JD Edwards EnterpriseOne Tools

UX One Deployment and Development Guide

9.2

JD Edwards EnterpriseOne Tools
UX One Deployment and Development Guide

9.2

Part Number: E79230-13

Contents

Preface

Welcome to the JD Edwards EnterpriseOne documentation.

Documentation Accessibility

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

Access to Oracle Support

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

Related Information

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

`http://learnjde.com`

Conventions

The following text conventions are used in this document:

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

1 Understanding UX One Pages

Understanding UX One Pages

JD Edwards UX One pages implement the Alert, Analyze, and Act model, bringing together the pieces of information that you need in specific job roles. These pages alert you to conditions you need to be aware of, enable you to analyze data to determine appropriate actions, and then provide a direct path to the application and form that you need to use to take those actions.

When you log in to JD Edwards EnterpriseOne using a UX One role, you see the UX One pages associated with that role. A UX One page consists of one or more of the following components:

- Watchlist Pane
- Springboard Pane
- Designer Pane
- Charts

For more information on UX One Pages, see *"Understanding UX One Role Navigation" in the JD Edwards EnterpriseOne Applications UX One Roles User Guide guide.*

You can modify the UX One pages or create your own pages that are based on the Alert, Analyze, Act methodology. You create them using Composed Pages and each pane can be made up of the following components:

- The Alert pane usually contains Watchlists that alert you to information or situations that require immediate attention.

  For more information on creating Watchlists, see *"Creating One View Watchlists" in the JD Edwards EnterpriseOne Applications One View Watchlists Implemenation Guide .*

- The Analyze pane contains graphic representations of your data, whether in bar chart, pie, or many other formats. These analytics enable you to quickly review data at a glance. These components are generally ADF applications or JET components.

  For more information on JET components, see *Building a JET Application Using the Template*.

- The Act pane is a springboard that contains icons which link you to the applications that you need to act upon.

  For more information on Springboard Panes, see "*Adding a Springboard Pane to a Composed EnterpriseOne Page" in the JD Edwards EnterpriseOne Tools Using and Approving User Defined Objects Guide .*

For more information on Composed EnterpriseOne Pages in general, see *"Composed EnterpriseOne Pages" in the JD Edwards EnterpriseOne Tools Using and Approving User Defined Objects Guide .*

2 Installing and Configuring UX One

Prerequisites

Before installing and configuring UX One, perform these prerequisites.

- You must be running a minimum of EnterpriseOne Tools Release 9.2.1 and apply any required ESUs for EnterpriseOne 9.2 as described in My Oracle Support.
- Deploy an Application Interface Services (AIS) Server.

  Make sure there is an AIS Server configured with your EnterpriseOne system. Certain components in UX One pages require an EnterpriseOne Application Interface Services (AIS) Server to interact with EnterpriseOne forms and data.

  You should set up an additional HTML Server instance for processing AIS Server requests only. This is recommended so that the performance of the EnterpriseOne HTML Server used by EnterpriseOne web client users is not impacted by AIS Server requests. For more information, see *"Additional EnterpriseOne HTML Server Instance for Processing AIS Requests" in the JD Edwards EnterpriseOne Application Interface Services Server Reference Guide*
- You must have a JD Edwards EnterpriseOne-supported Internet Browser.
- (Optional) Set up Oracle WebLogic Server with ADF runtime.

  Some UX One pages contain modernized user interfaces such as calendars and organizational charts. These are sometimes referred to as ADF applications. If you are going to use any of the modernized user interfaces, you must set up an Oracle WebLogic Server with ADF runtime (otherwise referred to as the ADF Server) to run those components.

  For a list of the applications that use ADF, search LearnJDE.com for "Information Center: Installation of ADF components."

  For more information on ADF runtime, see *"Setting Up Oracle WebLogic Server with ADF Runtime" in the JD Edwards EnterpriseOne Tools Deploying and Developing Oracle Application Development Framework (ADF) Applications for EnterpriseOne Guide.*

Installing UX One Content

The UX One updates consist of ESUs, ADF components, and UDO components. You need to install all of these UX One components for UX One to function properly.

You can find these components for UX One on the Oracle Software Delivery Cloud (OSDC) or by using the Change Assistant query "UX One - All" located under Search for Packages/JD Edwards/Electronic Software Updates/9.2.

These updates include:

- All ESUs needed
- ADF components

Access information for using specific ADF components, such as location maps and organization charts, at the bottom of this page:

`https://support.oracle.com/epmos/faces/DocContentDisplay?_afrLoop=564276148508333&id=1990959.1`

- UDO components

Data Packs for UX One

UX One landing page components are configured to work with certain roles and demo data. The "UX One" query in Change Assistant includes two data packs:

- **Data Pack 1: UX One Security Roles.** This data pack contains 50+ pre-configured user roles and the associated UDO view security records. The role names end with "JDE" to distinguish them from any roles that you have created. If you install this data pack, UDO view security is automatically enabled for the UX One UDOs. When you assign users to the applicable roles, the UX One landing pages and associated UDOs become available for them to use. If you do not install this data pack, you will need to set up UDO view security for each of the UX One composed pages and associated UDOs.
- **Data Pack 2: Demo Data.** This data pack contains demo data and should not be installed into any environment that contains your test or business data because it will overwrite that data. The purpose of the demo data is to provide data that works with the delivered queries to show the analytic components on the UX One landing pages.

For more information on the data packs, please refer to the overview document, included with the data pack, that describes how to deploy these data packs to your environment.

UX One Developer Environment and Examples

If you would like to customize the UX One content, you can download the "E1JETDeveloper_EnvironmentAndExamples" UDO zip file. You can find it by using the "UX One - All" query in Change Assistant.

Importing UX One Content

The following instructions, beginning with step 3, also apply to importing a zip file that you have created or customized.

To import UX One content:

1. In Change Assistant, open Search for Packages, JD Edwards, Electronic Software Updates, and then 9.2, in the tree structure. Select the "UX One - All" query and select Search on the Advanced Search tab to find the UX One updates described above.
2. Use Change Assistant to download and deploy the files.

   For more information, see *"Using Change Assistant" in the JD Edwards EnterpriseOne Tools Software Updates Guide*
3. In JD Edwards EnterpriseOne, navigate to Web OMW (P98220W).
4. Select Import from the Form menu.
5. Find and select the UX One zip file that you downloaded.
6. Click Load.

7. When the zip file is imported, it appears as a project with status 11, New Pending Project Review. At this point the imported UX One contents are at a status of 07, Pending Promote.
8. You must use the Approve/Share option from P98220U or P98220W in order to make the UDOs available to users.

   The Approve/Share action extracts the data and artifacts from the zip repository and loads the runtime tables and target locations.

   In addition, the project can be advanced with Transfer Activity Rules to transfer the UDO to other path codes. An Approve/Share must be performed by signing into the transfer environment.

For more information on transferring UDOs, see *JD Edwards EnterpriseOne Tools Object Management Workbench for the Web Guide* .

For more information on approving UDOs, see *JD Edwards EnterpriseOne Tools Using and Approving User Defined Objects Guide* .

View Dependencies (Release 9.2.3)

The UX One updates consist of many UDO components. View Dependencies enables you to identify all the dependent objects that are required to set up and implement a UX One role.

You can view detailed information of the object dependencies for Watchlist, Composite Application Framework (CafeOne), and Composite Page UDOs. You can also export the object dependency information to Microsoft Excel, a Comma Separated Values file, or to clipboard.

For more information, see "View Dependencies at Project Level" in the *JD Edwards EnterpriseOne Tools Using and Approving User Defined Objects Guide* .

Menu Indexing for Springboard

A Springboard Pane consists of a collection of tasks available to a role. The tasks launched from the Springboard Pane can be EnterpriseOne applications, batch programs, ADF applications, and One View Reports. When you launch a task item from the Springboard Pane, the system opens the associated application in a separate frame and launches a springboard at the top of the form. The Springboard Pane uses menu indexing to determine the correct set of task items to be displayed for a role. Menu indexing is supported by the following two tables:

- Item Detail (F90012): This table is the key to menu indexing. For each logged in role, the system traverses all the menu tree nodes, filters all the tasks assigned a certain role (Menu Filtering), and updates the Item Detail table (F90012).
- Task Index Status (F90013): The Task Index Status table (F90013) stores the status of the task build requests and triggers record generation in the Item Detail table (F90012).

Menu indexing occurs in two different ways:

- When a user or role logs into JD Edwards EnterpriseOne, a menu index build occurs automatically.
- You can use the Work with Task Index Builds application (P90013) to interact with the Task Index Status table to build the task index for roles or tasks in the Item Detail table.

It is recommended that you manually build the task index after installing the UX One content. For more information, see *"Working with Task Index Builds" in the JD Edwards EnterpriseOne Tools Foundation Guide* .

Setting Up User Roles

UX One is delivered with pre-configured user roles and associated security records. If you would like to run the UX One components with the delivered demo data, you need to perform the steps in this section to set up these roles.

Note: This section only applies if you have installed the appropriate data packs.

User Roles Overview

User roles and associated security settings are delivered with EnterpriseOne UX One. You should review these roles to see if they meet your organization's needs, update them as appropriate, and assign environments and individual users to the roles.

To review the user roles, go to P0092 to access the User Profiles application.

On the Work With User/Role Profiles form, select the Roles Only option, click Find, select a role in the grid, and then click Select.

For more information on user roles, see *"Setting Up Roles" in the JD Edwards EnterpriseOne Tools Security Administration Guide .*

Adding an Environment to a Role

Use the Work With User/Role Profiles form to assign one or more environments to a role. When a user signs in to JD Edwards EnterpriseOne, the Environment Chooser and Role Chooser present each user with a list of valid roles and environments.

For more information on assigning environments, see *"Adding an Environment to a Role" in the JD Edwards EnterpriseOne Tools Security Administration Guide .*

Creating Role Relationships

In order to use the roles delivered with EnterpriseOne UX One, you must associate users with them. You may also want to review and update the security associated with the role to provide the appropriate level of access to EnterpriseOne functions for your organization. You can assign more than one user to a role, or you can assign more than one role to a user. To establish a role relationship, you use the Role Relationships application (P95921), which enables you to add, remove, or revise a role relationship for a user.

For more detailed instructions, see *"Setting Up a Role Relationship" in the JD Edwards EnterpriseOne Tools Security Administration Guide .*

Note: All UX One roles are delivered with pre-defined Role Sequence numbers. It is recommended to not include these roles in *All option when setting up the role relationship. Selecting Include in *ALL can cause issues in determining the role hierarchy, if there is an issue with the collision of sequence numbers with other roles in the system. However, if Include in *ALL must be selected and a collision occurs, you have the option to manually override the role sequence numbers.

Copying User Roles

You can copy the role relationship records of one user to another from Role Relationships (P95921). You can either copy and add the records, which means that EnterpriseOne adds the copied records to the user's existing records; or you can copy and replace the records, which means that the copied records replace the user's existing records.

If you would like to copy any of the delivered roles, see *"Copying User Roles" in the JD Edwards EnterpriseOne Tools Security Administration Guide .*

User Defined Object (UDO) Security

UX One roles use a variety of UDOs, which enable end users to work smarter and more efficiently. These UDOs include Composed EnterpriseOne Pages, Springboards, Watchlists, queries, and One View reports, among others.

UX One pages contain a Springboard Pane that displays only those task items for which a role has permissions. Menu indexing determines the correct set of task items to be displayed for a role in the Springboard Pane. An administrator must rebuild the task index anytime the roles or tasks change.

For more information, see *"Working with Task Index Builds" in the JD Edwards EnterpriseOne Tools Foundation Guide .*

UDO Contents

As an administrator, you can view all of the UDO contents delivered with UX One in the User Defined Object Administration application (P98220U).

To view the UDOs, see *"Previewing UDOs" in the JD Edwards EnterpriseOne Tools Using and Approving User Defined Objects Guide .*

UDO Security

EnterpriseOne UX One is delivered with UDO security specific to the delivered roles. You can review these security records in P00950, where you can look up a particular role and see everything assigned to it. You can also copy the UDO view security from a delivered role to another role, or set up view security by specific user instead of role.

For more information, see *"Managing Security for User Defined Objects" in the JD Edwards EnterpriseOne Tools Security Administration Guide .*

If you want a particular component to show up in the tab strip in EnterpriseOne as its own individual tab, you can set up view security at the component level.

Enabling AIS Cache Responses (Release 9.2.1.4)

You can enable the AIS Server to cache responses of service requests for the analytic components on a UX One page. This enables the system to retrieve the data for the charts from the cache, thereby improving the system's performance by loading the chart data quickly.

For more information, see *"Enabling Service Request Response Caching" in the JD Edwards EnterpriseOne Application Interface Services Server Reference Guide .*

3 Setup for EnterpriseOne Page JET Development

Development Prerequisites

Complete the following prerequisites:

- An AIS Server configured with your EnterpriseOne system

  Certain components in UX One pages require an EnterpriseOne Application Interface Services (AIS) Server to interact with EnterpriseOne forms and data.

  You should set up an additional HTML Server instance for processing AIS Server requests only. This is recommended so that the performance of the EnterpriseOne HTML Server used by EnterpriseOne web client users is not impacted by AIS Server requests. For more information, see *"Additional EnterpriseOne HTML Server Instance for Processing AIS Requests" in the JD Edwards EnterpriseOne Application Interface Services Server Reference Guide .*
- JDeveloper 12.2.1 if you want to use the UX One examples out of the box

  See the "Oracle JDeveloper Get Started" documentation for installation instructions:

  `http://docs.oracle.com/middleware/12211/jdev/index.html`
- UX One EnterpriseOne UDOs

  If you have not already done so, download the latest EnterpriseOne UDOs for UX One content from the Update Center.

  See *Installing UX One Content*.
- EnterpriseOne JET Developer Environment and Examples UDOs

  If you have not already done so, download the UX One environment and examples zip file.

  See *Installing UX One Content*.

  Along with the UX One examples, this zip file includes JET libraries, helper JavaScript libraries, and a local development environment for creating your own pages.
- A JD Edwards EnterpriseOne-supported Internet Browser
- Knowledge of Oracle JET

  For more information on Oracle JET, see `http://www.oracle.com/webfolder/technetwork/jet/index.html`

Local Testing Environment Setup

UX One is delivered with a local environment that you can use for testing your JET applications. This enables you to test your JET applications before importing them into EnterpriseOne as Classic Pages. You can test charts, other analytics, and AIS calls locally, but not form interconnects.

To set up a local test environment, you need to update the e1pagehelper.js configuration file and register one or more browsers within JDeveloper.

e1pagehelper.js Configuration

Update the following items at the bottom of the e1pagehelper.js file, within the LocalDevConfig variable:

- this.AIS_HOST

  Update the AIS Server that you want to use.
- this.AIS_PORT

  Enter the port of the AIS Server.
- this.USER_NAME

  Enter a valid user for the JD Edwards EnterpriseOne environment.

  In the getUserInfo function, there is a userPreference variable that is hard coded for local testing. The elements in the variable should be configured to match the user preferences of the user set up in this.USER_NAME.

  Specifically, simpleDateFormat must be checked. JET applications using date filters will not work if the formatted dates in the JET application are not formatted the same as seen in JD Edwards EnterpriseOne for the user.

  Example: getUser Info ()

```
function getUserInfo()
{
   //alert("Not Available in Local Development Environment");
   var userPreference =
                {
                    addressNumber : "6001",
                    country : " ",
                    dateFormat : "MDE",
                    dateSeperator : "/",
                    decimalFormat : ".",
                    dstRule : "",
                    env : "JDV920",
                    fullName : "Allen, Ray",
                    lang : "en",
                    locale : "en",
                    role : "*All",
                    servicePack : "9.2.1",
                    simpleDateFormat : "MM/dd/yyyy",
                    timeFormat : "24",
                    userId : "JDE",
                    userTimeZone : "26",
                    e1Lang: "   "
                };
   return userPreference;
}
```

- this.PASSWORD

  Enter a valid password for the JD Edwards EnterpriseOne environment.

Register Browser in JDeveloper

In order to test the JET application, you need to run home.html in a browser. Within JDeveloper, you can do that by registering one or more browsers as external tools.

After both the browser and e1pagehelper.js file are set up, you can perform a test by running the six existing samples.

4 UX One Development with Oracle JET

UX One JET Examples

The JET Development download includes the files necessary to implement the template, along with six examples. With the template, you get a JET page with a single chart that contains the following features:

- Menu with persistence
- Title Configuration
- Query Builder

This download also enables you to build a JET application that does not follow the template by using the functions in e1pagehelper.js to make AIS calls to get data.

You can use these examples, included in the download, as a guide to implement your own JET applications:

Downloadable Examples

Chart	Request Type	Chart Type	Included Features
P13230_TotalEquipmentDaysbyLocation	Form Service Request	Bar	Group By, Looping through data array to sort
P90CG530_CaseDistribution	Data Aggregation Service Request	Pie	Complex Query
P13570_AveragePMFulfillment	Form Service Request	Bar	Select values/refresh chart - do not persist data, Data Sorting functions Navigation to base application, context menu navigation (Release 9.2.2)
P54HS260_NumberofReportableCases	Form Service Request	Bar	Select values/Menu Drawer - persist data Navigation to base application (Release 9.2.2)
P13560_CompletionTimelinessGroupby8Options	Form Service Request	Bar	Tool Tip Navigation to base application, context menu navigation (Release 9.2.2)
P40G210_ReceiptsbySupplier	Data Browser Service Request	Pie	Navigation to base application (Release 9.2.2)

Files Included with the Development Environment

To expedite the development of JET applications for EnterpriseOne, Oracle provides additional tools referred to as JDE AIS and JET Helpers.

Note: Do not modify any of these helper files.

These helpers include:

- e1pagehelper.js
  This file allows communication with the AIS Server to get and update data in JD Edwards EnterpriseOne.
- jetconfig.js
  This configuration file is only updated if you want to use JET components that are currently not included with the template.
- jetUtilities.js
  This file implements all of the features that you get when using the template.
- jetoverride.css
  This file defines much of the styling that is used by applications using the template.
- jetQueryBuilder.js
  This file contains the code used by the Query Builder feature of the template.
- jetNavigation.js (Release 9.2.2)
  This file contains the code used by the Navigation and Context Menu Navigation features of the template.

JET Application Files

JET applications that were created using the template have three files:

- E1AISCalls.js
  Update this file to configure the template (Menu, Title, Query Builder), as well as to define the AIS calls to retrieve and process the data needed to render the chart.
- home.html
  This files defines the layout of the page and should not be changed if using the template.
- main.js
  This file is the main JavaScript file used by home.html and should not be changed if using the template.

Building a JET Application Using the Template

1. Copy the three files from an example and place in a new folder with a different name.

2. These files are E1AISCalls.js, home.html, and main.js. If following the template, the only file that needs to be modified is the E1AISCalls.js.
3. In order to create a JET application using the template, you need to assign values to several variables in the E1AISCalls.js.

   Update these variables:

Variable	Value
self.componentId	Less than or equal to 20 characters, unique identifier used for storing persistent values.
self.objectName	Name of the EnterpriseOne application, business view, or table, off which the JET application is based (for example, P42101). The list of queries and Query Builder is also based on this object.
self.formName	The form (Wxxxxx) for the application that the JET application is based on. This is blank if using a table or business view.
self.labelObjectName	This is used to get translated labels for the JET application.
self.isDataRequest	This value is true when the object name from above is a table or business view. This can also be set to true if fetching the related business view from the application, which will improve performance. This is set to false if you actually want to run the application and get the data from the grid. For information on performance considerations, see *Using isDataRequest for AIS Call*.
self.hasVersions	Set to true if you are using an application that has processing options.
self.stackValue	This is set to "On" if you are defining a bar chart and you want the bars to be stacked.
self.orientationValue	Used for both line and bar charts to determine whether the orientation is horizontal or vertical.
self.chartType	This value can be Line, Bar, or Pie.

   For information on title configuration, see *JET Titles*.

   For information on menu configuration, see *Menu Functions*.

4. Rework the AIS call.

   For information on how to do this, see *AIS Call APIs*.

   To consume the response of the AIS call, you must structure the data as needed by the chart you are building. Use the examples for guidance. Based on the response, you can utilize Fetch Descriptions and Fetch UDC Descriptions to get other information (labels) to display in the charts. For more information, see *Fetch Descriptions* and *Fetch UDC Descriptions*.

General Process Flow for Creating a UX One Page Using a JET Component

The following list describes the high-level tasks involved in creating a UX One page using a JET component. In most cases, it also provides links to sections in this guide or other guides that discuss each task in more detail.

1. Complete the prerequisites as described in *Setup for EnterpriseOne Page JET Development* in this guide.
2. Create your JET application. For more information, see *Building a JET Application Using the Template*.
3. Test your JET application in your local environment. For more information, see *Local Testing Environment Setup*.
4. Zip up the files within the E1 Page folder in your local development environment.
5. Log into JD Edwards EnterpriseOne and access Classic Pages
6. Import your zip file as a Classic Page.

   For more information, see *"Creating EnterpriseOne (Classic) Pages in EnterpriseOne" in the JD Edwards EnterpriseOne Tools Foundation Guide .*
7. Go through the UDO approval process and share the classic page.

   For more information, see the *JD Edwards EnterpriseOne Tools Using and Approving User Defined Objects Guide .*
8. At this point, depending on how you have security set up, your classic page shows up in the tab strip at the top of the page. Instead, you should associate it with an external form so that it functions more like a standard EnterpriseOne application.
9. If you want to access it directly, from a Composed Page, or from CafeOne, you need to create an external form with which to associate it.

   For more information, see *"Understanding External Forms" in the JD Edwards EnterpriseOne Tools Form Design Aid Guide .*

5 EnterpriseOne JET Template Resources

AIS Call APIs

To make AIS calls from JavaScript, use the available APIs in the e1pagehelper.js. See the tables below for more information about the available APIs.

The following example shows a data service request with aggregation. This example demonstrates the callAISService API, using the DATA_SERVICE constant.

The first parameter is a JSON object defining the input message sent to AIS.

The second parameter is the DATA_SERVICE constant to indicate the request is a data request.

The final parameter is a JavaScript callback function accepting the response of the AIS call as input. This function will execute once a response is returned from the AIS call. More examples of callAISService can be found in the example JET applications included with the JET Development Template.

Example - Data Service Request Using Aggregation from P90CG530_CaseDistribution Application

```
var groupingCol = null;

/* group by status or assigned to */
if (self.savedGroupByValue() == 'assignee')
{
    groupingCol = "F1755.AN8";
}
else if (self.savedGroupByValue() == 'status')
{
    groupingCol = "F1755.CLST";
}

var savedQuery = self.savedQueryValue;
if(savedQuery == 'AllRecords')
{
    savedQuery = '';
}

var input =
{
    aliasNaming : true,
    outputType : "GRID_DATA",
    findOnEntry : "TRUE",
    maxPageSize : LocalJetVariables.maxPageSize,
    query :
    {
        autoFind : true, autoClear : true,
        complexQuery : [{query :
                {condition : [{value : [{content : '2', specialValueId : "LITERAL"}],
                              controlId : "F1755.STAW", operator : "EQUAL"}],
                    matchType : "MATCH_ALL"},
                    andOr : "AND"}]
    },
    queryObjectName : savedQuery,
    targetName : "V17550",
    targetType : "view",
    dataServiceType : "AGGREGATION",
    aggregation :
    {
        aggregations : [{column : groupingCol, aggregation : "COUNT"}],
        groupBy : [{column : groupingCol}],
        orderBy : [{column : groupingCol,
        direction : "ASC"}]
    }
};

if(self.hasVersions){
    input.version =  self.savedVersionValue;
}

/* add adhoc query if specified to complex query defined above */
if(self.adhocQuery)
{
     var adhocQ = {query: {condition: self.adhocQuery.condition,
                           matchType: self.adhocQuery.matchType},
                   andOr : "AND"};
     input.query.complexQuery.push(adhocQ);
}

self.currentGroupByValue = self.savedGroupByValue();

/* call AIS data service based on the above defined input */
callAISService(input, DATA_SERVICE, function (response)
{
    // consume response here
}
```

For more information, see *REST API for JD Edwards EnterpriseOne AIS Server.*

E1 Page Helper AIS Services

Service Name Constant	**REST API**	**In E1Pagehelper Release**
FORM_SERVICE	*Form Service*	9.2.0.2
DATA_SERVICE	*Data Service*	9.2.0.2
BATCH_FORM_SERVICE	*Batch Form Service*	9.2.0.2
APP_STACK_SERVICE	*Application Stack Service*	9.2.0.2
PO_SERVICE	*Processing Option Service*	9.2.0.2
LOG_SERVICE	*Logging Service*	9.2.0.2
JARGON_SERVICE	*Logging Service*	9.2.0.2
PREFERENCE_SERVICE	*Preference Service*	9.2.0.2
WATCHLIST_SERVICE	*Watchlist Service*	9.2.1.0
UDO_GETALL_SERVICE	*UDO Service*	9.2.1.0

UDO_GETKEY_SERVICE	*UDO Service*	9.2.1.0
MEDIA_OBJECT_TEXT	*Media Object Service* – Get Text	9.2.1.0
MEDIA_OBJECT_UPDATETEXT	*Media Object Service* – Update Text	9.2.2.6
MEDIA_OBJECT_ADDTEXT	*Media Object Service* – Add Text	9.2.3.0
MEDIA_OBJECT_DELETE	*Media Object Service* - Delete	9.2.3.0
MEDIA_OBJECT_LIST	*Media Object Service* - List	9.2.3.0
SETTINGS	*Default Config Service*	9.2.1.4
CAPABILITIES	*Default Config Service*	9.2.1.4
DISCOVER_SERVICE	*Orchestration Discovery Service*	9.2.2.1
DISCOVER_NOTIFICATIONS	*Notification Service* - List	9.2.2.1
REPORT_EXECUTE	*Report Service* - Execute	9.2.2.6
REPORT_DISCOVER	*Report Service* - Discover	9.2.2.6
REPORT_STATUS	*Report Service* – Get Status	9.2.2.6
MEDIA_OBJECT_UPLOAD	*Media Object Service* - Upload	9.2.7.0
MEDIA_OBJECT_DOWNLOAD	*Media Object Service* - Download	9.2.7.0
EXTERNAL_REST	*External Service* - REST	9.2.7.0
EXTERNAL_REST_REPORT	*External Service* – REST Report	9.2.7.0
EXTERNAL_FTP_REPORT	*External Service* – FTP Report	9.2.7.0
GET_OBJECT_LIST	*Utilities Service*	9.2.7.0
OPEN_API_CATALOG	*Swagger 2.0*	9.2.7.0
OPEN_API_CATALOG3	*OpenAPI 3.0*	9.2.7.0
NOTIFICATION	*Notification Service* - Execute	9.2.7.0
DATA_SERVICE_NEXT	*Data Service* – Next Record	9.2.7.0
APP_STACK_NEXT	*Application Stack Service* – Next Record	9.2.7.0
EXTERNAL_OPEN_API	*External Service* – Open API	9.2.7.0
BUSINESS_FUNCTION_SERVICE	*Business Function Service* - Execute	9.2.7.0
DISCOVER_BUSINESS_FUNCTION	*Business Function Service* - Discover	9.2.7.0
MEDIA_OBJECT_ADD_URL	*Media Object Service* – Add URL	9.2.7.0
MESSAGE_SERVICE	*Message Service*	9.2.7.0
NOTIFICATION_SUBSCRIPTIONS	*Message Service* – Get Subscriptions	9.2.7.0
SCHEDULER	*Scheduler Service*	9.2.7.0
TASK_AUTHORIZATION	*Task Authorization Service*	9.2.7.0

E1 Page Helper AIS APIs

API Method	Description	Inputs	Outputs
callAISService	Call a JSON based AIS Service defined with one of the constants.	JSON Object, Service Constant, Callback	Callback pas be called wit Object respo AIS REST AP
callAISServiceV2	Call the V2 endpoint of a JSON based AIS Service defined with one of the constants.	JSON Object, Service Constant, Callback	Callback pas be called wit Object respo AIS REST AP
callAISServiceV3	Call the V3 endpoint of a JSON based AIS Service defined with one of the constants.	JSON Object, Service Constant, Callback	Callback pas be called wit Object respo AIS REST AP
callAISServiceGetPath	Call a path parameter based AIS Service with GET method.	Service Constant, Version, Array of path parameters (strings), Callback	Callback pas be called wit Object respo AIS REST AP
callAISServicePath	Call JSON and path parameter based AIS Service with GET method.	JSON Object, Service Constant, Version, Array of path parameters (strings), Callback	Callback pas be called wit Object respo AIS REST AP
callAISServiceFileUpload	Call AIS multipart upload service (Media Object Upload)	JSON Object, Files from file picker, Service Constant, Version, Callback	Callback pas be called wit Object respo AIS REST AP
callAISServiceFileDownload	Call AIS download service (Media Object Download)	JSON Object, File name for download, Service Constant, Version, Callback	Callback pas be called wit Object respo AIS REST AP

SimpleGetDataService	Call AIS Simple Get Data service *Simple Table Query v2* Or *Simple View Query v2*	Use new operator to create a new function with parameters. isTable (boolean), tableOrView, allowCache, fields, filters, filterType, limit, outputType, sorts	Call the exec method on t function obj callback. Cal passed in wi with JSON C response of REST API
SimpleGetFormService	Call AIS Simple Get Form service *Simple Form Service Query v2* or *Simple Form Service Query with Version v2*	Use new operator to create a new function with parameters. appId, formId, appVersion, allowCache, fields, filters, filterType, limit, outputType, sorts	Call the exec method on t function obj callback. Cal passed in wi with JSON C response of REST API
callAISDataServiceNext	Executes next page link from data service response.	Links, Callback	Callback pas be called wit Object respo AIS REST AP
callAISAppStackNext	Executes next page link from app stack response.	Links, Callback	Callback pas be called wit Object respo AIS REST AP
callAISOrchestration	Executes the orchestration requested.	Orchestration, JSON Object, Callback	Callback pas be called wit Object respo AIS REST AP
callAISOrchestrationV2	Executes the orchestration requested.	Orchestration, JSON Object, Callback	Callback pas be called wit Object respo AIS REST AP
callAISOrchestrationUploadFiles	Execute the orchestration requested with Multipart request to upload files.	JSON Object, Files from file picker, orchestration, callback	Callback pas be called wit Object respo AIS REST AP
callAISOrchestrationDownloadFile	Execute the orchestration requested and download expected file output.	JSON Object, orchestration, callback	Callback pas be called. Fo file will be d in browser. I JSON Objec failure detai

callAISOrchestrationUploadAndDownloadFiles	Execute the orchestration requested sending files in and downloading expected file output.	JSON Object, Files from file picker, orchestration, callback	Callback pas be called. Fo file will be d in browser. I JSON Objec failure detai

JET Utility Functions

The JET utility functions are commonly used functions, such as date format, titles, menu functions, and description fetches, that are needed when implementing the template.

Date Format

The jetUtilities file provides a function that can either translate milliseconds or a date string to the EnterpriseOne user's preference date string.

Example - Date Format Function

```
/* Date format function based on user preference */
     var n = Date.now();
     var dateString = createDateString(n);

Date string
     var dateString = createDateString("7/31/2016");
```

JET Titles

This section describes various ways to put a title on your page. Title inputs are not saved when changes are made. They will default to a starting value every time you go back to that page. With the exception of combining a menu field with a title field, when you save the menu field, the title field will display the same value as the menu field.

Standard Title with Translations

To add a standard title to your page, define your title in the initPage of E1AISCalls.js:

```
setTitleFieldText("title1", "Number of New Contracts by Month/Year by Date");
```

Translate your title within jetTranslations. Verify that there is something in translatedArray, and then set the titles with the returned labels.

```
if(translatedArray)
    {
/* z_15.title comes from external form variable ID */
        title = translatedArray.z__15.title;
        setTitleFieldTranslation("title1", title);
    }
```

Title with Date Input

To add a title with date input to your page, define your title in the initPage of E1AISCalls.js:

```
setTitleFieldText("title1", "Number of New Contracts by Month/Year by Date");
setTitleFieldInput(self, "asOfDate", ojInputDate",
 oj.IntlConverterUtils.dateToLocalIso(new Date()), null, null, null, null, "25px");
setTitleFieldText("title2", "Through");
setTitleFieldInput(self, "asOfDateThrough", ojInputDate",
 oj.IntlConverterUtils.dateToLocalIso(new Date()));
```

Notice that in the first setTitleFieldInput, there is a "25px". This allows you to change the width size of the input variable.

Translate your title within jetTranslations. Verify that there is something in translatedArray, and then set the titles with the returned labels.

```
function jetTranslations(self, translatedArray)
{
    //Set your labels
    if(translatedArray)
    {
        title = translatedArray.z__15.title;
        setTitleFieldTranslation("title1", title);

        title = translatedArray.z__16.title;
        setTitleFieldTranslation("title2", title);
     }
}
```

Title with Drop-Down Box

To add a title with a drop-down box to your page, create your drop-down box selectable values with a ko.observableArray in the initPage of E1AISCalls.js:

```
//Build Group By drop-down list
self.selectValues = ko.observableArray([{value: 'assignedTo', label: 'Assigned To'}
                      {value: 'equipment', label: 'Equipment'},
                      {value: 'prodFamily', label: 'Product Family'}]);
```

Define the rest of your title:

```
//Create the change handler used when user changes the drop-down value
self.groupByChangeHandler = function (context, valuParam)
{
    if (valueParam.option == "value")
    {
        if (self.currentGroupByValue != valueParam.value [0])
        {
            var drawer = self.offcanvasMap () ["start"];
            drawer.launcherId = "start";
            drawer.displayMode = "overlay";

            // if it's the active offcanvas, close it
            if (drawer !== self._activeOffcanvas) {
                self.savedGroupByValue(valueParam.value[0]);
                self.getData();
            }
        }
    }
```

```
};

self.savedGroupByValue = ko.observableArray(["assignedTo"]);
self.currentGroupByValue = 'assignedTo';

setTitleFieldText("title1", "Total Downtime Hours Variance by");
setTitleFieldInput(self, "dropDownValue", "ojSelect", self.savedGroupByValue, null, null,
 self.selectValues, "groupByChangeHandler", "150px";
```

Reset your drop-down values in the jetTranslations function:

```
function jetTranslations(self, translated Array)
{
    var groupByTitle = "Group By";

    if(translatedArray)
    {
        //Set chart translated title
        setTitleFieldTranslation("title1", translatedArray.z_15.title);

        // Set y axis translated title
        self.yTitle(translatedArray.z_16.title);

        //set translated group by labels
        self.selectValues ([{value: 'assignedTo', label: translatedArray.z_17.title},
                             {value: 'equipment', label: translatedArray.z_18.title},
                             {value: 'prodFamily', label: translatedArray.z_19.title}]);
```

Menu Functions

Menu fields are found in the drawer, which opens when a user presses the menu icon. The menu field function is called from AISCalls.js in the JetTranslations function. Menu fields are saved when the user presses the save button in the drawer and are the values used when the user re-enters the application.

The function called setMenuFieldID looks like this:

```
setMenuFieldId(self, id, label, inputType, value, converter, validators, options,
 optionChange);

inputType, converter, validators, options, optionChange can be more defined in JET
 cookbook for each inputType.
```

The following table describes the variables used by the setMenuFieldID function:

Variable	Description
self	Self is passed into jetTranslations and needs to be passed on to the function.
id	This is the html ID of the component. This allows the user to reference the html component later on.
label	This is the label placed in front of the input component.

Variable	Description
inputType	inputType is the JET component-defined input. See *http://www.oracle.com/webfolder/technetwork/jet/index.html* for more information.
value	This is the value displayed when the html is rendered.
converter	This variable is used to convert the value that the user inputs.
validators	Validators allow the user to define whether the component should be validated.
options	Options are an array used to define values in a drop-down box.
optionChange	optionChange is a function that is used to detect when the drop-down box value changes.

Example - Setting Menu Fields

```
setMenuFieldId(self, "regularInput", "Regular Input", "ojInputText");
    setMenuFieldId(self, "inputDateTime", "Input Date Time", "ojInputDateTime",
 oj.IntlConverterUtils.dateToLocalIso(new Date()));
    setMenuFieldId(self, "regularNumber", "Regular Number", "ojInputNumber", 1);
    setMenuFieldId(self, "inputDate", "Input Date", "ojInputDate",
 oj.IntlConverterUtils.dateToLocalIso(new Date()));

    var selectValues = [{value: 'Internet Explorer', label: 'Internet Explorer'},
                          {value: 'Firefox',  label: 'Firefox'},
                          {value: 'Chrome',   label: 'Chrome'},
                          {value: 'Opera',    label: 'Opera'},
                          {value: 'Safari',   label: 'Safari'}
                        ];
    setMenuFieldId(self, "selectBox1", "Test", "ojSelect", ["Chrome"], null, null,
 selectValues);

    /* using converter */
    self.converter =
 oj.Validation.converterFactory(oj.ConverterFactory.CONVERTER_TYPE_NUMBER).createConverter(
        {
          "maximumFractionDigits" : 0,
          "minimumFractionDigits" : 0,
          "minimumIntegerDigits" : 2,
          "style" : "decimal",
          "useGrouping" : true
        });

    setMenuFieldId(self, "number", "number", "ojInputText", "123654789", self.converter);

    /* using custom validator. Notice you can make your own error messages used for
 translations */
    /* value is required */
    var validator = [{
        type: 'required', options: {
        required: true,
        messageSummary: 'some title',
        messageDetail: 'Blank not allowed'}}];

    setMenuFieldId(self, "requiredInput", "Required Input", "ojInputText", "", null,
validator);

    /* value is required and length needs to be 3*/
    validator = [{
        type: 'required', options: {
        required: true,
        messageSummary: 'some title',
        messageDetail: 'Blank not allowed'}},
        {
        type: 'length', options: {
        min: 3,
        messageSummary:{
          rangeUnderflow: 'Length needs to be 3 or greater.'},
        messageDetail: {
          rangeUnderflow: 'The length is not in the expected range; it is too low.'}}}];

    setMenuFieldId(self, "requiredInputLength", "Required Input/Length", "ojInputText",
"", null, validator);

    /* Max/min number */
    validator = [{
        type: 'numberRange', options: {
        min: 5,
        max: 10,
        messageSummary:{
          rangeOverflow: 'Number Greater than max.',
          rangeUnderflow: 'Number less than min.'},
        messageDetail: {
          rangeOverflow: 'The value \'{value}\' is not in the expected range; it is too
high.',
          rangeUnderflow: 'The value \'{value}\' is not in the expected range; it is too
low.'}}}];

    setMenuFieldId(self, "inputNumber1", "Input Number range", "ojInputNumber", 0, null,
validator);

    /* date validator */
    validator = [{
        type: 'datetimeRange', options: {
        min: oj.IntlConverterUtils.dateToLocalIso(new Date(1930, 00, 01)),
        max: oj.IntlConverterUtils.dateToLocalIso(new Date(1995, 11,31)),
        messageSummary:{
          rangeOverflow: 'Date later than max.',
          rangeUnderflow: 'Date earlier than min.'},
        messageDetail: {
          rangeOverflow: 'The value \'{value}\' is not in the expected range; it is too
high.',
          rangeUnderflow: 'The value \'{value}\' is not in the expected range; it is too
low.'}}}];

    setMenuFieldId(self, "bDate", "Birth Date", "ojInputDate",
oj.IntlConverterUtils.dateToLocalIso(new Date()), null, validator);

/* selectValues */
self.selectValues ([{value: 'assignedTo',  label: translatedArray.z__19.title},
      {value: 'leadCraft', label: translatedArray.z__20.title},
        {value: 'prodFamily', label: translatedArray.z__21.title},
        {value: 'supervisor', label: translatedArray.z__22.title}]);

    setMenuFieldId(self, "groupByValue", groupByTitle, "ojSelect", self.savedGroupByValue,
 null, null, self.selectValues);
```

Fetch Descriptions

The template has built-in functions to fetch descriptions from six different tables. Some tables have a corresponding 'D' table used to store translated descriptions. In those cases, the translated description based on the User Profile Language preference will be retrieved if one exists.

All descriptions are fetched using the fetchDescriptions function. The function takes an array of description keys and returns a description map. The array of description keys can contain any of the supported tables so the requests can be put into a batch to speed up processing. The tables supported by fetchDescriptions are listed below along with the function used to add to the description array:

Table Name	Function Used
F0006/F0006D - Business Unit Master	addBU(descArray, mcu)
F4101/F4101D - Item Master	addShortItem(descArray, item), addSecondItem(descArray, item), addThirdItem(descArray, item)
F1201/F1201D - Asset Master File	addAsset(descArray, numb)
F0901/F0901D - Account Master	addAccount(descArray, aid)
F0101 - Address Book Master	addAddressBook(descArray, an8)
F0010 - Company Constants	addCompany(descArray, company)

Example - Fetch Descriptions

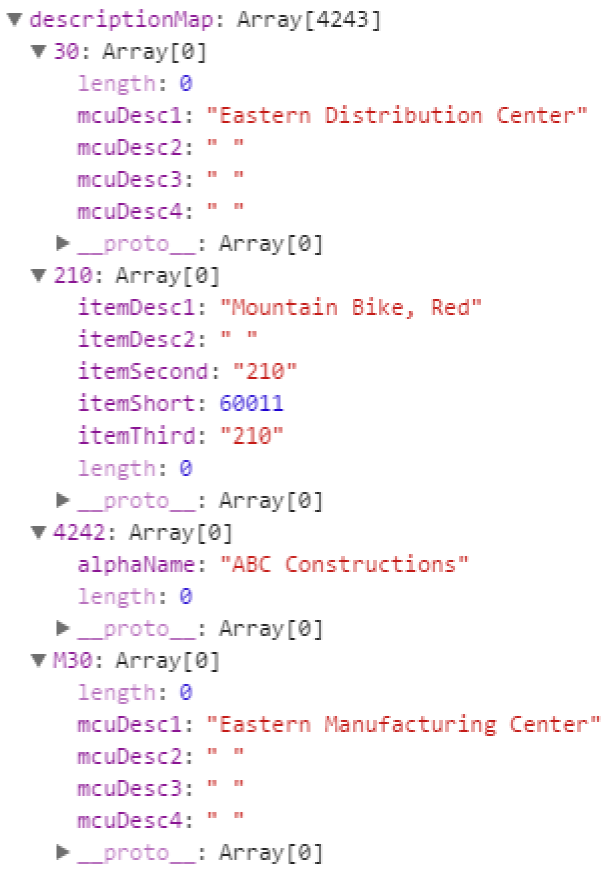
```
    var descArray = [];

    addBU(descArray, "30");
    addBU(descArray, "M30");
    addAddressBook(descArray, "4242");
    addSecondItem(descArray, "210");

    fetchDescriptions(descArray, function(descriptionMap)
    {

    );
```

The descriptionMap variable contains name value pairs. The name corresponds to the key sent into the 'add' function. The value is a grouping of all of the descriptions returned for that key.



```
▼ descriptionMap: Array[4243]
  ▼ 30: Array[0]
      length: 0
      mcuDesc1: "Eastern Distribution Center"
      mcuDesc2: " "
      mcuDesc3: " "
      mcuDesc4: " "
    ▶ __proto__: Array[0]
  ▼ 210: Array[0]
      itemDesc1: "Mountain Bike, Red"
      itemDesc2: " "
      itemSecond: "210"
      itemShort: 60011
      itemThird: "210"
      length: 0
    ▶ __proto__: Array[0]
  ▼ 4242: Array[0]
      alphaName: "ABC Constructions"
      length: 0
    ▶ __proto__: Array[0]
  ▼ M30: Array[0]
      length: 0
      mcuDesc1: "Eastern Manufacturing Center"
      mcuDesc2: " "
      mcuDesc3: " "
      mcuDesc4: " "
    ▶ __proto__: Array[0]
```

Fetch UDC Descriptions

The template has built in functions to fetch UDC descriptions. The function, fetchUDCDescriptions, takes a udcArray that you define prior to calling, as well as a boolean used to fetch all values for a user defined code.

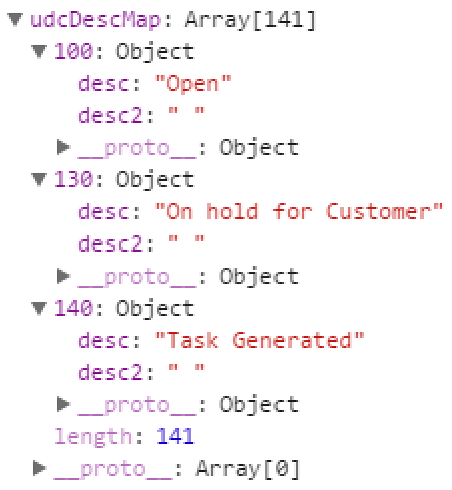
Use setUDCArray to define the udcArray prior to calling fetchUDCDescriptions. The function takes the Product Code and User Defined Code for the UDC, as well as the code to lookup, assuming that fetch all is not being done.

Once setUDCArray has been called for all of the codes to be looked up, fetchUDCDescriptions can be called and it will return a map of all the UDC values.

Example - Fetch UDC Descriptions

```
var udcArray = [];
     setUDCArray(udcArray, "17", "ST", "100");
     setUDCArray(udcArray, "17", "ST", "130");
     setUDCArray(udcArray, "17", "ST", "140");

     fetchUDCDescriptions(udcArray, false, function (udcDescMap)
     {
     });
```



```
▼udcDescMap: Array[141]
  ▼100: Object
     desc: "Open"
     desc2: " "
   ▶__proto__: Object
  ▼130: Object
     desc: "On hold for Customer"
     desc2: " "
   ▶__proto__: Object
  ▼140: Object
     desc: "Task Generated"
     desc2: " "
   ▶__proto__: Object
   length: 141
  ▶__proto__: Array[0]
```

Query Builder

When implementing the template, the Query Builder feature provides a form accessed from the menu that allows the user to define filter criteria for the data. The form is generated dynamically based on the application and form specified in E1AISCalls.js. If an application is specified, all of the form filter fields and QBE columns of the grid are presented to the user for selection and filtering. If a business view or table is specified, all of the columns are available for filtering.

Session State APIs in JET or JavaScript Pages (Release 9.2.5.4)

Note: You must be aware that browser session storage has size limits and potential impact on the performance. Storing a large amount of data in the browser session storage might result in higher browser memory consumption and the high frequency of storage calls can impede the application's performance. It is important to be aware of these limitations when you determine what to store and how often to access the stored data.

Starting with Tools Release 9.2.5.4, the `jetUtilities.js` file provides APIs that can be used to save the session state in an EnterpriseOne JavaScript page. The `jetUtilities.js` file is used in the UX One JavaScript pages and can be used in all EnterpriseOne JavaScript pages.

You can use the session state utilities by leveraging the browser session storage along with the existing external form framework to maintain state in the JavaScript pages. With this capability, when a user navigates away from the page or launches another EnterpriseOne application and then returns to that page, the system restores the state as it was before the user left the page.

Each session state management method depends on a unique key that identifies the stored object based on the instance of the external form and the E1PAGE ID. This key is designed so that the page is uniquely identifiable in all scenarios such as execution of an action from the Fast Path, on the Welcome pages, within a composite page, or on a CafeOne layout. The key is determined by the APIs in the `jetUtilities.js` file.

The state for the page is always stored as a JSON string and the utility methods always attempt to manage the state as an object. Therefore, you need to reference only the state object (and not the string).

The following table describes the four new helper methods used to store the session state:

Function	Parameters	Value Returned	Description
`jde__restoreState()`	None	State Object	Returns the object for this page instance if a state is already stored. If no state is stored, null is returned.
`jde__clearState()`	None	None	Clears the stored state for this page instance.
`jde__saveState(state)`	State Object	None	Saves the object passed in session storage for this page instance.
`jde__manageState(state)`	State Object	None	Makes the required call in document ready function to manage the state.

Session Storage Utilities in JavaScript

This section describes how to use the session storage utilities in the JavaScript pages.

Initializing the State

To use the session storage utilities, you must initialize them in the ready function of your JET model by using the following guidelines:

- Always call the restore state method if it exists so that when the page is refreshed (or the user returns to the page from another application), the state is restored.
- Always call the manage state method during the window unload event. This method ensures that the correct action is performed for the state when the page is unloaded.
- Define a method in your model that applies the state that was restored to the fields on the page.

The following example shows how you can initialize the session storage utilities in the ready function of your JET model:

Example: ready (function ()

```
$(document).ready(function ()
{
 var model = new Model();
 ko.applyBindings(model, document.getElementById('container'));
 if(typeof jde__restoreState === 'function')
 {
 model.applyRestore(jde__restoreState());
 }

 $(window).bind('unload', function()
 {s
 if(typeof jde__manageState === 'function')
 {
 jde__manageState(model.state);
```

```
  }
  });
 });

  //in the Model
  this.applyRestore = function(restoredObject)
  {
  if(restoredObject)
  {
  this.filterValue(restoredObject.filterValue);
  }
  }
```

Saving the State

While using the application, at some point you might want to update the stored state object, for example, after a filter value changes or after a search is performed. During these instances, you would not want the users to lose their work. You call the save state method on these events.

To save the state, use the following guidelines:

- Define an object in the model that is the state object.
- On events where the state should be updated, update the object and use the save state method to store it.

The following example shows how to define the save state method:

Example: saveState Method

```
this.state =
{
 filterValue: this.filterValue()
}

this.saveValueState = function(event, data, bindingContext)
{
 if(typeof jde__saveState === 'function')
 {
 bindingContext.$data.state.filterValue = event.detail.value;
 jde__saveState(bindingContext.$data.state);
 }
}
```

Clearing the State

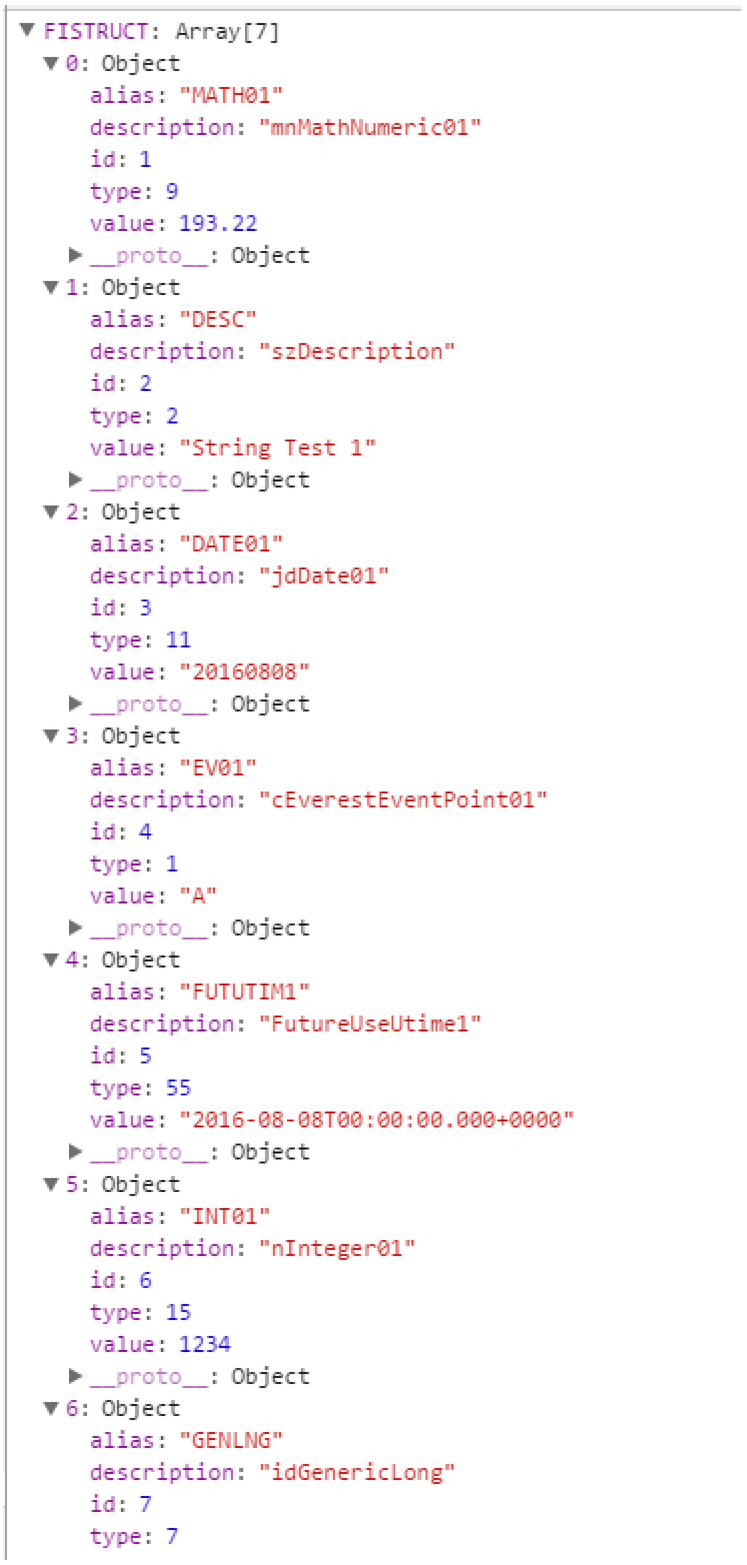
If the instance of the form is closed or refreshed, the system automatically clears the state. To manually clear the state in your application, you can use the clear function.

6 EnterpriseOne JET Tools and Tips

Consuming Form Interconnect Values from External Forms

Although consuming form interconnect values is not commonly used within the template, you can use the following guidelines to consume form interconnect values.

When an external form launches a Classic EnterpriseOne Page (JET Page), the form interconnect or CafeOne layout passes values to the JavaScript object, FISTRUCT. If there is no value for the interconnect value, then it will not be passed to the JavaScript object in the FISTRUCT.



```
▼FISTRUCT: Array[7]
  ▼0: Object
      alias: "MATH01"
      description: "mnMathNumeric01"
      id: 1
      type: 9
      value: 193.22
    ▶__proto__: Object
  ▼1: Object
      alias: "DESC"
      description: "szDescription"
      id: 2
      type: 2
      value: "String Test 1"
    ▶__proto__: Object
  ▼2: Object
      alias: "DATE01"
      description: "jdDate01"
      id: 3
      type: 11
      value: "20160808"
    ▶__proto__: Object
  ▼3: Object
      alias: "EV01"
      description: "cEverestEventPoint01"
      id: 4
      type: 1
      value: "A"
    ▶__proto__: Object
  ▼4: Object
      alias: "FUTUTIM1"
      description: "FutureUseUtime1"
      id: 5
      type: 55
      value: "2016-08-08T00:00:00.000+0000"
    ▶__proto__: Object
  ▼5: Object
      alias: "INT01"
      description: "nInteger01"
      id: 6
      type: 15
      value: 1234
    ▶__proto__: Object
  ▼6: Object
      alias: "GENLNG"
      description: "idGenericLong"
      id: 7
      type: 7
```

Before attempting to use the FI values you should check if you have the structure. You will not have it if the page is running in the welcome tab strip:

```
if (formInterconnectAvailable())
```

You can get individual values from the structure by id:

```
getFormInterconnectById(1)
```

You can get individual values from the structure by description:

```
getFormInterconnectByDesc("szDescription")
```

Or you can get the value (or type) by index, directly from the structure (or any other members of the object):

```
FISTRUCT[i].value
FISTRUCT[i].type
```

You should also check for blank/null values before using the values since they may not have been passed into the form interconnect or in the CafeOne layout configuration.

Consuming Application Version and Processing Options from External Forms (Release 9.2.5.4)

Starting with Tools Release 9.2.5.4, four additional JavaScript variables are available when a JavaScript (JET) page runs on an external form. The new JavaScript variables are used to enable the JET page to consume the application version and the processing options for the external application.

The following table describes the JavaScript variables used to consume the external application version and processing options:

Variable	Value
`jde__appName`	The ID (string) of the application that is associated with the external form.
`jde__appForm`	The ID (string) of the external form.
`jde__appVersion`	The currently running version (string) of the application associated with the external form.
`jde__POs`	An object with entries (object) for each of the processing options, with their type (integer) and value (string).

The following is an example of JavaScript values passed to an external application, P99EXTF2:

- **`jde__appName: "P99EXTF2"`**
- **`jde__appForm: "W99EXTF2A"`**
- **`jde__appVersion: "ZJDE0001"`**
- **`jde__POs: Object`**
  - **`cCHAR_4: {type: 1, value: "1"}`**
  - **`jdDate01_3: {type: 11, value: "20210217"}`**

- `mnAddressNumber_1:`
    - `type: 9`
    - `value: "6001"`
- `nInteger01_5: {type: 15, value: "1234"}`
- `szDescription001_2: {type: 2, value: "abcde"}`

With Tools Release 9.2.5.4, the processing option helper API in the `jetUtilites.js` file has been updated. The `getExtAppPODetails(self)` method populates the standard `poDetails` object in the application model with the processing option values from the external form and the system no longer makes an AIS call to retrieve the processing option values. Use the `getExtAppPODetails(self)` method to populate the `poDetails` value in the `self` (model) object.

After the `getExtAppPODetails(self)` method is processed, the `poDetails` variable in the `self` (model) object is populated as shown in the following example:

- `self.poDetails: Object`
    - `cCHAR_4: {type: 1, value: "1"}`
    - `jdDate01_3: {type: 11, value: "20210217"}`
    - `mnAddressNumber_1: {type: 9, value: "6001"}`
    - `nInteger01_5: {type: 15, value: "1234"}`
    - `szDescription001_2: {type: 2, value: "abcde"}`

About Running JET Applications in Composite Application Framework

You can launch JET applications from a Composite Application Framework only if you are using an external form with the JET application.

You cannot maximize a JET or ADF application in a Composite Application Framework.

For more information on external forms, see *"Understanding External Forms" in the JD Edwards EnterpriseOne Tools Form Design Aid Guide .*

For more information on Composite Application Framework, see *"Creating EnterpriseOne Form Content" in the JD Edwards EnterpriseOne Tools Composite Application Framework (CafeOne) User's Guide .*

7 Appendix A - Troubleshooting

AIS Login Error

If you see an AIS login error in the chart area or in the log after you've deployed to EnterpriseOne (even though your code worked locally), it might be caused by one of the following:

- The JAS Server is not configured to point to the AIS Server.
- The AIS Server does not have the PSToken login type enabled. This is configured in Server Manager.

503 or 500 Could Not Process Request Errors

The AIS call is very sensitive and has a specific structure that you must follow. If you are receiving these errors, examine he JSON payload of the request that is built into your page very closely for issues.

You Cannot Add More Content to a Composed Application Framework Layout

The maximum content for Composed Application Framework on the EnterpriseOne HTML Server has a default value of 4, in order to prevent a layout with too many satellite forms and the user running out of allowed application instances, which is 10 by default. However, you can change this value.

Go to Server Manager for the EnterpriseOne HTML Server, and select "Advanced" under Configuration >Web Runtime. Under the Web Objects Settings section, increase the value of the CafeOne: Maximum contents per layout property and bounce the server.

Performance Considerations

This section contains the following topics:

- *HTML Server Instance for Processing AIS Server Requests*
- *"Run When Selected" Option for Queries*
- *Using isDataRequest for AIS Call*
- *Concurrent Calls to AIS Server (Release 9.2.1.1)*

HTML Server Instance for Processing AIS Server Requests

The performance of each UX One page depends on the number of rows each component retrieves, the amount of logic each component processes, and the number of graphical and tabular components on the page.

To improve the performance of UX One pages, you should set up an additional HTML Server instance for processing AIS Server requests only. This is recommended so that the performance of the EnterpriseOne HTML Server used by EnterpriseOne web client users is not impacted by AIS Server requests.

For more information, see *"Additional EnterpriseOne HTML Server Instance for Processing AIS Requests" in the JD Edwards EnterpriseOne Application Interface Services Server Reference Guide*

"Run When Selected" Option for Queries

When creating a query used by Watchlists, make sure that the "Run When Selected" option is selected. When creating a query to be used by a JET analytical component on a Composed Page, it is recommended that this option be turned off for better performance.

Using isDataRequest for AIS Call

When performing an AIS call to get data from an application, you have the option to get the data directly from the business view associated with that application by setting isDataRequest to true. This will perform better than running the application using a form service request and loading the grid, but there are some things to keep in mind. There may be event rules running in the application that get additional data like associated descriptions or modify data that you want to display in your JET application. These may not be in the business view that the application is using. In those cases, it may be better to perform a form service request at the expense of some performance.

By using "isDataRequest = true" over an application rather than building your chart directly from a table or business view, you get the benefit of using the queries defined for that application (users may not have permissions to define queries in Data Browser), and the Query Builder form will be based on the application. It is usually easier to build a query in Query Builder when it is based on an application since you see the filter field labels as they appear in the form rather than the Data Dictionary descriptions of all of the columns in the table or business view.

Concurrent Calls to AIS Server (Release 9.2.1.1)

If a UX One Page is taking too long to refresh, you might have a memory issue.

In Server Manager, use the "AIS Maximum Concurrent Calls" setting in the HTML Server settings to adjust the number of concurrent calls to the AIS Server. This setting can be found within the Web Runtime, Form Service settings for the HTML Server.

If you still encounter performance issues, you can inspect Garbage Collection logs to check the amount of space and memory allocation. If you notice that the memory usage is high, you might need to boost your memory or CPU size.