5.2.6, "Deleting Data"](#)
- [Section 5.2.7, "Placing Events in the Proper Order"](#)
- [Section 5.2.8, "Considering Hidden Filters and Hidden QBE"](#)
- [Section 5.2.9, "Available Actions or Events"](#)

5.2.1 Overview

AIS Server calls that retrieve data from forms in the EnterpriseOne web client are referred to as form service requests. AIS client applications use form service requests to interact with EnterpriseOne web client forms. Form service requests, formatted as REST service calls that use POST, contain form service events or commands that invoke actions on an EnterpriseOne form.

A form service request enables you to perform various operations on a single form. By sending an ordered list of commands, a form service request can replicate the actions taken by an EnterpriseOne web client user, including populating fields, pressing buttons, and other actions.

To send a form service request to the AIS Server, send a POST to the following URL and send JSON in the body:

```
http://<AIS Server>:<Port>/formservice
```


If testing with a REST testing tool, you can send JSON directly.

The following list is an example of the operations required to perform a query in the find/browse form of the Address Book application (P01012_W01012B):

1. Enter a value into the Search Type field and into the QBE field for address number.
2. Click the check boxes to show the extra grid columns and press the **Find** button.

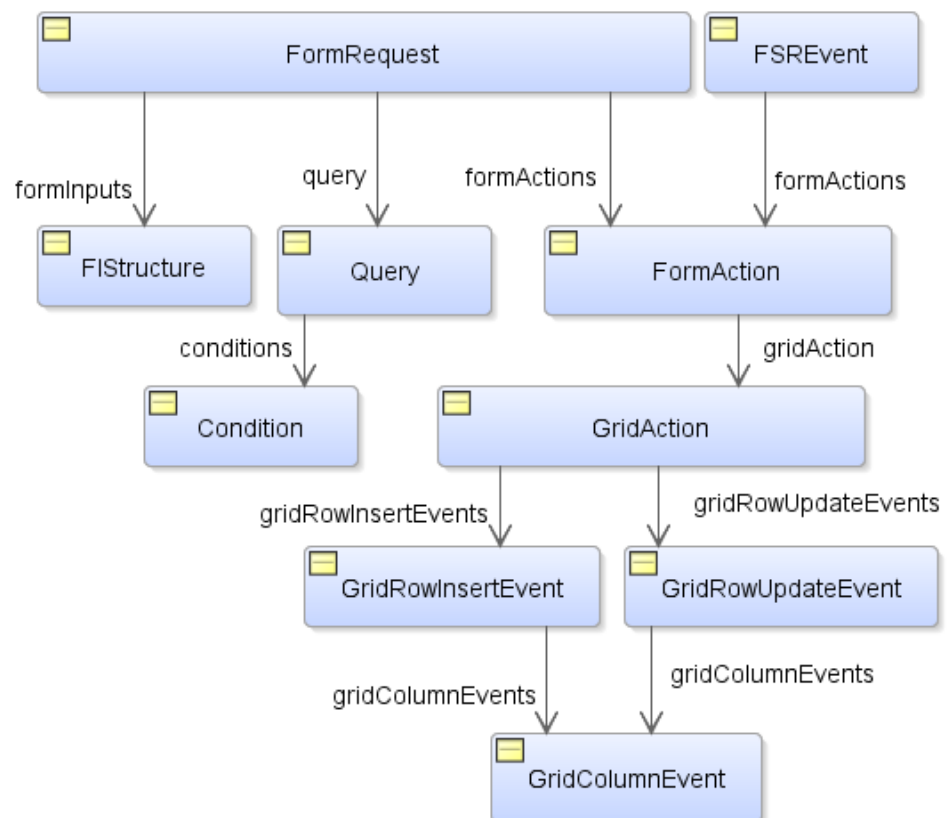
This populates the grid with the data matching the query.

The form service returns the form parent object representing the form after it is populated with the data.

5.2.2 Form Service Request Structure

The class diagram in [Figure 5–1](#) represents the basic structure of a form service request. The collections under `FormRequest` are optional (0 to many); you do not have to have `FIStructures`, `FormActions`, `GridActions`, and so forth. A form service request (FSR) event is a set of `FormActions` that you first compile into an FSR event and then add to the `FormRequest` using the `add` method.

Figure 5–1 Form Service Request Structure Diagram



5.2.3 Control ID Notation for Return Control IDs

You can use the Property Browser in FDA to identify control IDs for fields on each EnterpriseOne form. You can also find control IDs using the Item Help option in the form in the EnterpriseOne web client. In the EnterpriseOne web client form, click the

Help button (question mark in the upper right corner of a form) and then click the Item Help option to access field-level help. With the field level help activated, you can click in a field or column to access the control ID and business view information, which is displayed under the Advanced Options section.

For fields on the main form, the control ID will be a single value, such as 25.

Grids also have control IDs. For a traditional form, the grid ID is usually 1. For power forms, subforms, and reusable subforms the grid ID is typically a value other than 1.

The columns within a grid also have unique IDs and are often referenced in conjunction with the grid ID. For example column 28 and 29 in grid 1 would be 1[28,29].

Power forms have more complex IDs. The fields on the main power form are represented with single values. The fields on a subform are complex with an underscore separating them. So field 6 on subform 12 is 12_6. The ID of a re-usable subform is available when viewing the power form that the subform is used on. The IDs of individual fields, a grid, or columns on a re-usable subform is shown in FDA when viewing the subform directly; you cannot get these values when viewing the subform alias.

The returnControlIDs string is bar delimited, without a starting or ending bar.

Example 5-3 Requesting fields and grid columns on a traditional form.

```
formRequest.setReturnControlIDs("19|20|60|125|1[45,49,88]");
```

In this example, 19|20|60|125 represent field control IDs.

1[45,49,88] represents columns in the grid.

Example 5-4 Requesting main form fields, subform fields, main form grid columns, and subform grid columns.

```
formRequest.setReturnControlIDs("33|34|17[24,26,28]|50_45|50_53|50_9[35,39,41]");
```

In this example, 33|34 represent fields on the main form.

50_45|50_53 represent fields on the subform.

17[24,26,28] represent main form grid columns.

50_9[35,39,41] represent subform grid columns.

5.2.4 Reading Data

The code in [Example 5-5](#) shows an example of a form service request that reads data from an EnterpriseOne form. In this example, the code results in populating the P01012 form parent object with data that can be displayed or manipulated.

Example 5-5 Form Service Request for Reading Data

```
public P01012_W01012B_FormParent P01012()
{
    P01012_W01012B_FormParent p01012form = null;

    try{
        //populate the request information
        FormRequest formRequest = new FormRequest(loginEnv);
        formRequest.setFormName("P01012_W01012B");
        formRequest.setFormServiceAction("R");
```

```

formRequest.setMaxPageSize("30"); //only get 30 records
formRequest.setReturnControlIDs("54|1[19,20]");

FSREvent fsrEvent = new FSREvent();

fsrEvent.setFieldValue("54", "E"); //customers
//include >= operator in QBE
fsrEvent.setQBEValue("1[19]", ">=" + "6001");
fsrEvent.checkBoxChecked("62"); //show address
fsrEvent.checkBoxChecked("63"); //show phone
fsrEvent.doControlAction("15"); //find

formRequest.addFSREvent(fsrEvent); //add the events to the request
String response = JDERestServiceProvider.jdeRestServiceCall(loginEnv, formRequest,
JDERestServiceProvider.POST_METHOD, JDERestServiceProvider.FORM_SERVICE_URI);

//de-serialize the JSON string into the form parent object
p01012form = loginEnv.getMapper().readValue(response, P01012_W01012B_
FormParent.class);
}
catch(JDERestServiceException e)
{
    //get more specific error string
    String error = JDERestServiceProvider.handleServiceException(e);
    System.out.println(error);
}
catch(Exception e)
{
    //handle other exceptions
    System.out.println(e.getMessage());
}

return p01012form;
}

```

5.2.5 Adding Data

The code in [Example 5–6](#) shows an example of a form service request that adds a new phone number in the EnterpriseOne phones application and saves it. After saving the phone number, the form service sends a response with the new number in the grid.

Example 5–6 Form Service Request for Adding Data

```

public P0115_W0115A_FormParent addPhone(){
P0115_W0115A_FormParent p0115_W0115A = null;

//indicate using grid capability
//(alternatively could use required capability)
loginEnv.getUsedCapabilities().add("grid");

if (AISClientCapability.isCapabilityAvailable(loginEnv, "grid"))
{
    try{

FormRequest formRequest = new FormRequest(loginEnv);
formRequest.setFormName("P0115_W0115A");

```

```
formRequest.setFormServiceAction("U");

//open this form with specific record for AB 7500, contact 0
formRequest.addToFISet("4", "7500");
formRequest.addToFISet("5", "0");

FSREvent fsrEvent = new FSREvent();
//create grid action
GridAction gridAction = new GridAction(loginEnv);
//create grid row insert event
GridRowInsertEvent gri = new GridRowInsertEvent();

//set the column values
gri.setGridColumnValue("27", "HOM");
gri.setGridColumnValue("28", "303");
gri.setGridColumnValue("29", "123-4567");

//add the row to grid ID "1"
gridAction.insertGridRow("1", gri);

//add the grid action to the events
fsrEvent.addGridAction(gridAction);

//press OK button
fsrEvent.doControlAction("4");

//add the FSR event to the request
formRequest.addFSREvent(fsrEvent);

String response = JDERestServiceProvider.jdeRestServiceCall(loginEnv, formRequest,
JDERestServiceProvider.POST_METHOD, JDERestServiceProvider.FORM_SERVICE_URI);

//de-serialize the JSON string into the form parent object
p0115_W0115A = loginEnv.getMapper().readValue(response, P0115_W0115A_
FormParent.class);
    }
    catch(CapabilityException e)
    {
//handle capability exception
        System.out.println(e.getMessage());
    }
    catch(JDERestServiceException e)
    {
//get more specific error string
        String error = JDERestServiceProvider.handleServiceException(e);
        System.out.println(error);
    }
}

return p0115_W0115A;
}
```

5.2.6 Deleting Data

The code in [Example 5–7](#) shows an example of a form service request that deletes the phone at index 0 and returns a response with a set of records without the removed phone number record.

Example 5-7 Form Service Request for Deleting Data

```

public P0115_W0115A _FormParent deletePhone(){

    P0115_W0115A _FormParent p0115_W0115A = null;
    try{

        FormRequest formRequest = new FormRequest(loginEnv);
        formRequest.setFormName("P0115_W0115A");
        formRequest.setFormServiceAction(formRequest.ACTION_UPDATE);

        //open form with record for AB 7500 contact 0
        formRequest.addToFISet("4", "7500");
        formRequest.addToFISet("5", "0");

        FSREvent fsrEvent = new FSREvent();

        //select the row to delete from grid with ID "1", based on row index 0
        fsrEvent.selectRow("1", 0);

        //press Delete button
        fsrEvent.doControlAction("59");

        //press OK button
        fsrEvent.doControlAction("4");

        //add the FSR event to the request
        formRequest.addFSREvent(fsrEvent);

        String response = JDERestServiceProvider.jdeRestServiceCall(loginEnv, formRequest,
            JDERestServiceProvider.POST_METHOD, JDERestServiceProvider.FORM_SERVICE_URI);

        //de-serialize the JSON string into the form parent object
        p0115_W0115A = loginEnv.getMapper().readValue(response, P0115_W0115A_
            FormParent.class);
    }
    catch(JDERestServiceException e)
    {
        //get more specific error string
        String error = JDERestServiceProvider.handleServiceException(e);
        System.out.println(error);
    }
    catch(Exception e)
    {
        //handle other exceptions
        System.out.println(e.getMessage());
    }

    return p0115_W0115A;
}

```

5.2.7 Placing Events in the Proper Order

Place the events in the request in the order you want them to execute, for example, populate a filter field value and then press the Find button. Remember that the FDA Form Service Request event occurs before the events you add to this list. Do not set the Find On Entry option when using the event model; the extra "find" is not necessary because it executes before the events you requested.

5.2.8 Considering Hidden Filters and Hidden QBE

By default, values are not written to hidden filter fields or hidden QBE columns. You must use the Form Service Event in FDA to show the fields and columns first. Then you can add values to these fields and subsequently run the query.

5.2.9 Available Actions or Events

The preceding examples in this chapter only show some of the operations you can perform in a form service request. The tables in this section describe other operations you may want to perform.

Table 5–1 Form Service Request Events

Action or Event	Description	Parameters
Set Control Value	Sets the value of a control on a form, like filter fields or any other form control.	controlID ("25") value("Bob" or "01/01/2015")
Set QBE Value	Sets the value of a QBE column	controlID ("1[42]" or "1_2[25]") value ("Jill" or "55")
Set Checkbox Value	Sets the value of a check box	controlID ("77") value ("on" or "off")
Set Radio Button	Sets the value of a radio button	controlID ("87") value ("87")
Set Combo Value	Sets the value of a combo box entry	controlID ("125") value (2) - index of the entry
Do Action	Presses a button or Hyper Item	controlID ("156")
Select Row	Selects the specified row in a grid	controlID ("1.30") - ID of the Grid dot then row index (zero based).
Select All Rows*	Select all rows in the specified grid (if multiple selection is allowed)	controlID ("1") - ID of the Grid
Un Select All Rows*	Un-selects all rows in the specified grid (if multiple selection is allowed)	controlID ("1") - ID of the Grid
Un Select Row*	Un-selects the specified row in a grid	controlID ("1.30") - ID of the Grid dot then row index (zero based).
Click Grid Cell**	Clicks the hyperlink in a grid cell (if the cell is enabled as a link)	controlID ("1.5.22") - ID of the g Grid dot then row index dot grid column id.

*New event available in 9.1.5.

**New event available in 9.1.5.2.

In addition to interacting with fields on the form, you can interact with grids using grid action events. If you use a grid action event, you must include "grid" as a required capability in the LoginEnvironment constructor. See [Understanding AIS Server](#)

[Capabilities](#) for more information.

The types of grid action events include:

- Selecting grid rows

This action enables you to delete records in the grid by sending a row select event, followed by a delete button press event, and then finally an OK button press event. This is the exact sequence that a user would follow to delete a record in an EnterpriseOne application.

- Inserting grid rows

This action enables you to insert one or more rows into a grid, setting the column value for each row. This includes text entry columns, drop-down columns, or check box columns. You must include an OK button pressed event to commit the inserts.

- Updating grid rows

This action enables you to update one or more existing grid rows by setting the column values for each row. This includes text entry columns, drop-down columns, or check box columns. You must include an OK button pressed event to commit the updates.

The following table describes the commands that you can use in grid column events to set values for a cell in a grid insert or update event:

Table 5–2 Grid Column Events in a Form Service Request

Grid Column Events	Description	Parameters
Set Grid Cell Value	Sets the value of a cell in a grid.	"value": "720", "command": "SetGridCellValue", "columnID": "28"
Set Grid Combo Value	Sets the value of a dropdown column in a grid. The value you send is the 'Code' for the UDC associated with that column.	"value": "ABC", "command": "SetGridComboValue", "columnID": "43"

5.3 Batch Form Service

If you make several sequential calls to forms without any data dependencies between them, consider using the Batch Form Service. Batch form service requests are used to execute multiple EnterpriseOne forms during a single request, which improves your AIS client's performance.

Use the AIS Client Class Generator to generate the classes for all the forms that you need to call in the batch request. Then declare a parent class that contains all of the same forms in the order in which they appear in the batch request (including an index number).

[Example 5–8](#) shows a batch form service request that calls the same form three times with different inputs each time, followed by a call to another form.

Example 5–8 Batch Form Service Request

```
public class BatchRequestParent {

    private P54HS220_W54HS220A fs_0_P54HS220_W54HS220A;
    private P54HS220_W54HS220A fs_1_P54HS220_W54HS220A;
    private P54HS220_W54HS220A fs_2_P54HS220_W54HS220A;
```

```
private P54HSPT_S54HSPTA fs_3_P54HSPT_S54HSPTA;

public BatchRequestParent() {
    super();
}

public void setFs_0_P54HS220_W54HS220A(P54HS220_W54HS220A fs_0_P54HS220_
W54HS220A) {
    this.fs_0_P54HS220_W54HS220A = fs_0_P54HS220_W54HS220A;
}

public P54HS220_W54HS220A getFs_0_P54HS220_W54HS220A() {
    return fs_0_P54HS220_W54HS220A;
}

public void setFs_1_P54HS220_W54HS220A(P54HS220_W54HS220A fs_1_P54HS220_
W54HS220A) {
    this.fs_1_P54HS220_W54HS220A = fs_1_P54HS220_W54HS220A;
}

public P54HS220_W54HS220A getFs_1_P54HS220_W54HS220A() {
    return fs_1_P54HS220_W54HS220A;
}

public void setFs_2_P54HS220_W54HS220A(P54HS220_W54HS220A fs_2_P54HS220_
W54HS220A) {
    this.fs_2_P54HS220_W54HS220A = fs_2_P54HS220_W54HS220A;
}

public P54HS220_W54HS220A getFs_2_P54HS220_W54HS220A() {
    return fs_2_P54HS220_W54HS220A;
}

public void setFs_3_P54HS230_W54HS230A(P54HS230_W54HS230A fs_3_P54HS230_
W54HS230A) {
    this.fs_3_P54HS230_W54HS230A = fs_3_P54HS230_W54HS230A;
}

public P54HS230_W54HS230A getFs_3_P54HS230_W54HS230A() {
    return fs_3_P54HS230_W54HS230A;
}

public void setFs_4_P54HS240_W54HS240A(P54HS240_W54HS240A fs_4_P54HS240_
W54HS240A) {
    this.fs_4_P54HS240_W54HS240A = fs_4_P54HS240_W54HS240A;
}

public P54HS240_W54HS240A getFs_4_P54HS240_W54HS240A() {
    return fs_4_P54HS240_W54HS240A;
}

public void setFs_3_P54HSPT_S54HSPTA(P54HSPT_S54HSPTA fs_3_P54HSPT_S54HSPTA) {
    this.fs_3_P54HSPT_S54HSPTA = fs_3_P54HSPT_S54HSPTA;
}

public P54HSPT_S54HSPTA getFs_3_P54HSPT_S54HSPTA() {
    return fs_3_P54HSPT_S54HSPTA;
}
}
```


Example 5–9 Deserialize the Response to the BatchRequestParent

This sample code shows how after calling forms, you can call the service and deserialize the response to the BatchRequestParent.

```
public BatchRequestParent batchRequest(){

    BatchRequestParent batchParent = null;
    try{
        // Get resource bundle for incident category text
        BatchFormRequest batchFormRequest = new BatchFormRequest(loginEnv);

        //recentIncidents - Index 0
        SingleFormRequest formRequest = new SingleFormRequest();
        //formRequest.setFindOnEntry("TRUE");

        formRequest.setReturnControlIDs("1[19,20,21,27,28,41,45,46,47,48,49,50,51,52,54,55,92,174,177,178,181]");
        formRequest.setFormName("P54HS220_W54HS220A");

        //create event holder
        FSREvent recentFSREvent = new FSREvent();
        //add filter actions in order
        // Incident From Date
        recentFSREvent.setFieldValueDate(loginEnv, "150", cal.getTime());
        // Potential Incident
        recentFSREvent.setQBEValue("1[30]", "0");
        // Exclude from Safety Statistics
        recentFSREvent.setQBEValue("1[39]", "0");
        // Press Find Button
        recentFSREvent.doControlAction("15");
        //add event holder to the form request
        formRequest.addFSREvent(recentFSREvent);

        batchFormRequest.getFormRequests().add(formRequest);

        //recentInjuryIllnessIncidents - Index 1
        formRequest = new SingleFormRequest();
        //formRequest.setFindOnEntry("TRUE");

        formRequest.setReturnControlIDs("1[19,20,21,27,28,41,45,46,47,48,49,50,51,52,54,55,92,174,177,178,181]");
        formRequest.setFormName("P54HS220_W54HS220A");

        //create event holder
        FSREvent injuryFSREvent = new FSREvent();
        //add filter actions in order
        // Incident From Date
        injuryFSREvent.setFieldValueDate(loginEnv, "150", cal.getTime());
        // Potential Incident
        injuryFSREvent.setQBEValue("1[30]", "0");
        // Exclude from Safety Statistics
        injuryFSREvent.setQBEValue("1[39]", "0");
        // Injury/Illness checkbox
        injuryFSREvent.setQBEValue("1[33]", "1");
        // Press Find Button
        injuryFSREvent.doControlAction("15");
        //add event holder to the form request
        formRequest.addFSREvent(injuryFSREvent);
```

```
        batchFormRequest.getFormRequests().add(formRequest);

        // recentEnvironmentalIncidents - Index 2
        formRequest = new SingleFormRequest();
        //formRequest.setFindOnEntry("TRUE");

formRequest.setReturnControlIDs("1[19,20,21,27,28,41,45,46,47,48,49,50,51,52,54,55
,92,174,177,178,181]");
        formRequest.setFormName("P54HS220_W54HS220A");

        //create event holder
        FSREvent environFSREvent = new FSREvent();
        //add filter actions in order
        // Incident From Date
        environFSREvent.setFieldValueDate(loginEnv, "150", cal.getTime());
        // Potential Incident
        environFSREvent.setQBEValue("1[30]", "0");
        // Exclude from Safety Statistics
        environFSREvent.setQBEValue("1[39]", "0");
        // Environmental checkbox
        environFSREvent.setQBEValue("1[34]", "1");
        // Press Find Button
        environFSREvent.doControlAction("15");
        //add event holder to the form request
        formRequest.addFSREvent(environFSREvent);

        batchFormRequest.getFormRequests().add(formRequest);

        // scoreboard - Index 3
        formRequest = new SingleFormRequest();
        formRequest.setFindOnEntry("TRUE");
        formRequest.setReturnControlIDs("1_20|1_22");
        formRequest.setFormName("P54HSPT_S54HSPTA");
        batchFormRequest.getFormRequests().add(formRequest);

        String response = JDERestServiceProvider.jdeRestServiceCall(loginEnv,
batchFormRequest, JDERestServiceProvider.POST_METHOD,
JDERestServiceProvider.BATCH_FORM_SERVICE_URI);

        //de-serialize the JSON string into the batchParent object
        batchParent = loginEnv.getMapper().readValue(response,
BatchRequestParent.class);
    }
    catch(JDERestServiceException e)
    {
        //get more specific error string
        String error = JDERestServiceProvider.handleServiceException(e);
        System.out.println(error);
    }
    catch(Exception e)
    {
        //handle other exceptions
        System.out.println(e.getMessage());
    }
}
```

```

    return batchParent;
}

```

5.4 Application Stack Service (Tools Release 9.1.5)

The application stack service enables an AIS client to interact with multiple applications running in an ongoing EnterpriseOne web client session. The application stack service enables more complex interactions with applications that have cross-form transaction boundaries, for example where you do not want to save the header until the details are added.

The application stack service supports form interconnects in EnterpriseOne to receive data from the resulting form. For example, you may want to use an existing sequence of tasks in EnterpriseOne that involves interacting with multiple forms to perform a transaction: open an initial form; select a record and navigation to a second form; perform an update that might automatically flow to a third form where you enter more data; and then finally complete the transaction. The application stack service allows for this type of interaction with EnterpriseOne forms.

To use the application stack service, you must first create an `ApplicationStack` object, which contains these three types of operations:

- **Open.** Open starts a new stack, opening the first form and performing any operations included in the `FormRequest`.
- **Execute.** Subsequent actions on that application stack must use the `Execute` operation, where you can pass an `ActionRequest` with any actions to be performed on the currently open form.
- **Close.** You can pass a `Close` operation to close the stack and any open forms on it.

Each response to a stack request includes the current form which might be the form originally requested, or it could be a new form if navigation to a new form occurred.

Make sure that you are executing actions on the right form. You should use the `getLastAppStackResponse().checkSuccess` method before executing actions so you can be sure of the current form. You must include the form in the request for actions. If the form in the request does not match the current form on the stack, the actions will not execute.

The sample code in [Example 5–10](#) performs operations in a stack of applications in this order:

1. Opens the stack first with the Address Book find/browse form (P01012_W0101B).
2. Executes an action to select a record on that form.
3. Executes another action on the P01012_W01012A form and updates the Name field.
4. Executes another action to press the OK button.
5. Executes another action to press the Close button on W01012A to close the form.
6. Closes the stack.

Example 5–10 Application Stack

```

public void appStack() throws Exception
{

    loginEnv.getUsedCapabilities().add("applicationStack");
}

```

```
ApplicationStack appStackAddress = new ApplicationStack();
FormRequest formRequest = new FormRequest(loginEnv);
formRequest.setReturnControlIDs("1");
formRequest.setFormName("P01012_W01012B");

formRequest.setReturnControlIDs("54|1[19,20]");
formRequest.setFormServiceAction("R");
formRequest.setMaxPageSize("5");
FSREvent findFSREvent = new FSREvent();

findFSREvent.setFieldValue("54", "E");
findFSREvent.doControlAction("15"); // Find button
formRequest.addFSREvent(findFSREvent);

ObjectWriter writer = loginEnv.getObjectMapper().writerWithDefaultPrettyPrinter();
out.println(writer.writeValueAsString(formRequest));

//open P01012_W01012B
String response = appStackAddress.open(loginEnv, formRequest);
out.println(writer.writeValueAsString(loginEnv.getObjectMapper().readTree(response)));

//check if in find browse
if (appStackAddress.getLastAppStackResponse().checkSuccess("P01012_W01012B"))
{
    //select a record
    ActionRequest actionRequest = new ActionRequest();
    actionRequest.setReturnControlIDs("28"); //the form changes these return control
    IDs are for the next form
    actionRequest.setFormOID("W01012B");
    FSREvent selectFSREvent = new FSREvent();
    selectFSREvent.selectRow("1", 3);
    selectFSREvent.doControlAction("14"); //select button
    actionRequest.addFSREvent(selectFSREvent);

    response = appStackAddress.executeActions(loginEnv, actionRequest);
    out.println(writer.writeValueAsString(loginEnv.getObjectMapper().readTree(response)));

    //check if in fix inspect
    if (appStackAddress.getLastAppStackResponse().checkSuccess("P01012_W01012A"))
    {
        //Change name - now on form A
        ActionRequest actionRequestName = new ActionRequest();
        actionRequestName.setReturnControlIDs("54|1[19,20]"); //form is going to change
        again these are for the next form
        actionRequestName.setFormOID("W01012A");
        FSREvent updateFSREvent = new FSREvent();

        updateFSREvent.setFieldValue("28", "AIS APP Stack TEST"); //change name field
        updateFSREvent.doControlAction("11"); //ok

        actionRequestName.addFSREvent(updateFSREvent);

        response = appStackAddress.executeActions(loginEnv, actionRequestName);
        out.println(writer.writeValueAsString(loginEnv.getObjectMapper().readTree(response)));

        //IMPORTANT: here you would have to de-serialize the response to check if there
```

```

were errors on the form after pressing okay, if so you could continue to close the
A form and go back to the B form
if (appStackAddress.getLastAppStackResponse().checkSuccess("P01012_W01012A"))
{
    //press find again (to see name change) then close the stack
    ActionRequest actionRequestClose = new ActionRequest();
    FSREvent closeFSREvent = new FSREvent();
    actionRequestClose.setReturnControlIDs("54|1[19,20]"); //form is changing these
    are the controls of the returned form
    actionRequestClose.setFormOID("W01012A");
    closeFSREvent.doControlAction("12"); //close
    actionRequestClose.addFSREvent(closeFSREvent);

    response = appStackAddress.close(loginEnv, actionRequestClose);
    out.println(writer.writeValueAsString(loginEnv.getMapper().readTree(response
    )));
}
}

}

}

```

5.5 Media Object Operations

Media objects in EnterpriseOne store file attachments and text attachments. The media object operations in the AIS Client Java API use the following items to identify individual media object attachments for a record:

- Media object name, for example GT00202.
- Media object key to identify the record. This is a bar delimited key string, for example 6540|3|1.
- Sequence number to identify the individual attachment for a record.

5.5.1 GetText

This operation returns the text in the first text attachment, as shown in the code in [Example 5-11](#):

Example 5-11 Media Object Get Text Operation

```

try{
    MediaObjectGetTextRequest moGetText = new MediaObjectGetTextRequest(loginEnv);

    moGetText.setFormName("P01012_W01012B");
    moGetText.setVersion("ZJDE0001");
    moGetText.setMoStructure("ABGT");

    //set mo key - in this case it's just AB number
    moGetText.addMoKeyValue("7");

    MediaObjectGetTextResponse response =
    MediaObjectOperations.getTextMediaObject(loginEnv, moGetText);

    System.out.println(response.getText());
}
catch (Exception e){

```

```
//handle exception
}
```

5.5.2 Update Text

This operation updates (replaces or appends to) the first text media object, as shown in the code in [Example 5–12](#):

Example 5–12 Media Object Update Text Operation

```
try{
MediaObjectUpdateTextRequest moSetText = new
MediaObjectUpdateTextRequest(loginEnv);
moSetText.setFormName("P01012_W01012B");
moSetText.setVersion("ZJDE0001");
moSetText.setMoStructure("ABGT");

//set mo key
moSetText.addMoKeyValue("7");
moSetText.setAppendText(true);
//set text
moSetText.setInputText("Append This text");

MediaObjectUpdateTextResponse response =
MediaObjectOperations.updateTextMediaObject(loginEnv, moSetText);

System.out.println("Status " + response.getUpdateTextStatus());
}
catch (Exception e){
//handle exception
}
```

5.5.3 List

MediaObjectListRequest is the input to the media object getMediaObjectList operation. The following table describes the attributes in the request that control the list that is returned:

Field	Type	Description
includeURLs	boolean	Valid values are: <ul style="list-style-type: none">■ True■ False When true, includes the URL for downloading the media object, which can be used in conjunction with a download request at a later time. This only applies to the media object file type.
includeData	boolean	Valid values are: <ul style="list-style-type: none">■ True■ False When true, if the file is an image, it includes the base64 encoded data for a thumbnail sized image.

Field	Type	Description
moTypes	String	Use a constant defined in <code>MediaObjectListRequestValid</code> , which includes these constants: <ul style="list-style-type: none"> ■ <code>MediaObjectListRequest.MO_TYPE_TEXT</code> ■ <code>MediaObjectListRequest.MO_TYPE_FILE</code> ■ <code>MediaObjectListRequest.MO_TYPE_QUEUE</code> ■ <code>MediaObjectListRequest.MO_TYPE_URL</code>
extensions	String	File extensions to include in the response, which enables you to filter out undesired extensions.
thumbnailSize	<String> int	Size of the thumbnail image returned as base64 data.

The code in [Example 5–13](#) shows an example of saving the set of thumbnail images for image media object attachments. It includes specified extensions for the first file type attachments. The `includeData` value is set to include the thumbnail data. If the file is not a PDF (a non-image type), the thumbnail data is saved to a local file.

Example 5–13 Saving Thumbnail Images for Image Media Object Attachments

```
import Java.awt.image.BufferedImage;
import sun.misc.BASE64Decoder;

public void listMediaObject() throws Exception
{
    final String MO_STRUCTURE = "ABGT";
    final String MO_APP = "P01012_W01012B";
    final String MO_VERSION = "ZJDE0001";
    final String MO_KEY = "479";
    final int MO_THUMBSIZE = 50;
    final String FILE_LOCATION = "C:\\temp\\AISClientDownloads\\";

    //set request info include URLs so they don't have to be fetched later
    MediaObjectListRequest mediaObjectListRequest = new
    MediaObjectListRequest(loginEnv);
    mediaObjectListRequest.setFormName(MO_APP);
    mediaObjectListRequest.setVersion(MO_VERSION);
    mediaObjectListRequest.setIncludeURLs(false);
    mediaObjectListRequest.setIncludeData(true);
    mediaObjectListRequest.setMoStructure(MO_STRUCTURE);
    mediaObjectListRequest.setThumbnailSize(MO_THUMBSIZE); //available in tools 9.1.5+
    only

    //set the moKey
    mediaObjectListRequest.addMoKeyValue(MO_KEY);

    // - Date Example, if MO key includes a date value -
    //mediaObjectListRequest.addMoKeyValue(AISClientUtilities.convertMillisecondsToYMD
    String(mydate.getTime()));

    //I only want filesmediaObjectListRequest.addMoType(mediaObjectListRequest.MO_
    TYPE_FILE);mediaObjectListRequest.addMoType(mediaObjectListRequest.MO_TYPE_QUEUE);

    //I only want these types
```

```
mediaObjectListRequest.addExtension("jpg");
mediaObjectListRequest.addExtension("gif");
mediaObjectListRequest.addExtension("jpeg");
mediaObjectListRequest.addExtension("pdf");

//get the list of available files for this media object
MediaObjectListResponse mediaObjectListResponse =
MediaObjectOperations.getMediaObjectList(loginEnv, mediaObjectListRequest);

if (mediaObjectListResponse != null)
{

for (int i = 0; i < mediaObjectListResponse.getMediaObjects().length; i++)
{

FileAttachment fileAt = new FileAttachment();
fileAt.setThumbFileLocation(mediaObjectListResponse.getMediaObjects()[i].getThumbFileLocation());
fileAt.setItemName(mediaObjectListResponse.getMediaObjects()[i].getItemName());
fileAt.setFileName(mediaObjectListResponse.getMediaObjects()[i].getFile());
fileAt.setDownloadUrl(mediaObjectListResponse.getMediaObjects()[i].getDownloadUrl());
fileAt.setSequence(mediaObjectListResponse.getMediaObjects()[i].getSequence());

//if it's an image, save the thumbnail data to a file
if (!fileAt.getFileName().contains("pdf"))
{
BufferedImage image =
decodeToImage(mediaObjectListResponse.getMediaObjects()[i].getData());
if (image != null)
{
File file = new File(fileAt.getFileName());

File outputfile = new File(FILE_LOCATION + "thumb_" + file.getName());
ImageIO.write(image, "jpg", outputfile);
}

}

}

}

public static BufferedImage decodeToImage(String imageString)
{

BufferedImage image = null;
byte[] imageByte;
try
{
BASE64Decoder decoder = new BASE64Decoder();
imageByte = decoder.decodeBuffer(imageString);
ByteArrayInputStream bis = new ByteArrayInputStream(imageByte);
image = ImageIO.read(bis);
bis.close();
}
catch (Exception e)
{
e.printStackTrace();
}

}
```



```
        return image;
    }
}
```

5.5.4 Upload

To upload a file, you need to provide the media object data structure key information:

- A string with the location of the local file to be uploaded.
- A name for the item. If you do not supply a name, the file name is used.

The code in [Example 5-14](#) uploads a file to the Address Book media object for address book number 479. The response to the upload request will print the name and sequence number of the new record.

Example 5-14 Media Object Upload

```
public void uploadFile(String fileLocation, String itemName) throws Exception
{

    final String MO_STRUCTURE = "ABGT";
    final String MO_APP = "P01012_W01012B";
    final String MO_VERSION = "ZJDE0001";
    final String MO_KEY = "479";

    MediaObjectUploadRequest mediaObjectUploadRequest = new
    MediaObjectUploadRequest(loginEnv);
    mediaObjectUploadRequest.setFormName(MO_APP);
    mediaObjectUploadRequest.setVersion(MO_VERSION);
    mediaObjectUploadRequest.setMoStructure(MO_STRUCTURE);

    //set the moKey
    mediaObjectUploadRequest.addMoKeyValue(MO_KEY);

    String fileLocation = "C:\\temp\\images\\IMG_20001.jpg";
    String itemName = "Joe's Photo";
    FileAttachment newFileAttachment = new FileAttachment();
    newFileAttachment.setFileLocation(fileLocation);
    newFileAttachment.setItemName(itemName);

    //set the file to the new one they just saved
    mediaObjectUploadRequest.setFile(newFileAttachment);

    //Upload to Server
    MediaObjectUploadResponse response =
    MediaObjectOperations.uploadMediaObject(loginEnv, mediaObjectUploadRequest);

    out.println("NEW MO: " + response.getItemName());
    out.println("NEW MO SEQ: " + response.getSequence());

}
```

5.5.5 Download

To download a file, you can provide the following input:

- **downloadURL.** (String) (Optional) If you requested this value from the list request, send it to the server and it will save the step of fetching this URL. If you do not pass a value, the URL will be fetched by AIS.

- **sequence.** (int) (Required) The sequence number of the attachment for this media object record.
- **height.** (int) (Optional) If the file you are downloading is an image, the AIS Server will scale the image to the requested height.
- **width.** (int) (Optional) If the file you are downloading is an image, the AIS Server will scale the image to the requested width.
- **fileName.** (String) (Required) Provide a name for the downloaded file, if desired you can use the same name returned in the list response.

The code in [Example 5–15](#) is an example of downloading a media object attachment. Executing the *getMediaObjectList* operation produces a *FileAttachment* object that contains the sequence and media object file name. It passes the *FileAttachment* object into this method where a call is made to *downloadMediaObject* operation (passing a desired file location). The response will include the location of the saved file.

Example 5–15 Media Object Download

```
public void downloadFile(FileAttachment fileAt) throws Exception
{

    final String MO_STRUCTURE = "ABGT";
    final String MO_APP = "P01012_W01012B";
    final String MO_VERSION = "ZJDE0001";
    final String MO_KEY = "479";
    final String FILE_LOCATION = "C:\\temp\\AISClientDownloads\\";

    //set the download request info - don't need mo key because we have the list
    already
    MediaObjectDownloadRequest mediaObjecDownloadRequest = new
    MediaObjectDownloadRequest(loginEnv);

    mediaObjecDownloadRequest.setFormName(MO_APP);
    mediaObjecDownloadRequest.setVersion(MO_VERSION);
    mediaObjecDownloadRequest.setMoStructure(MO_STRUCTURE);
    mediaObjecDownloadRequest.setWidth(700);
    mediaObjecDownloadRequest.addMoKeyValue(MO_KEY);
    mediaObjecDownloadRequest.setSequence(fileAt.getSequence());
    mediaObjecDownloadRequest.setFileName(fileAt.getFileName());

    // download the file and save to file location
    MediaObjectDownloadResponse mediaObjecDownloadResponse =
    MediaObjectOperations.downloadMediaObject(loginEnv, mediaObjecDownloadRequest,
    FILE_LOCATION);

    out.println("Downloaded File: " +
    mediaObjecDownloadResponse.getFile().getFileLocation());

}
```

5.5.6 Add URL (Tools Release 9.1.5.2 and API Version 1.0)

To add a URL type media object, provide the media object keys as well as the URL text, such as:

```
http://www.domainname.com.
```

The code in [Example 5–16](#) shows an example of adding a URL type media object:

Example 5–16 Adding a URL Media Object

```
public void addURL() throws Exception {

    final String MO_STRUCTURE = "ABGT";
    final String MO_APP = "P01012_W01012B";
    final String MO_VERSION = "ZJDE0001";
    final String MO_KEY = "479";
    final String URL_TEXT = "http://www.google.com";

    //set request info include URLs so they don't have to be fetched later
    MediaObjectAddUrlRequest mediaObjectAddUrlRequest = new
    MediaObjectAddUrlRequest(loginEnv);

    mediaObjectAddUrlRequest.setFormName(MO_APP);
    mediaObjectAddUrlRequest.setVersion(MO_VERSION);
    mediaObjectAddUrlRequest.setMoStructure(MO_STRUCTURE);

    mediaObjectAddUrlRequest.addMoKeyValue(MO_KEY);
    mediaObjectAddUrlRequest.setUrlText(URL_TEXT);

    MediaObjectAddUrlResponse mediaObjectAddUrlResponse = new
    MediaObjectAddUrlResponse();

    mediaObjectAddUrlResponse = MediaObjectOperations.addUrlMediaObject(loginEnv,
    mediaObjectAddUrlRequest);

    System.out.println("Saved URL: " + mediaObjectAddUrlResponse.getSaveURL());
    System.out.println("Sequence: " + mediaObjectAddUrlResponse.getSequence());

}
```

5.5.7 Delete

To delete a media object file, provide the media object keys and the individual sequence of the attachment you want to delete.

The code in [Example 5–17](#) shows an example of deleting a media object file. This example assumes the FileAttachment object has already been created. It uses the sequence and location values from that object to request the delete operation.

Example 5–17 Deleting a Media Object

```
public void deleteFile(FileAttachment fileAt) throws Exception
{

    MediaObjectDeleteRequest mediaObjectDelete = new
    MediaObjectDeleteRequest(loginEnv);

    //set request info
    mediaObjectDelete.setFormName(MO_APP);
    mediaObjectDelete.setVersion(MO_VERSION);
    mediaObjectDelete.setMoStructure(MO_STRUCTURE);

    //set mo key
    mediaObjectDelete.addMoKeyValue(MO_KEY);
```

```

mediaObjectDelete.setSequence(fileAt.getSequence());
mediaObjectDelete.setFileLocation(fileAt.getFileLocation());

//call delete operation to remove from E1 server and remove from local file system
MediaObjectDeleteResponse response =
MediaObjectOperations.deleteMediaObject(loginEnv, mediaObjectDelete);

System.out.println("MO Delete Response Status  " + response.getDeleteStatus());

}

```

5.6 Processing Option Service

The AIS Server provides a processing option service that enables you to retrieve the processing option fields and values for an application and version in EnterpriseOne.

The key strings can be derived by creating a type definition on the PO Data Structure in Object Management Workbench (OMW). The italicized portion of the #define below shows the key string for the example.

```
#define IDERRmnNetQuebecTaxCredit_27          27L
```

There are six supported data types. These are based on the data item used in the Processing Option Design Aid for each option.

You can get the type of the option before attempting to cast it, which is the recommended method. Or you can just cast it to the type you expect, because it is unlikely to change. The default is String, so you will always be able to get to a string version of the option value.

Type Code	Type Constant	Java Type	JDE DD Type
1	STRING_TYPE	String	String
2	CHAR_TYPE	String	Character
9	BIG_DECIMAL_TYPE	BIG Decimal	Math Numeric
11	DATE_TYPE	Date	Date
15	INTEGER_TYPE	Integer	Integer
55	CALENDAR_TYPE	Calendar	Utime

The code in [Example 5–18](#) shows an example of retrieving the processing options for P0801, version ZJDE0001. First it populates the request values and calls the poRequest service. After deserializing the response to a ProcessingOptionSet object, it uses the getOptionValue method to retrieve the value for a specific processing option based on the key string.

Example 5–18 Retrieving Processing Options with the Processing Option Service

```

public void processingOption()throws Exception{

//add capability to used (or add during login to required)
loginEnv.getUsedCapabilities().add("processingOption");
ProcessingOptionRequest poRequest = new ProcessingOptionRequest(loginEnv);
poRequest.setApplicationName("P0801"); //application
poRequest.setVersion("ZJDE0001"); //version

```

```

String response =
JDERestServiceProvider.jdeRestServiceCall(loginEnv, poRequest,
JDERestServiceProvider.POST_METHOD, JDERestServiceProvider.PO_SERVICE);

//response can be serialized to ProcessingOptionSet class
ProcessingOptionsSet poSet = loginEnv.getObjectMapper().readValue(response,
ProcessingOptionsSet.class);

//get the value for quebec tax credit using key string
BigDecimal quebecTaxCred = (BigDecimal)poSet.getOptionValue("mnNetQuebecTaxCredit_
27");

System.out.println("mnNetQuebecTaxCredit_27 value: " + quebecTaxCred);
}

```

5.7 Task Authorization Service

The task authorization service enables you to retrieve the authorized tasks in a specific EnterpriseOne task view or under a specific task within a task view.

The code in [Example 5–19](#) shows an example of retrieving the tasks under task view 18.

Example 5–19 Retrieving Tasks with the Task Authorization Service

```

public void taskAuthorization() throws Exception
{

String taskViewId = "18";
loginEnv.getRequiredCapabilities().add("taskAuthorization");
TaskAuthorizationRequest taksAuthReq = new TaskAuthorizationRequest(loginEnv);
taksAuthReq.setTaskViewId(taskViewId);

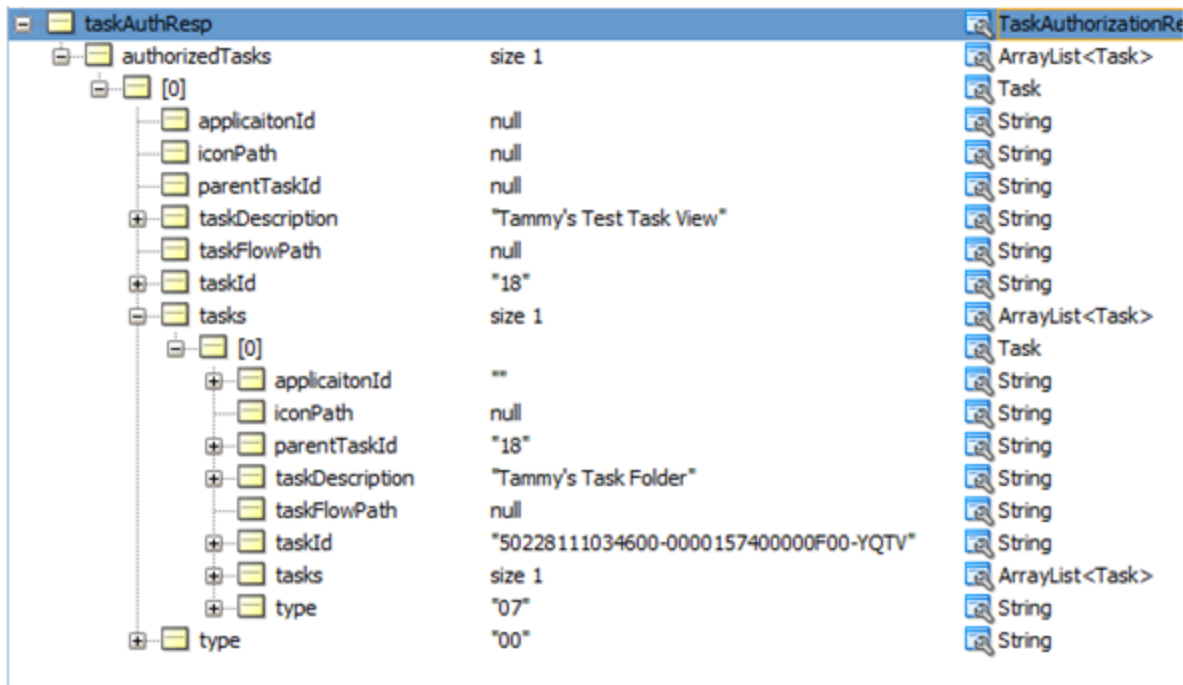
String response = JDERestServiceProvider.jdeRestServiceCall(loginEnv, taksAuthReq,
JDERestServiceProvider.POST_METHOD, JDERestServiceProvider.TASK_AUTHORIZATION);

//response can be serialized to TaskAuthorizationResponse class
TaskAuthorizationResponse taskAuthResp =
loginEnv.getObjectMapper().readValue(response, TaskAuthorizationResponse.class);
System.out.println(writer.writeValueAsString(taskAuthResp));

}

```

The TaskAuthorizationResponse object contains an array of Task type object, each with its own array of Task type objects. Although the structure supports an "infinite" number of levels, the service returns only two levels below the top task view or menu requested. You may call the service again to drill down two more levels, and so on.



To drill down another two levels, pass values in for `taskViewId`, `taskId`, and `parentTaskId` (which you received from the original request). The results will include children two levels down from the `taskViewId` passed.

See Also:

- *JD Edwards EnterpriseOne Tools Solution Explorer Guide* for information about tasks and task views in EnterpriseOne.

5.8 Logging Service

The logging service enables the AIS client to log a message in the AIS Server log. The code in [Example 5-20](#) shows an example of using the logging service:

Example 5-20 Logging Service Code

```
//do this once and it will be stored in the login environment to be used over and
over, values are optional, they will just show as null in the log if you don't set
them
loginEnv.setApplicationName("My Client Application");
loginEnv.setApplicationVersion("Client Version");

//do this every time you want to send a log to AIS
AISClientLogger.log(loginEnv, "Warn Log sent from Client to AIS
Server",AISClientLogger.WARN);
```

In this example, the log entry in the AIS Server log would be:

```
AIS LOG REQUEST: --Level 2 --Application: My Client Application --Application
Version: Client Version --User: jde --Device Name: javaclient --Log Message: Warn
Log sent from Client to AIS Server
```

5.9 Query (Release 9.1.5.2)

You can configure a form service request to send ad hoc queries to EnterpriseOne web client application forms that support the query control.

To add a query, you include a single query object in the form service request. A query object includes parameters that contain the same query criteria that you would use to set up a query in EnterpriseOne. The parameters determine:

- How the query runs.

You can configure query option parameters to load grid records in the form or clear all other fields in the form before the query runs. You can also specify whether the results of the query should match all (AND) or any (OR) of the conditions specified in the query.

- The conditions of the query.

The query object includes condition parameters that specify the control ID of the columns or fields that you want to query and an operator for filtering results that are equal to, greater than, or less than a particular value.

Important: Queries will work only if the field or columns identified in the query are part of the business view.

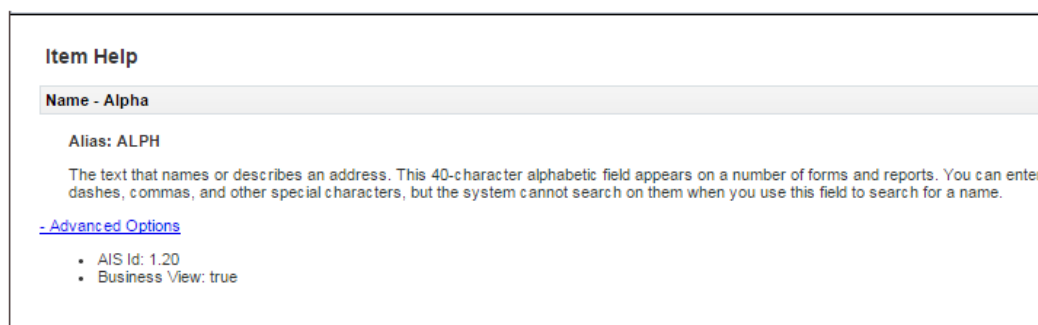
- The value used for the search criteria in the query.

The query object includes value parameters that specify the value or range of values that you want displayed in the query results.

Before you add a query object to a form service request, access the form in the EnterpriseOne web client and use the query control to gather the criteria for the query object parameters. For more information about setting up a query, see "Understanding the Query Control" in the *JD Edwards EnterpriseOne Tools Foundation Guide*.

Also, in the EnterpriseOne form, you need to identify the control ID of the field or column that you want to query, and verify that the field or column is part of the business view. To do so, click the Help button (question mark in the upper right corner of a form) and then click the Item Help option to access field-level help. With the field level help activated, you can click in a field or column to access the control ID and business view information, which is displayed under the Advanced Options section as shown in [Example 5–21](#).

Example 5–21 Example of Control ID and Business View Information Displayed under Advanced Options in the EnterpriseOne Web Client Item Help



Item Help

Name - Alpha

Alias: ALPH

The text that names or describes an address. This 40-character alphabetic field appears on a number of forms and reports. You can enter dashes, commas, and other special characters, but the system cannot search on them when you use this field to search for a name.

[- Advanced Options](#)

- AIS Id: 1.20
- Business View: true

In the Item Help, the syntax of the control ID is 1.20 with 1 representing the grid ID and 20 representing the column ID, which are separated by a dot (.). In the parameter

for the query request, the same control ID must be presented with the following syntax: 1[20]. See [Table 5–4](#) for more information.

5.9.1 Query Object Parameters

The following tables provide descriptions of the option, condition, and value parameters for a query object.

Table 5–3 Query Option Parameters

Parameter	Description	Values
autoFind	Directs the query to automatically press Find on the form to populate the grid records. You do not need to include events to press the Find button if you use autoFind.	true, false
matchType	Determines if you want the query to search for records that match all (AND) or any (OR) of the specified conditions.	MATCH_ALL, MATCH_ANY
autoClear	Determines if you want to clear all other fields on the form (for example default filter fields).	true, false

Table 5–4 Query Condition Parameters

Parameter	Description	Value
controlId	The control ID that the condition applies to. This is the field that you add to the query from the form when using the web client to create a Query. It is either a filter field or a grid column that is associated with the business view.	Example of control IDs: "28", "1[34]"
operator	The comparison operation to use with the query.	For all types, valid values are: BETWEEN, LIST, EQUAL, NOT_EQUAL, LESS, LESS_ EQUAL, GREATER, GREATER_EQUAL For strings, valid values are: STR_START_WITH, STR_ END_WITH, STR_CONTAIN, STR_BLANK, STR_NOT_ BLANK

Table 5–5 Query Value Parameters

Parameter	Description	Value
content	This is either a literal value to be used in the comparison operation, or it relates to a special value ID.	Examples of values are: "23", "Joe", "2"

Table 5–5 (Cont.) Query Value Parameters

Parameter	Description	Value
specialValueId	This is a special value, mostly for dates that might be the current day (TODAY), or calculated dates from the current day. For calculated dates, the content field is used in the calculation.	Valid values are: LITERAL, TODAY, TODAY_ PLUS_DAY, TODAY_ MINUS_DAY, TODAY_ PLUS_MONTH, TODAY_ MINUS_MONTH, TODAY_ PLUS_YEAR, TODAY_ MINUS_YEAR

Example 5–22 Query - Java API

The sample code shows a query executed in the W42101C form. This query attempts to match the following specified conditions:

- Line Number equal to 2.
- Requested Date within the last 2 years.
- Sold To between 7000 and 8000
- Company is one of the values in the list "00070,00077".
- The response will contain the JSON for the form with the matching records in the grid.

```
public void queryP42101() throws Exception
{
    loginEnv.getUsedCapabilities().add("query");
    FormRequest formRequest = new FormRequest(loginEnv);
    formRequest.setFormName("P42101_W42101C");
    formRequest.setReturnControlIDs("350|360|41[129,130,116,125]");
    formRequest.setFormServiceAction(formRequest.ACTION_READ);
    formRequest.setFindOnEntry("TRUE");
    formRequest.setMaxPageSize("20");
    Query query = new Query(loginEnv);

    //auto find
    query.setAutoFind(true);

    //match all
    query.setMatchType(Query.MATCH_ALL);

    //clear any defaulted filters
    query.setAutoClear(false);

    //line number equals 2
    NumberCondition condN = query.addNumberCondition("41[129]",
NumericOperator.EQUAL());
    condN.setValue(2);

    //Requested Date within two years from today
    DateCondition condD = query.addDateCondition("41[116]",
DateOperator.GREATER());
    condD.setSpecialDateValue(DateSpecialValue.TODAY_MINUS_YEAR(), 2);
}
```

```
//Sold To 125
BetweenCondition condST = query.addBetweenCondition("41[125]");
condST.setValues("7000", "8000");

//company in list
ListCondition list1 = query.addListCondition("360");
list1.addValue("00070");
list1.addValue("00077");

//set it in the request
formRequest.setQuery(query);

ObjectWriter writer =
loginEnv.getMapper().writerWithDefaultPrettyPrinter();
out.println(writer.writeValueAsString(formRequest));

String response = JDERestServiceProvider.jdeRestServiceCall(loginEnv,
formRequest, JDERestServiceProvider.POST_METHOD, JDERestServiceProvider.FORM_
SERVICE_URI);

}
```

5.10 Jargon Service (Release 9.1.5.3)

The jargon service enables you to retrieve data item descriptions for any EnterpriseOne data dictionary item based on the users language and jargon (system) code. This service depends on language packs applied to the EnterpriseOne system as well as data item description overrides entered with jargon codes. If there is no language pack or overrides, the base data item description is returned.

The capability name for the jargon service is "jargon". The AIS Server must have this capability to be able to process jargon service requests.

Example 5–23 Jargon Service Java API

In this example, several data items are loaded into the JargonRequest object, the service is called, and the descriptions in the response are printed out.

```
public void jargonService() throws Exception
{
    //this uses the jargon capability
    loginEnv.getRequiredCapabilities().add("jargon");

    //create the request object, seeding it with a default system code of 01
    JargonRequest jargonRequest = new JargonRequest(loginEnv, "01"); // with
    default system code

    //fill the list in the request with data items
    jargonRequest.addDataItem("AN8"); //uses default system code
    jargonRequest.addDataItem("MCU","04"); //use system code 04 for this one
    jargonRequest.addDataItem("PAN8");
    jargonRequest.addDataItem("ITM","55");

    //call the jargon service
    String response = JDERestServiceProvider.jdeRestServiceCall(loginEnv,
```

```

jargonRequest, JDERestServiceProvider.POST_METHOD, JDERestServiceProvider.JARGON_
SERVICE);

//response can be serialized to JargonResponse class
JargonResponse jargonResponse =
loginEnv.getMapper().readValue(response, JargonResponse.class);

//print the response
if(jargonResponse != null)
{
    if(jargonResponse.getRequestedItems() != null &&
jargonResponse.getRequestedItems().size() >0 )
    {
        for(JargonResponseItem item: jargonResponse.getRequestedItems())
        {
            System.out.println("Item " + item.getSzDict() + " " +
item.getRowDescription());
        }
    }
}
}

```

5.11 Data Service (API Release 1.1.0 and EnterpriseOne Tools Release 9.1.5.5)

Starting with EnterpriseOne Tools 9.1.5.5, the AIS Server provides an endpoint called "dataservice" for data query or count requests over tables or business views.

Data service calls are made using the DataRequest object. If you use the data service, you must include the "dataservice" capability in the required or used capabilities list.

Table 5–6 Data Service Request Required Parameters

Parameter	Description	Values
targetName	The name of the table or view to count or query.	Example values: F0101 or V4210A
targetType	The object type to count or query: table or business view.	DataRequest.TARGET_TABLE DataRequest.TARGET_VIEW
dataServiceType	The type of operation to be performed: count or query (represented by the value BROWSE).	DataRequest.TYPE_COUNT DataRequest.TYPE_BROWSE

Table 5–7 Data Service Request Optional Parameters

Parameter	Description	Values
findOnEntry	This parameter determines if the service performs an automatic find.	FormRequest.TRUE FormRequest.FALSE
returnControlIDs	The columns of the table or business view to be returned in a query response (pipe delimited).	Example values: F0101.AN8 F0101.PA8 F0101.ALP H
query	A query object which is built using column IDs for the control IDs.	

Example 5-24 Data Service Java API

This example shows both a browse and a count of the F0101 table, including a query. The response from the browse is assembled into a class (not included) that was generated with the class generator for F0101 data service. The count response is assembled into a simple HashMap and printed.

```
//add to the used capabilities
loginEnv.getUsedCapabilities().add(AISClientCapability.DATA_SERVICE);

//create a new DataRegeust
DataRequest f0101 = new DataRequest(loginEnv);

//Set table information, this is a browse of F0101
f0101.setDataServiceType(DataRequest.TYPE_BROWSE);
f0101.setTargetName("F0101");
f0101.setTargetType(DataRequest.TARGET_TABLE);
f0101.setFindOnEntry(FormRequest.TRUE);

//set return control ids, only these three columns will be in the response
f0101.setReturnControlIDs("F0101.AN8|F0101.ALPH|F0101.AT1");

//only return the first 10 records
f0101.setMaxPageSize("10");

//create a new query, for address numbers greater than 7000
Query greaterQ = new Query(loginEnv);
greaterQ.setAutoFind(true);
greaterQ.setMatchType(Query.MATCH_ALL);
greaterQ.addStringCondition("F0101.AN8", StringOperator.GREATER(), "7000");
f0101.setQuery(greaterQ);

//execute the data request
String response = JDERestServiceProvider.jdeRestServiceCall(loginEnv, f0101,
JDERestServiceProvider.POST_METHOD, JDERestServiceProvider.DATA_SERVICE_URI);

//marshal the response to a formparent class generated by the class generator
DATABROWSE_F0101_FormParent f010Data
=loginEnv.getMapper().readValue(response,DATABROWSE_F0101_FormParent.class);

//modify the type to count, and get a count response for the same query
f0101.setDataServiceType(DataRequest.TYPE_COUNT);
String countresponse = JDERestServiceProvider.jdeRestServiceCall(loginEnv, f0101,
JDERestServiceProvider.POST_METHOD, JDERestServiceProvider.DATA_SERVICE_URI);

//loop through the records in the response printing out the values
ArrayList<DATABROWSE_F0101_GridRow> rowSet = f010Data.getFs_DATABROWSE_
F0101().getData().getGridData().getRowset();
if (rowSet.size() > 0)
{
    for (DATABROWSE_F0101_GridRow row: rowSet)
    {
        System.out.println("Name: " + row.getSAlphaName_54().getValue());
        System.out.println("Number: " + row.getMnAddressNumber_51().getValue());
        System.out.println("Search Type: " + row.getSSchTyp_59().getValue());

        System.out.println(" ");
    }
}
```

```

else
{
    fail("No Records in Reponse");
}

//marshal and print out the count response
HashMap countRespMap = loginEnv.getObjectMapper().readValue(countresponse,
HashMap.class);
HashMap countMap = (HashMap)countRespMap.get("ds_F0101");
System.out.println(countMap.get("count"));

```

5.12 Orchestration Support (API Release 1.1.0 and EnterpriseOne Tools Release 9.1.5.5)

Starting with EnterpriseOne Tools 9.1.5.5, the AIS Server supports form service request calls from orchestrations. The AIS Server must be configured to work with orchestrations. See "Prerequisites" in the *JD Edwards EnterpriseOne Tools Internet of Things Orchestrator Guide* for more information.

This section describes how to invoke the orchestration using the AIS Client API.

Orchestrator requests are stateless. The entire orchestration is executed in a single call and returns the results of the orchestration.

Example 5–25 Using JDE Standard Input Format for an Orchestration

This example uses the JDE Standard input format. The orchestration must be configured to accept this input format.

```

OrchestrationRequest req = new OrchestrationRequest(AIS_SERVER ,USER_NAME,
PASSWORD,DEVICE_NAME);

req.setOrchestration("GetAddressBook_Simple");
req.getInputs().add(new OrchestrationInputValue("AddressBookNumber", "7500"));
req.getInputs().add(new OrchestrationInputValue("SearchType", "E"));

try{

    String output = req.executeOrchestrationRequest();

    //consume output, you can deserialize it to a class generated by the AIS Class
    Generator
}
catch(Exception e)
{
    //handle exceptions
}

```

Example 5–26 Using Generic Input Format for an Orchestration

This example uses the Generic input format. The orchestration must be configured to accept this input format.

```

OrchestrationRequest req = new OrchestrationRequest(AIS_SERVER ,USER_NAME,
PASSWORD,DEVICE_NAME);

orchRequest.setOrchestration("AddCBM");

```

```
//populate values to send from this instance, simple name value pairs hash map
HashMap<String,String> vals = new HashMap<String,String>();
vals.setSerialNumber("02a0bd30-d883-11e4-b9d6-1681e6b88ec1");
Date jDate = new Date();
vals.put("date",String.valueOf(jDate.getTime()));
SimpleDateFormat sdf = new SimpleDateFormat("hh:mm:ss");
vals.put("time",sdf.format(jDate));
vals.put("temperature","201");
vals.put("description","Temp 201");

try
{

    String response = orchRequest.executeOrchestrationRequest(values);

    //consume response, you can deserialize it to a class generated by the AIS Class
    Generator
}
catch(Exception e)
{
    //handle exceptions
}
```

Glossary

AIS Server

A REST services server that when configured with an EnterpriseOne HTML Server, enables access to EnterpriseOne forms and data.

AIS Server capability

A behavior of the AIS Server that an AIS client can use to perform a particular EnterpriseOne task, such as update a grid record or fetch a processing option.

AIS client

An application that uses the AIS Server to communicate with EnterpriseOne.

AIS Server endpoint

An endpoint on the AIS Server that provides a service for the AIS client. An AIS client can access an AIS Server endpoint through a URL. In turn, the endpoint performs a particular service for the AIS client in EnterpriseOne.

AIS service

A service in an AIS Server endpoint. An AIS service interacts with EnterpriseOne based on input from an AIS client and provides a response in JSON format.

form service request

An AIS Server call that retrieves data from a form in EnterpriseOne. Form service requests, formatted as REST service calls that use POST, contain form service events or commands that invoke actions on an EnterpriseOne form.

instantiate

A Java term meaning "to create." When a class is instantiated, a new instance is created.

JDeveloper Project

An artifact that JDeveloper uses to categorize and compile source files.

JSON (JavaScript Object Notation)

A light-weight format used for the interchange of data between the AIS Server and EnterpriseOne.

processing option

A data structure that enables users to supply parameters that regulate the running of a batch program or report. For example, you can use processing options to specify default values for certain fields, to determine how information appears or is printed,

to specify date ranges, to supply runtime values that regulate program execution, and so on.

QBE

An abbreviation for query by example. In JD Edwards EnterpriseOne, the QBE line is the top line on a detail area that is used for filtering data.

serialize

The process of converting an object or data into a format for storage or transmission across a network connection link with the ability to reconstruct the original data or objects when needed.