Oracle Fusion Cloud Customer Experience

Integrating Sales with JD Edwards EnterpriseOne

April 2023

Oracle Fusion Cloud Customer Experience
Integrating Sales with JD Edwards EnterpriseOne

April 2023

F77855-01

Author: David Yetter

Contents

Get Help

There are a number of ways to learn more about your product and interact with Oracle and other users.

Get Help in the Applications

Use help icons to access help in the application. If you don't see any help icons on your page, click your user image or name in the global header and select Show Help Icons.

Get Support

You can get support at *My Oracle Support*. For accessible support, visit *Oracle Accessibility Learning and Support*.

Get Training

Increase your knowledge of Oracle Cloud by taking courses at *Oracle University*.

Join Our Community

Use *Cloud Customer Connect* to get information from industry experts at Oracle and in the partner community. You can join forums to connect with other customers, post questions, suggest *ideas* for product enhancements, and watch events.

Learn About Accessibility

For information about Oracle's commitment to accessibility, visit the *Oracle Accessibility Program*. Videos included in this guide are provided as a media alternative for text-based topics also available in this guide.

Share Your Feedback

We welcome your feedback about Oracle Applications user assistance. If you need clarification, find an error, or just want to tell us what you found helpful, we'd like to hear from you.

You can email your feedback to *oracle_fusion_applications_help_ww_grp@oracle.com*.

Thanks for helping us improve our user assistance!

1 About This Guide

Audience and Scope

This guide is intended for anyone who is involved in integrating JD Edwards EnterpriseOne with Oracle CX Sales.

You must perform the integration steps in this guide to integrate JD Edwards EnterpriseOne with Oracle CX Sales.

If you want to set up and work with the additional features of Oracle CX Sales, see Oracle CX Sales documentation on Oracle Help Center at https://docs.oracle.com.

This document describes features available to users under Oracle CX Sales, Oracle Fusion Service, and Oracle Engagement Cloud licensing agreements.

Related Topics

Related Guides

The following table lists related guides which provide more information about the integration tasks covered in this guide.

Title of Guide	Guide Description
Getting Started with Your Sales Implementation	Describes how to set up a sales automation solution in Oracle CX Sales using a case study to describe concepts and procedures.
Implementing Sales	Describes how to configure and set up CX Sales.
Extending CX Sales and Fusion Service	Describes how to use tools to configure CX Sales and Fusion Service.
Understanding Import and Export Management for CX Sales and Fusion Service	Describes how to import legacy and other data into Oracle CX Sales and Fusion Service using the Import and Export Management feature.
JD Edwards EnterpriseOne Documentation	Describes core features of JD Edwards EnterpriseOne.

Related Topics

2 Introduction to the Integration

Overview of Oracle CX Sales and JD Edwards EnterpriseOne Integration

This guide outlines the implementation and configuration steps that are required to integrate customer- and opportunity-management processes in Oracle CX Sales with business processes in JD Edwards EnterpriseOne.

> **Note:** This guide assumes a familiarity with Oracle CX Sales Application Composer, and JD Edwards Business Services. For information about JD Edwards Business Services, refer to: https://support.oracle.com/epmos/faces/DocumentDisplay?id=967281.1. For information about Oracle CX Sales Application Composer and configurations, refer to the Extending Sales and Service guide on Oracle Help Center.

The integration is designed to support customers who want to take advantage of the latest capabilities of the Oracle CX Sales application, and use their existing but still use their existing investment in an on-premise Enterprise Resource Planning (ERP) system that provides pricing and discount information, and quote and sales order processing. In the integration, JD Edwards EnterpriseOne quotes can be created from Oracle CX Sales opportunities. The JD Edwards EnterpriseOne Engagement Order Entry user interface which is used to enter and update sales orders and sales quotes is embedded within a tab in the Opportunity Detail screen in Oracle CX Sales.

This guide is meant to be used as a template. The guide is a starting point that shows how Oracle CX Sales and JD Edwards EnterpriseOne can be connected to create a value-added business process and user experience. You must enter the documented configurations and install the required JD Edwards EnterpriseOne Electronic Software Update (ESU) JM16431. For more information about installing the ESU, refer to the Applying the JD Edwards EnterpriseOne Electronic Software Update (ESU) topic within the Software Requirements for JD Edwards EnterpriseOne topic.

However, it is not a turnkey solution. Each implementation of Oracle CX Sales and JD Edwards EnterpriseOne is unique, and each customer has different needs that have led them to implement application configurations that support their unique business requirements. While the steps in this guide describe how to connect a non-configured Oracle CX Sales instance to a non-configured JD Edwards EnterpriseOne instance, they can be combined with configurations that have already been applied to a customer's instances. Note that most settings related to security are not included in this document. For information and recommendations, refer to Oracle CX Sales Security Reference and Securing Oracle CX Sales both on Oracle Help Center (https://docs.oracle.com).

Related Topics

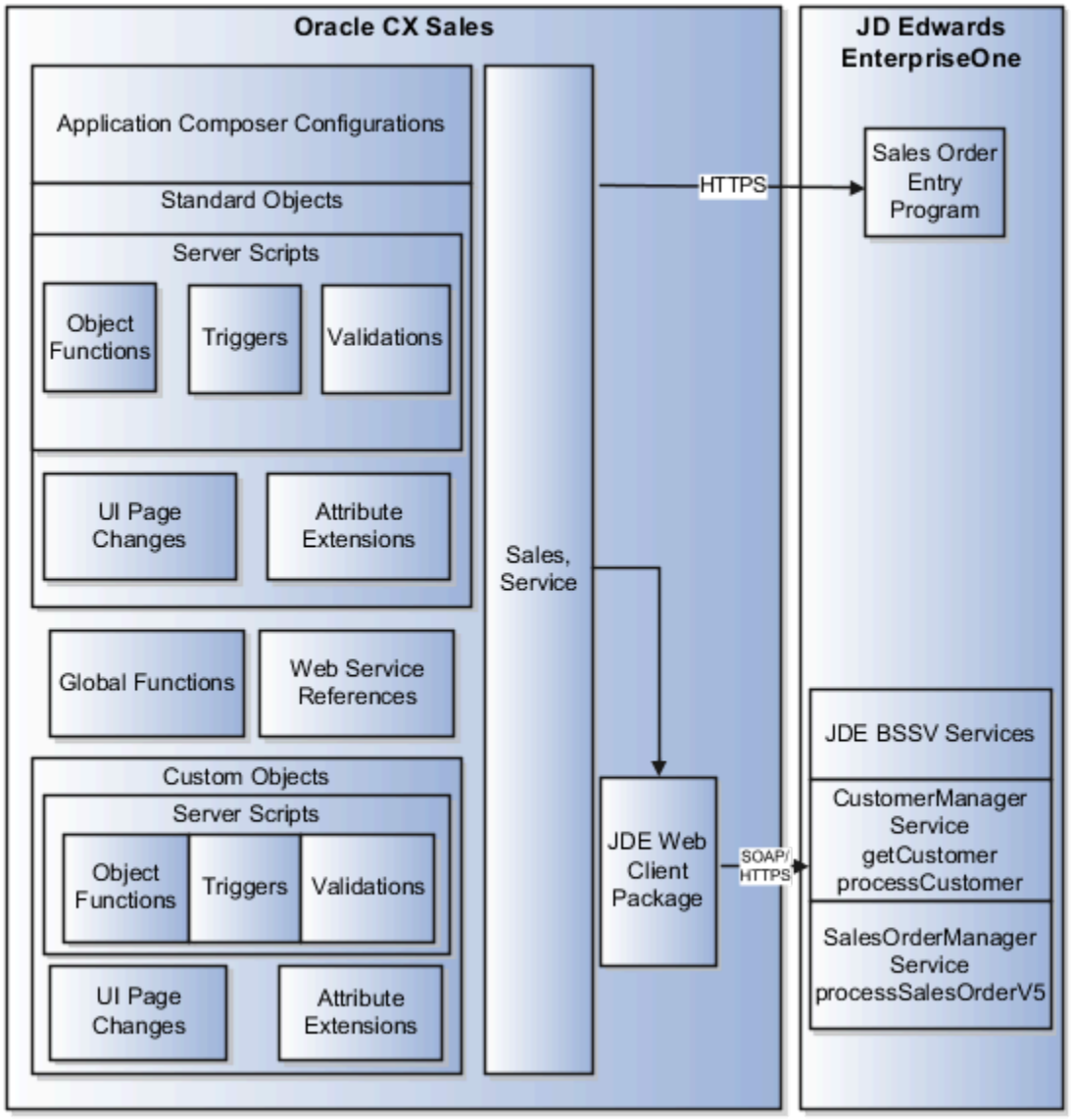
- Software Requirements for JD Edwards EnterpriseOne

Oracle CX Sales and JD Edwards EnterpriseOne Integration Component Architecture

The following lists the files required to implement the integration between Oracle CX Sales and JD Edwards EnterpriseOne.

Refer to the Related Topics section in this topic for a link to Integrating Oracle CX Sales with JD Edwards EnterpriseOne (Doc ID 1645923.1) on My Oracle Support. The files are located in the Attachments section of the article. The following figure shows the components of the integration between Oracle CX Sales and JD Edwards EnterpriseOne and their relationship with each other.

- **Oracle CX Sales standard objects.** Standard objects are configured by adding fields, object functions, triggers, and validations, and by configuring UI pages.
- **Oracle CX Sales custom objects.** Custom objects are created for the integration, and fields, pages, functions, and so on are added to the objects.
- **Groovy functions.** Groovy scripts create functions, triggers, and validations that modify the action of Oracle CX Sales objects.
- **JD Edwards EnterpriseOne web services.** Oracle CX Sales calls JD Edwards EnterpriseOne web services directly. For more information about web services, refer to the Integration Services topic.
- **JD Edwards EnterpriseOne Sales Order Entry Program.** Opportunities in the Sales application in Oracle CX Sales are integrated with the Sales Order Entry Program in JD Edwards EnterpriseOne.



Related Topics

JD Edward EnterpriseOne Integration Services

The following JD Edwards EnterpriseOne web services are used in the integration:

- **CustomerManager Service**. This web service manages customer records and is used in the customer synchronization process between Oracle CX Sales and JD Edwards EnterpriseOne by searching for matching customers in JD Edwards EnterpriseOne as well as creating customers if no suitable match is found.

- **SalesOrderManager Service**. This web service manages order and quote records and is used to create quotes and orders in JD Edwards EnterpriseOne based on Oracle CX Sales opportunities.

Oracle CX Sales and JD Edwards EnterpriseOne Integration Process Flows

The integration supports the following main process flows:

- Matching customer data between Oracle CX Sales and JD Edwards EnterpriseOne.
- Creating sales orders and sales quotes in JD Edwards EnterpriseOne from Oracle CX Sales opportunities.
- Importing JD Edwards EnterpriseOne Items into Oracle CX Sales Catalog.
- Viewing reports in JD Edwards EnterpriseOne from Oracle CX Sales opportunities.

How Customer Data is Matched in JD Edwards EnterpriseOne

This process lets you match customer records in Oracle CX Sales with JD Edwards EnterpriseOne. During the customer synchronization process, existing JD Edwards EnterpriseOne customer records are searched in order to find suitable matches.

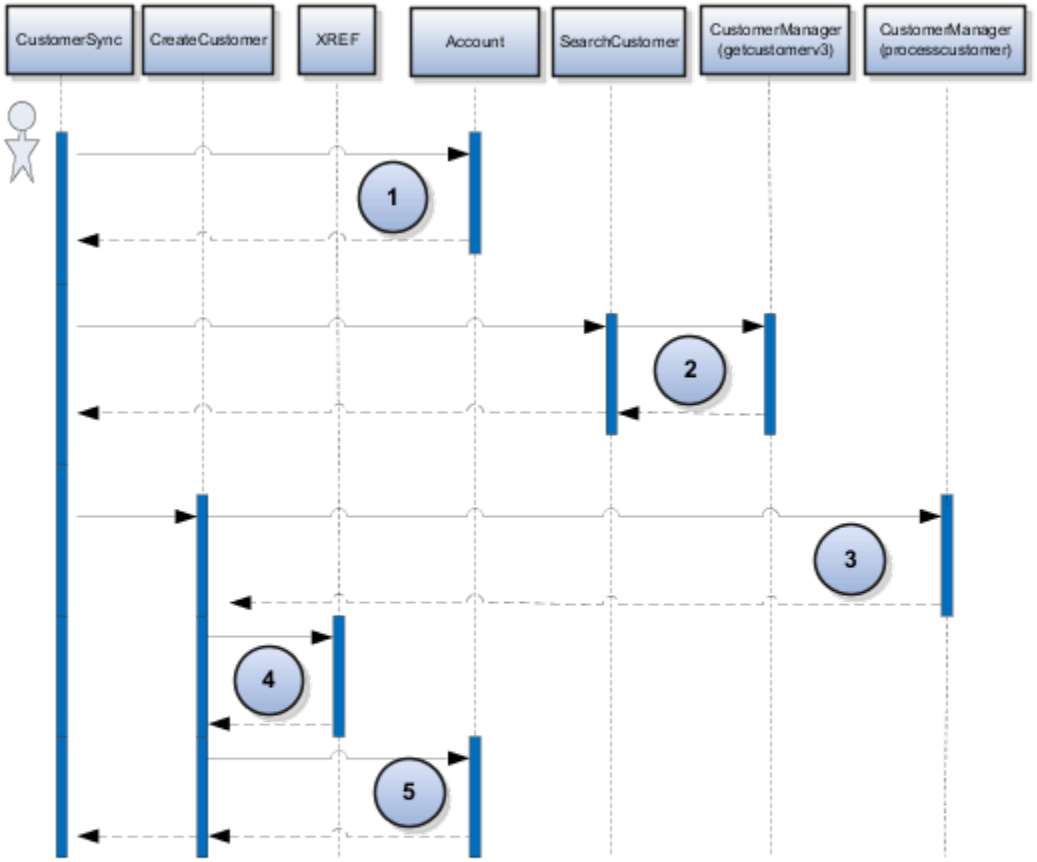
If the customer record exists in JD Edwards EnterpriseOne and there is only one match, then the process saves a cross-reference in Oracle CX Sales to the JD Edwards customer record. The cross-reference contains the local identification in Oracle CX Sales and the remote identification in JD Edwards. The integration can be configured, as described in this chapter, to disable automatic matching when one exact match is found, thus activating the manual matching process described in the following paragraph. Automatic matching is controlled by the "enableAutoModeForSingleMatch" integration configuration parameter. For more information on configuring this parameter, refer to Connecting Oracle CX Sales to JD Edwards EnterpriseOne.

If the customer record exists in JD Edwards EnterpriseOne and has multiple potential matches, then the user is presented with a UI where the potential matches are displayed. The user selects the candidate match. The process then associates the record in JD Edwards EnterpriseOne with the record in Oracle CX Sales by saving the cross-reference. If none of the returned matches is suitable the user also has the option to create a new customer in JD Edwards EnterpriseOne.

If the customer record does not exist in JD Edwards EnterpriseOne, then a new customer record is created in JD Edwards EnterpriseOne. The process then saves a cross-reference in Oracle CX Sales with the local identification in Oracle CX Sales and the remote identification in JD Edwards EnterpriseOne.

Detailed Customer Data Matching Process

The following figure illustrates the customer data matching sequence.



The process checks the current Account synchronization status in Oracle CX Sales. If the Account is not synchronized then the process calls the CustomerManager published web service which calls the getcustomerv3 method to perform a customer search in JD Edwards EnterpriseOne. The customer search returns a list of matching customer records which are then stored in the customer matches object in Oracle CX Sales.

If there is no matching record, then the CustomerManager published web service calls the processcustomer method which returns the remote customer identification for the new customer created. If there was one exact match, or a new customer was created in JD Edwards EnterpriseOne, the process creates an XREF custom object record in Oracle CX Sales using the local identification in Oracle CX Sales and the remote identification from JD Edwards EnterpriseOne. The Groovy logic invoked by triggers sets the synchronization status to Synchronized.

For additional information about the getcustomerv3 and processcustomer web services used to process customer information, refer to the Customer Master Data chapter in the JD Edwards EnterpriseOne Applications Business

Interface Implementation Guide. You can access this guide on the Cross-Product tab of the JD Edwards EnterpriseOne Applications Documentation Library, located at: http://docs.oracle.com/cd/E16582_01/index.htm.

How Sales Orders and Sales Quote are Created in JD Edwards EnterpriseOne

This process lets you create sales quote and sales order records in JD Edwards EnterpriseOne based on opportunities created in Oracle CX Sales.

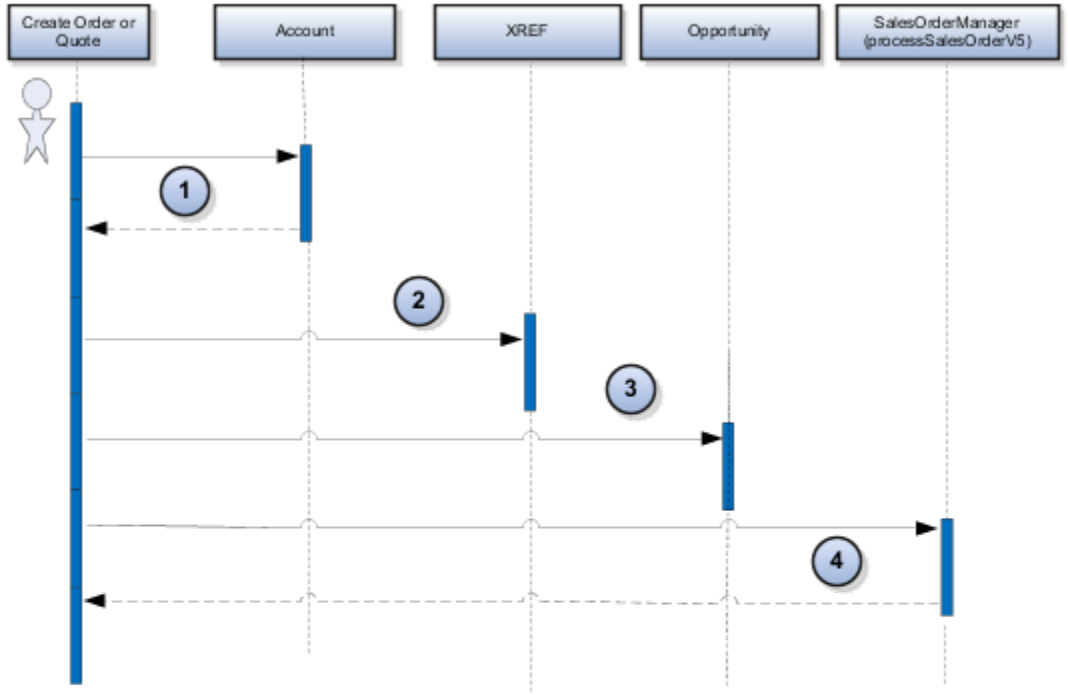
After the customer record is created in Oracle CX Sales and synchronized with JD Edwards EnterpriseOne, you can access the JD Edwards Sales Order Entry program to enter sales quotes and sales orders.

This embedded Sales Order Entry program UI allows the user to take advantage of JD Edwards EnterpriseOne directly in Oracle CX Sales. All of the product, pricing, and configuration details associated with the sales quote and sales order are captured in JD Edwards EnterpriseOne. The process also maps revenue line items in opportunity records in Oracle CX Sales to quote line items in JD Edwards EnterpriseOne.

Note that the user will be able to access JD Edwards Enterprise One Quotes and Orders from both the Opportunity UI as well as the Account UI in Oracle CX Sales. This is accomplished by embedding portions of the JD EdwardsEnterprise One UI inside Oracle CX Sales. The Sales Order and Quote Creation process described in the following example is used when creating Quotes or Orders directly in the Oracle CX Sales Opportunity UI based on the Opportunity Revenue Lines.

Detailed Sales Order and Sales Quote Creation Process

The following figure details the sequence of sales order and sales quote creation.



When a sales representative clicks Create Quote or Create Order on an Opportunity page in Oracle CX Sales:

1. The process returns the Account associated with the opportunity in Oracle CX Sales, and determines the synchronization status.
2. If the Account is synchronized, then the process reads the remote contact identification from the XREF object. If the Account is not synchronized, then an error is returned.
3. The quote/order creation process collects opportunity details such as revenue line items from the Opportunity Revenue object.

4. The quote/order creation process synchronizes all the revenue lines using Product Groups imported from JD Edwards EnterpriseOne Items to the quote/order being created. If the integration is configured to map all revenue lines and if at least one does not map then an error is returned.

   Revenue mapping is controlled by the "enforceRevenueLineMapping" integration configuration parameter. For more information on configuring this parameter, refer to Connecting Oracle CX Sales to JD Edwards EnterpriseOne.
5. The quote/order creation process calls the SalesOrderManager published business service which calls the processSalesOrderV5 method. The method returns the new quote/order number from JD Edwards EnterpriseOne.
6. The quote/order creation status, which includes the quote/order number, is updated in the Opportunity. This status does not persist with the Opportunity.

   For additional information about quote and sales order configuration and processing in the JD Edwards EnterpriseOne, refer to the JD Edwards EnterpriseOne Applications Sales Order Management Implementation Guide. You can access this guide on the SCM and MFG tab of the JD Edwards EnterpriseOne Applications Documentation Library, located at: http://docs.oracle.com/cd/E16582_01/index.htm.

How JD Edwards EnterpriseOne Items are Imported to an Oracle CX Sales Catalog

To allow you to take advantage of the Oracle CX Sales opportunity to JD Edwards EnterpriseOne quote and order integration, you must first import all JD Edwards EnterpriseOne Items into the Oracle CX Sales Catalog.

These imported Items will then be used when creating revenue lines in Oracle CX Sales opportunities.

For additional information about the Item Batch Outbound program (R4101ZO) used to create the item XML file, refer to the Item Master chapter in the JD Edwards EnterpriseOne Applications Business Interface Implementation Guide. You can access this guide on the Cross-Product tab of the JD Edwards EnterpriseOne Applications Documentation Library, located at: http://docs.oracle.com/cd/E16582_01/index.htm.

Detailed Item Import Process

When a user first exports a JD Edwards EnterpriseOne items list to XML:

1. The user processes the export XML file using a provided script to transform the file to the format expected by the Oracle CX Sales Import Management process.
2. The user creates a new top-level JD Edwards EnterpriseOne Catalog Product Group in Oracle CX Sales. All JD Edwards EnterpriseOne items included in the XML file will be imported into this product group.

   Note: This step must only be performed only once during the initial import.
3. The user starts a new Product Group import using the XML file generated in step 1. You can repeat this process as required to import changes to JD Edwards EnterpriseOne items into Oracle CX Sales product catalog.

   JD Edwards EnterpriseOne item export is done using UTF-8 character encoding.

   For more information about how to complete these steps, refer to the Postconfiguration Tasks chapter.

Related Topics

- Overview of Post-Configuration Tasks
- Import Products
- Create a JD Edwards EnterpriseOne Catalog
- Perform Initial Import of JD Edwards EnterpriseOne Items from the XML Import File

How to View Reports in JD Edwards EnterpriseOne

The integration also exposes two JD Edwards EnterpriseOne One View Reporting (OVR) reports which are called Open Sales Analysis and Sales by Opportunity.

These reports are generated using the Oracle CX Sales customer context. You access the reports from within the Oracle CX Sales Account Details screen once the account has been synchronized withJD Edwards EnterpriseOne.

For additional information about the One View reports used to view sales opportunity information, refer to the One View Reporting for Sales Order Management chapter in the JD Edwards EnterpriseOne Applications One View Reporting User Guide. You can access this guide on the One View tab of the JD Edwards EnterpriseOne Applications documentation library, located at: http://docs.oracle.com/cd/E16582_01/index.htm.

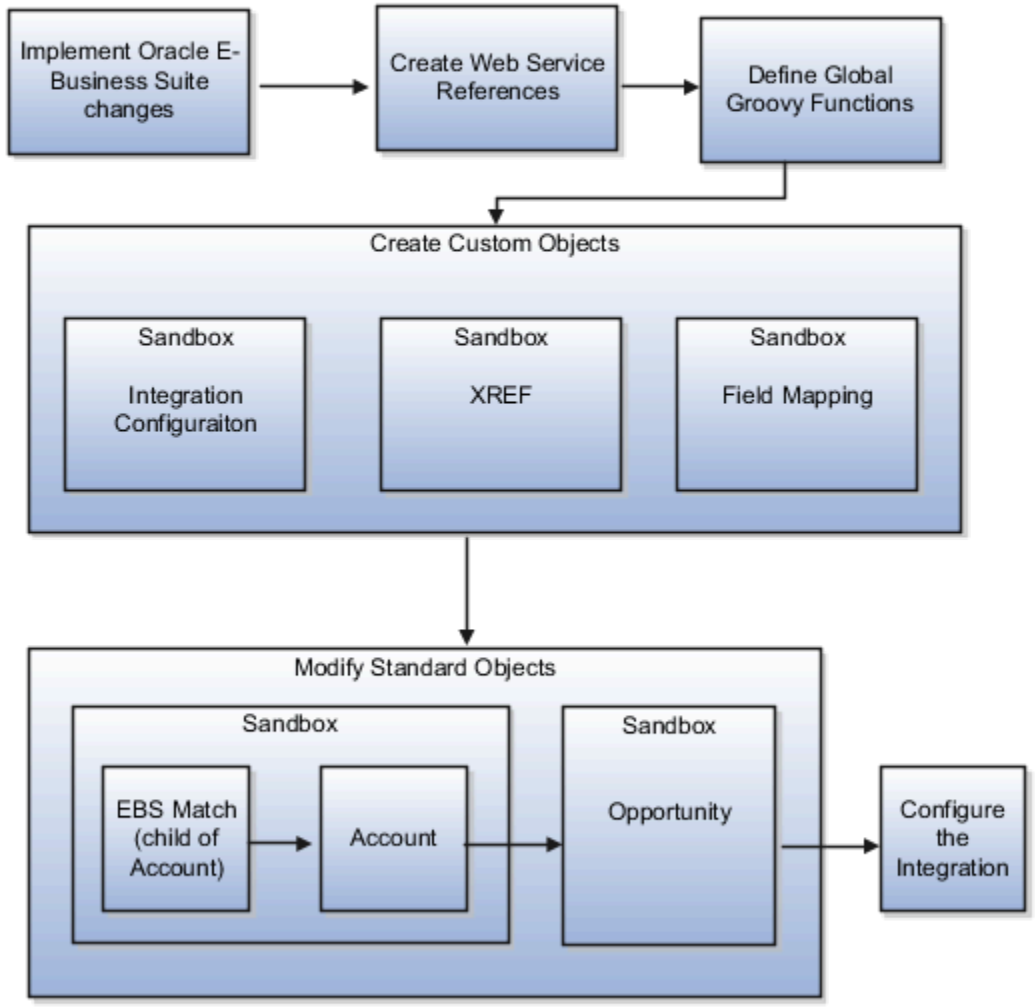
Related Topics

Configuration Roadmap

To use this integration, you must configure application artifacts in both Oracle E-Business Suite and Oracle CX Sales.

In the Oracle E-Business Suite instance several patches must be applied, and web services must be configured and exposed. In the Oracle CX Sales instance, custom Groovy logic must be added, custom objects created, and standard objects enhanced. These steps must be executed in order, because many of them rely on the successful completion of previous steps. The following figure shows a high-level overview of the configuration sequence.

The following figure displays a high-level overview of the configuration sequence.



Many of the Oracle CX Sales configuration steps in Chapter 3 have several component procedures, such as creating sandboxes, creating or augmenting objects, defining fields, defining business logic with functions, defining rules or triggers, and defining user interface changes. These component procedures are used in many different configuration steps, but not all procedures are used in all steps. For example, the procedure to create an object or the process to define a trigger is the same from one step to the next, but the names and details of the objects or triggers will change from step to step.

The general process for each component procedure is listed before the specific objects that are to be created or configured. The topics for the specific objects provide details on each of the configuration steps, and which component procedures and details are required in each step.

3 JD Edwards EnterpriseOne Configuration

Software Requirements for JD Edwards EnterpriseOne

This integration requires JD Edwards EnterpriseOne Tools release 9.1.2.0 or higher.

For the list of supported web browsers for JD Edwards EnterpriseOne, see JD Edwards EnterpriseOne 9.2.X Web Client Support Statement (Doc ID 2059836.1) on My Oracle Support, and JD Edwards EnterpriseOne 9.1.X Web Client Support Statement (Doc ID 1487909.1) on My Oracle Support.

Applying the JD Edwards EnterpriseOne Electronic Software Update

Install the required JD Edwards EnterpriseOne Electronic Software Updates from JD Edwards EnterpriseOne Update Center. Make sure you build and deploy the BSSV package with JAX-WS option ONLY. To install the JD Edwards EnterpriseOne Electronic Software Update do the following:

1. Locate the JD Edwards EnterpriseOne Update Center is at the following location: https://updatecenter.oracle.com/apps/WebSearch/updatecenter.jsp?
2. Perform a search for the following required updates which you must apply:
    a. Perform a search for an ESU which contains Bug 17848773 (Electronic Software Updates type on the Update Center).
    b. Perform a search for an ESU which contains Bug 18862946 (One View Reporting|E1 Pages type on the Update Center).
    c. Perform a search for an ESU which contains Bug 20600497 (Electronic Software Updates type on the Update Center).
    d. Perform a search for an ESU which contains Bug 26549687 (Electronic Software Updates type on the Update Center).

Upgrade JD Edwards Tools to Release 9.1.4.7

Oracle recommends that you upgrade your JD Edwards Tools version to at least version 9.1.4.7 to resolve known issues with the embedded UI.

You can upgrade your tools maintenance pack by navigating to Software Update Installation Guides - All JD Edwards EnterpriseOne Application Release.

https://support.oracle.com/epmos/faces/DocumentDisplay?id=741506.1

If you do not want to upgrade your JD Edwards Tools pack, verify the expiration against the Tools release version you are currently using. If you are using a supported release, contact JD Edwards EnterpriseOne support for an engineered fix for the following bugs: 18186551,18451145, and 18186594.

> **Note:** You should apply this POC if you are using Internet Explorer 9 or later, or any supported release of Firefox, or Chrome browser.

Related Topics

Set the Default Branch Plant Value

You must set the default branch plant for JD Edwards EnterpriseOne Sales Order Entry, and One View Report processing applications.

Before you begin configuring Oracle CX Sales for the integration, you must set the following processing option values in the JD Edwards EnterpriseOne.

1. Set the Default Branch Plant option in the Sales Order Entry program (P4210) as described in the following example.

   In the Default Branch Plant option on the Default tab of the Sales Order Entry (P4210) processing options. For additional information about these processing options, see Setting Processing Options for Sales Order Entry (P4210) in the JD Edwards EnterpriseOne Applications Sales Order Management Implementation Guide. You can find this guide by selecting the **SCM and MFG** link on the JD Edwards EnterpriseOne Applications Documentation Library, located at: http://docs.oracle.com/cd/E16582_01/index.htm.
2. Set the Branch Plant - Detail and the Order Company (Order Number) options on both the Open Sales Inquiry program (P42270), and the One View Sales To Date program (P42272) as described in the following example.

   For additional information about the One View report processing options, see the One View Reporting for Sales Order Management chapter in the JD Edwards EnterpriseOne Applications One View Reporting User Guide. You can find this guide on the One View tab of the JD Edwards EnterpriseOne Applications Documentation Library, located at: http://docs.oracle.com/cd/E16582_01/index.htm.

Embedding the JD Edwards EnterpriseOne Application Inside an IFrame: Explained

To ensure that JD Edwards EnterpriseOne application is properly embedded within an IFrame, verify that your security settings are configured as they are described in this topic.

If these settings are not properly configured, JD Edwards EnterpriseOne will not launch inside an IFrame, but will instead display in a new window. To embed JD Edwards

EnterpriseOne inside an IFrame do the following:

1. If you are modifying the jas.ini file directly, add the following entries in the Security section:

```
[SECURITY]
frameBustingForApp=never
frameBustingForE1Menu=never
frameBustingForLogin=never
```

2. If you are making the modifications on the server manager, change the settings through the UI to the settings listed in the following table.

Parameter	Value
Strict Version Security	Web client version security model
Frame Busting For Login	Never
Frame Busting For E1 Menu	Never
Frame Busting For E1 Application	Never

3. Bounce the server for the changes to take effect.

4 Oracle CX Sales Configuration

Software Requirements for Oracle CX Sales

The integration works with Oracle CX Sales.

For the list of supported web browsers for Oracle CX Sales, see: https://www.oracle.com/system-requirements/index.html.

Related Topics

Create References to JD Edwards EnterpriseOne

You must create CX Sales references to the JD Edwards EnterpriseOne web services required for the integration. This is performed in the Application Composer.

When registering the first web service reference in Oracle CX Sales, you must create a new credential key, which Oracle CX Sales will use to authenticate the web service calls to JD Edwards EnterpriseOne. Choose a JD Edwards EnterpriseOne administrative user account. All web service calls will be executed by authenticating as this user.

Creating Web Service References

To create Web service references for the Sales application do the following:

1. Click **Navigator**, and select **Application Composer**.
2. Click **Web Services** on the Overview page.
3. Click the **Create Web Service Reference** icon on the Web Service page.
4. For connection type, select **SOAP**, and click **OK**.

   > **Note:** If a warning appears with the following message: "The preferred security options could not be determined. Contact your web service provider or consult your web service documentation to obtain the correct security details. If errors occur during service invocation, contact your system administrator with the web service and error details" click **OK** to continue.

5. Enter the name and WSDL for the CustomerManager web service reference, as listed in the following table, and then click **Read WSDL**.

Field	Value
Name	O_INT_JDE_CustomerManager
WSDL	https://<host>:<port>/<server>/CustomerManager?wsdl

6. Enter or confirm the fields listed in the following table, and then click **Save and Close**.

Field	Value
Service	CustomerManagerService
Port	CustomerManagerPort
Security Scheme	Invoke with separate user credentials over SSL
Credential Key	JDE_<username>_KEY
Disable time stamp verification	Select the check box.

7. Repeat the previous steps for the SalesOrderManager web service reference listed in the following table:

Field	Value
Name	O_INT_JDE_SalesOrderManager
WSDL	https://<host>:<port>/<server>/SalesOrderManager?wsdl
Service	SalesOrderManagerService
Port	SalesOrderManagerPort
Security Scheme	Invoke with separate user credentials over SSL
Credential Key	JDE_<username>_KEY
Disable time stamp verification	Select the check box

Create and Activate Sandboxes

To make changes to the application, you must first store the changes in an active sandbox. You can either create a sandbox or select an existing one, and make it active.

Watch video

You must activate the configuration tools you want to use in your sandbox. If you plan to use Page Composer in your sandbox and edit pages at a layer other than Site, you need to create a sandbox just for that layer, and activate only Page Composer in it.

Create and Activate Sandboxes

Follow these steps to create and activate sandboxes for most configuration tools. For flexfields, use the **Manage Descriptive Flexfields** task or the **Manage Extensible Flexfields** task instead.

1. Click **Navigator** > **Configuration** > **Sandboxes**.
2. On the Sandboxes page, click **Create Sandbox**.
3. Enter a name and description for your sandbox.
4. In the **Publishable** field, select **Yes** or **No**. If you set this option as **No**, you can just use your sandbox for testing purposes, but can never publish it.
5. In the All Tools section, select the tools you want to activate for this sandbox. The context layers for all selected tools are set as **Site** by default. So the changes you make using these tools affect all users.
6. If you select Page Composer, you can click the **Edit Sandbox Context** icon and change the context layer from **Site** to another layer, for example **Internal**. You can find the **Edit Sandbox Context** icon in the Support Context column. Make sure you select a context that's supported by the page you want to edit. Otherwise, you won't be able to edit the page.

   > **Note:** If you want to use other tools along with Page Composer in your sandbox, don't change the context layer for Page Composer, even though you can. That's because all tools except Page Composer support only a single context layer, Site. So if you change the context layer for Page Composer from Site to any other layer, all other tools that you might have selected earlier will be deselected.

7. Click **Create** to just create the sandbox, or **Create and Enter** to enter or activate the sandbox after creating it.

Here are a few things to know about activating tools in your sandbox.

- If you try to use a configuration tool in a sandbox without activating the tool in it, you get a message prompting you to activate the tool. You can add more tools to your sandbox later also.
- To create and manage saved searches and make UI adjustments (for example, change a table's column width) just for yourself, you must leave your sandbox before making these changes. But if you want to make these changes for others too, then make the changes with Page Composer open, in which case you also must be in a sandbox.

Activate Existing Sandboxes

Follow these steps to activate a sandbox.

1. Click **Navigator** > **Configuration** > **Sandboxes**.
2. From the list of sandboxes, if available, find the one you want to activate, and click the **Enter Sandbox** icon for that sandbox. Your sandbox is activated, and you can see its name on the sandbox bar before the global header. You can use the options available on the sandbox bar to quickly do some activities, such as view sandbox details, publish the sandbox, or leave the sandbox.

Publish Sandboxes

After you're done making changes to the application, publish the sandbox to make your changes available to all users. You must have the Administer Sandbox (FND_ADMINISTER_SANDBOX_PRIV) privilege to publish sandboxes.

> **Note:** Remember, you can't make further changes in the sandbox once you publish it.

Before you start, do these tasks:

- Test or validate your changes in the sandbox in preview mode before actually publishing it. If you made changes using Page Composer, don't forget to close it before testing. To preview your changes, on the sandbox bar before the global header, click **Sandbox Mode**, and select **Preview as if Published (Context: All)**.

  > **Note:** You can see the sandbox bar only when you're in an active sandbox.

- Resolve all conflicts flagged in the merge log of your sandbox.

To publish a sandbox:

1. Click **Navigator** > **Configuration** > **Sandboxes**.
2. On the Sandboxes page, click the name of the sandbox you want to publish.
3. Click **Publish**.

   > **Note:** You might not be able to publish your sandbox because of various reasons. For example, you haven't yet made any changes in your sandbox, or the **Control Publish Sandbox Action in Production Environment** profile option (FND_ALLOW_PUBLISH_SANDBOX) is set to **No**.

4. Click **Continue** to Publish. The sandbox is published to the mainline metadata.
5. Click **Done**.

Related Topics

- Create and Activate Sandboxes

Overview of Functions

There are two types of functions:

- **Global Functions**. Global functions work in all applications in Oracle CX Sales that have extensibility enabled. They are created in Application Composer.
- **Object Functions**. Object functions are attached to a specific object. They are created in the Object Functions tab of the Server Scripts section of the object definition in the Application Composer.

  Each function for the integration is defined in a Groovy script file. At the beginning of each file, a set of comments are listed in the following order:

- Function name

- Function return type
- Function parameters (one per line)

  Following the comments, the body of the function is listed.

Overview of Global Functions

The global functions for the integration are defined in the following Groovy script files:

- O_INT_GetLogMsg.groovy
- O_INT_GetSysParam.groovy
- O_INT_AddMultiValueCriteriaItem.groovy
- O_INT_ApplyFilter.groovy
- O_INT_Debug.groovy
- O_INT_Error.groovy
- O_INT_FindRowByKey.groovy
- O_INT_GetLogMsg.groovy
- O_INT_GetRecordCount.groovy
- O_INT_GetRecords.groovy
- O_INT_EBS_GetSysParam.groovy
- O_INT_Info.groovy
- O_INT_Log.groovy
- O_INT_LogMessage.groovy
- O_INT_Warn.groovy

The content of the Groovy files must be entered into Oracle CX Sales using the Application Composer. Before setting up global functions, create a sandbox and activate it. For more information, see the Create and Activate Unified Sandboxes topic.

Creating Global Functions: Explained

To create a global function, do the following:

1. Click **Navigator**, and select **Application Composer**.
2. In the Overview window, click **Global Functions**.
3. For each Groovy script file do the following:
   a. Select the row, then click the **Action** drop-down list, and select **Add**.

      The Create Global Function window appears.
   b. In the **Function Name** field, enter the function name from the first comment in the Groovy file.

c. From the **Returns** drop-down list, select the return type specified in the second comment in the Groovy file.
d. In the **Parameters** list, click the **Action** drop-down list, then click the **Add** icon.
e. Enter the name and type for the first parameter listed in the comments in the Groovy file.
f. Repeat the previous steps for the remaining parameters.
g. Copy the Groovy code from the file into the **Function Body** window.
h. Click **Validate** to validate the function.

> **Note:** When validating, certain functions trigger warnings. This is expected; you can click **OK** to continue. However, if an error is triggered, then contact Oracle Global Customer Support.

i. Click **Save and Close**.

Creating Object Functions: Explained

The object functions for the integration are defined in Groovy script files. These files are listed along with the specific objects that must be created or configured. Use the following general procedure to create object functions.

1. Click **Navigator**, and select **Application Composer**.
2. Expand the object for which you want to create a function, and then click **Server Scripts**.
3. In the Server Scripts window, click the **Object Functions** tab.
4. For each Groovy script file do the following:
   a. Click the **Action** drop-down list, and select **Add**.

      The Create Object Function window appears.
   b. In the **Function Name** field, enter the function name from the first comment in the Groovy file.
   c. From the **Returns** drop-down list, select the return type specified in the second comment in the Groovy file.
   d. In the **Parameters** list, click the **Add Parameter** icon.
   e. Enter the name and type for the first parameter listed in the comments in the Groovy file.
   f. Repeat the previous steps for the remaining parameters.
   g. Copy the Groovy code from the file into the **Function Body** window.
   h. Click **Validate** to validate the function.

      > **Note:** When validating, certain functions may trigger a warning. This is expected; you can click **OK** to continue. However, if an error is generated, then contact Oracle Global Customer Support.

   i. Click **Save and Close**.

Create Validation Rules

Validation rules are created in the Validation Rules tab of the Server Scripts section of the object definition in Application Composer.

Validation rules for the integration are defined in Groovy script files. At the beginning of each file, a set of comments is listed in the following order:

- Name
- Description
- Error message

Following the comments, the body of the rule is listed. The Groovy files are displayed as rules to be created for specific objects. Use the following general procedure to create validation rules for objects.

1. Click Navigator, and then select **Application Composer**.
2. Expand the object for which you want to create a validation rule, and then click **Server Scripts**.

   In the Server Scripts window, the Validation Rules tab is shown by default.
3. In the Object Rules area, click the **Add a new validation rule** icon.
4. In the Create Object Validation Rule window, in the **Name** field, enter the rule name and description from the comments in the Groovy file.
5. In the **Error Message** field, enter the error message from the comments in the Groovy file.
6. Copy the Groovy code from the file into the **Rule Definition** window.
7. Click **Save and Close**.

Create Custom Objects

Create a sandbox for each custom object.

After creating each custom object, validate the object and publish the sandbox before moving on to the next custom object. Only create a new sandbox after publishing the previous one. Never have two sandboxes active at the same time.

You create custom objects in the Application Composer.

Use the following procedure to create a custom object:

1. Click **Navigator**, and then select **Application Composer**.
2. In the Custom Objects list, click the **Create a New Object** icon.
3. In the Create Custom Object dialog box, enter the object information, and then click **OK**.

   The custom object appears in the Custom Objects list in the Objects menu.

Adding Fields to a Custom Object

You add fields using the Fields link nested within the custom object in the Application Composer. Use the following procedure to add fields to a custom object.

1. Expand the custom object's node in the Custom Objects list in the **Objects** menu.
2. Click **Fields**.
3. On the Fields page, click the **Create a custom field** icon.
4. In the **Select Field Type** dialog box, select the appropriate field type, and then click **OK**.
5. In the Create <Object Type> Field window, enter the parameters, and then click **Save and Close**.
6. Repeat the previous steps for other fields as necessary.

Creating Pages for a Custom Object

After you create a custom object and add fields to it, you create pages to expose the new object and its fields to users. Every top-level Oracle CX Sales object has an overview page, a creation page, and a details page. These pages make up a work area. You use an object's Pages area in Application Composer to create pages. Use the following procedure to create pages for a custom object.

1. Expand the custom object's node in the **Custom Objects** list in the **Objects** menu.
2. Click **Pages**.
3. On the landing page, click **Edit Summary Table** then do the following:
    a. In the **Configure Summary Table** area, add the fields that you want to display in the Summary Table to the Selected Fields list.
    b. In the **Configure Summary Table: Buttons and Actions** area, make sure Create is in the Selected Buttons list.
    c. Click **Save and Close**.
4. In the **Creation Page Layouts** area, click the **Default Custom Layout** link, and then do the following:
    a. Click the **Edit** icon.
    b. In the **Configure Detail Form** area, add the fields you want to display in the detailed summary.
    c. Click **Save and Close**.
    d. In the Creation Layout: Standard layout page, click **Done**.
5. In the **Details Page Layouts** area, select click the **Standard layout** link, then do the following.
    a. On the Details Layout: Standard layout, in the Summary area, click the **Edit** (pencil) icon.
    b. In the Configure Detail Form area, select the fields you want to display in the default summary.
    c. Click **Save and Close**.
    d. In the Details Layout: Standard layout page, click **Done**.

Create the Integration Configuration Custom Object

You create the Integration Configuration custom object in the Common application. Use a new sandbox to create the Integration Configuration object.

Enter the parameters listed in the following table for the Integration Configuration custom object, using the Creating Custom Objects procedure.

Parameter	Value
Display Label	Integration Configuration.
Plural Label	Integration Configuration.
Record Name Label	Integration Configuration ID.
Record Name Data Type	Select Automatically Generated Sequence from the drop-down list

Parameter	Value
Sequence Format	{000000000000000} (15 zeroes inside braces)
Object Name	O_INT_IntegrationConfig
Description	Leave blank.

Creating Fields for the Integration Configuration Custom Object

Create the fields listed in the following table for the Integration Configuration custom object using the Creating Custom Objects procedure.

Parameter	Field: Key	Field: RemoteSystemID	Field: Value
Field Type	Text	Text	Text
Display Label	Key	RemoteSystem	Value
Name	Key	RemoteSystemID	Value
Display Width	20	20	50
Display Type	Simple Text Box	Simple Text Box	Simple Text Box
Constraints	Required, Updatable, Searchable, Indexed	Required, Updatable, Searchable, Indexed	Required, Updatable, Searchable
Maximum Length	100	80	500

Creating Pages for the Integration Configuration Custom Object

Create the pages listed in the following tables for the Integration Configuration custom object using the Creating Custom Objects procedure.

The following table lists the required information to configure pages for the Integration Configuration custom object .

Page Layout	Layout Name	Item	Value
Landing Page Layouts	Default Custom Layout	Drill Down Column	Configuration ID
		Selected Fields	• Remote System • Configuration ID

Page Layout	Layout Name	Item	Value
			• Key • Value
		Buttons and Actions	Select both the Export and Create options.
Creation Page Layouts	Default Custom Layout	Selected Fields	• Integration Configuration ID • Remote System • Key • Value
Details Page Layouts	Default Custom Layout	Selected Fields	• Integration Configuration ID • LastUpdateBy • LastUpdateDate • Remote System • Key • Value

Creating a Validation Rule for the Integration Configuration Custom Object

Create the object validation rule from the UniqueKey.groovy file, using the Creating Validation Rules procedure.

Creating Global Functions for the Integration Configuration Custom Object

Create global functions from the following Groovy files, using the Creating Global Functions procedure.

- O_INT_GetIntegConfigParameter.groovy (returns a specific parameter for a category)
- O_INT_GetIntegConfigParameters.groovy (returns all parameters for a category)

> **Note:** These global functions depend on the Integration Configuration custom object being created, so they were not created previously in the Creating Global Functions procedure..

Validating and Publishing the Sandbox

Validate and publish the sandbox for the Integration Configuration custom object, using the Publish Unified Sandboxes topic.

Create the XREF Custom Object

You create the XREF custom object in the Common application. Use a new sandbox to create the XREF custom object.

Creating the Object for the XREF Custom Object

Enter the parameters listed in the following table for the XREF custom object, using the Creating Custom Objects procedure.

Parameter	Value
Display Label	XREF
Plural Label	XREF
Record Name Label	XREF ID
Record Name Data Type	Select Automatically Generated Sequence from the drop-down menu.
Sequence Format	{000000000000000} (15 zeroes inside braces)
Object Name	O_INT_XREF
Description	Leave blank.

Creating Fields for the XREF Custom Object

Create the fields listed in the following table for the XREF custom object using the Creating Custom Objects procedure.

Parameter	FusionObjectType Field	FusionRecordID Field	RemoteObjectType Field	RemoteRecordID	RemoteSystemID
Field Type	Text	Text	Text	Text	Text
Display Label	Fusion Object Type	Fusion Record ID	Remote Object Type	Remote Record ID	Remote System
Name	FusionObjectType	FusionRecordID	RemoteObjectType	RemoteRecordID	RemoteSystemID
Display Width	20	20	20	20	20
Display Type	Simple Text Box	Simple Text Box	Simple Text Box	Simple Text Box	Simple Text Box
Constraints	Required, Updatable, Searchable, Indexed	Required, Updatable, Searchable, Indexed	Required, Updatable, Searchable, Indexed	Required, Updatable, Searchable, Indexed	Required, Updatable, Searchable, Indexed
Maximum Length	100	50	100	50	80

Creating Global Functions for the XREF Custom Object

Create global functions from the following Groovy files, using the Creating Global Functions procedure.

- O_INT_CreateXREF.groovy
- O_INT_GetAllXREF.groovy
- O_INT_GetXREF.groovy
- O_INT_UpdateXREF.groovy

Note: These global functions depend on the XREF custom object being created, so they were not created previously in the Creating Global Functions procedure.

Validating and Publishing the Sandbox

Validate and publish the sandbox for the XREF custom object, using the Publish Unified Sandboxes topic.

Creating the JDE Match Child Object: Explained

Use the following procedure to create the JDE Match child object. Creating a child object is similar to Creating a Custom Object, but you also complete the Child Collection Name field to specify the internal name for the set of child object records.

To create the JDE Match child object do the following:

1. Click **Navigator**, and select **Application Composer**.
2. Expand the **Standard Objects** node, and then select **Account**.
3. Click the **Create Child Object** button.
4. In the **Create Custom Child Object** dialog box, enter the object information listed in the following table, and then click **OK**.

Parameter	Value
Display Label	JDE Match
Plural Label	JDE Match
Record Name Label	JDE Match ID
Record Name Data Type	Select Automatically Generated Sequence from the drop-down list.
Sequence Format	{000000000000000} (15 zeroes inside braces)

Parameter	Value
Object Name	O_INT_JDE_MATCH
Description	Leave blank.
Child Collection Name	O_INT_JDE_MatchCollection

Adding Fields to the Child Object

Create the fields for the JDE Match child object using the following procedure:

1. In the Application Composer, expand the **Account** node from the Standard Objects list.
2. Click **JDE Match** child object, and, then expand the JDE Match node in the Custom Objects list.
3. Click **Fields**.
4. On the Fields page, click the **Create a custom field** icon.
5. In the **Select Field Type** dialog box, select the appropriate field type, and then click **OK**.
6. In the Create Object_Type Field window, enter the parameters, and then click **OK**.
7. Repeat the previous steps for other fields as necessary.

Parameter	Address1 Field	City Field	State Field	Province Field	PostalCode Field	Country Field
Field Type	Text	Text	Text	Text	Text	Check box
Display Label	Address Line 1	City	State	Province	Postal Code	Selected
Name	Address1	City	State	Province	PostalCode	Selected
Display Width	20	20	20	20	8	Not applicable
Display Type	Simple Text Box	Simple Text Box	Simple Text Box	Simple Text Box	Simple Text Box	Not applicable
Constraints	Updatable, Searchable	Updatable, Searchable	Updatable, Searchable	Updatable, Searchable	Updatable, Searchable	Updatable, Searchable
Maximum Length	80	80	80	80	15	Not applicable

The following table lists more fields required to create the JDE Match child object.

Parameter	PartyName Field	PartyID Field	PartyType Field	IndustryClassification Field	PrimaryPhoneAreaCode Field
Field Type	Text	Text	Text	Text	Text

Parameter	PartyName Field	PartyID Field	PartyType Field	IndustryClassification Field	PrimaryPhoneAreaCode Field
Display Label	City	State	Country	Industry Classification	Primary Phone Area Code
Name	City	State	Country	IndustryClassification	PrimaryPhoneAreaCode
Display Width	20	20	20	10	20
Display Type	Simple Text Box	Simple Text Box	Simple Text Box	Simple Text Box	Simple Text Box
Constraints	Updatable, Searchable	Updatable, Searchable	Updatable, Searchable	Updatable, Searchable	Updatable, Searchable
Maximum Length	80	80	80	80	80

The following table lists more fields required to create the JDE Match child object.

Parameter	PrimaryPhoneNumber Field	PrimaryEmail Field
Field Type	Text	Text
Display Label	Primary Phone Number	Primary Email
Name	PrimaryPhoneNumber	PrimaryEmail
Display Width	20	20
Display Type	Simple Text Box	Simple Text Box
Constraints	Updatable, Searchable	Updatable, Searchable
Maximum Length	80	80

Adding the Constraint Expression for the Selected Check Box

After creating and saving the Selected check box, you edit it to add an expression to the Updatable constraint. To add an expression to the Updatable constraint for the Selected check box do the following:

1. After adding fields to the JDE Match Child object, click the link for the **Selected** check box on the **Fields** page.
2. In the **Edit Checkbox Field: Selected** window, click the **Expression Builder** icon.

   The Expression Builder appears, with the Functions tab displayed by default.
3. In the expression window, enter the following text:`Selected_c == null || Selected_c == 'N'`
4. Click **OK**.

5. In the **Edit Checkbox Field: Selected** window, click **Save and Close**.

Configuring Standard Objects: Explained

Adding fields to a standard object is very similar to Adding Fields to a Custom Object. To add a field to a standard object do the following:

1. Click **Navigator**, and then select **Application Composer**.
2. Select the object, and then click **Fields**.
3. On the Fields page, click the **Create a custom field** icon.
4. In the **Select Field Type** dialog box, select the appropriate field type, and then click **OK**.
5. In the Create Object_Type Field window, enter the parameters, and then click **OK**.
6. Repeat the previous steps for other fields as necessary.

Adding Actions and Links to an Object

Use the following procedure to add actions and links to an object.

1. Click **Navigator**, and then select **Application Composer**.
2. Select the object, and then click **Actions and Links**.
3. On the Object_Name: Actions and Links page, click the **Create** icon.
4. Enter the display label, name, and description.
5. Select the type, **Action** or **Link**.
6. Select the source:
    - **Script.** Select a method name, or click the **New** icon to enter a new object function.
    - **URL.** Define a URL expression.
7. Click **Save**.
8. Repeat the previous steps to add other actions or links as necessary.

Creating Triggers for an Object

Triggers are created in the Triggers tab of the Server Scripts section of the object definition in the Application Composer. Triggers for the integration are generally defined in Groovy script files. At the beginning of each file, a set of comments is listed in the following order:

- Type (object or field)
- Trigger
- Name
- Error message

Following the comments, the trigger definition is listed. The Groovy files are listed after the triggers to be created for specific objects.

Use the following general procedure to create triggers for objects.

1. Click **Navigator**, and then select **Application Composer**.
2. Expand the object for which you want to create a trigger, and then click **Server Scripts**.
3. In the Server Scripts window, click the **Triggers** tab.

4. Click the **Add a new Trigger** icon for Object Triggers or Field Triggers.
   The Create Object Trigger or Create Field Trigger window appears.
5. Select the trigger from the drop-down list, and then enter the name.
   The trigger and name come from the second and third comments in the Groovy file, respectively.
6. In the **Error Message** area, enter the error message, if any is provided.
7. Copy or enter the Groovy code from the file into the **Trigger Definition** window.
8. Click **Save and Close**.

Configuring the Account Standard Object: Explained

Make the following changes to the Account standard object.

Adding Fields

Add the fields listed in the following table to the Account standard object, using the procedure Adding Fields to a Standard Object.

Parameter	Field
Field	JDE Sync Status
Field Type	Text
Display Label	JDE Sync Status
Name	O_INT_JDE_Sync_Status
Display Width	Leave blank
Display Type	Simple Text Box
Constraints	Updateable (Constraint Expression: false)
Maximum Length	25
Default Value	Fixed Value = Not Synchronized

Creating Global Functions

Create functions from the following Groovy files, using the procedure Creating Global Functions:

- O_INT_JDE_GetBaseURL.groovy

- O_INT_JDE_GenerateSalesDocumentURL.groovy

Creating Object Functions

Create functions from the following Groovy files, using the procedure in Creating Object Functions:

- O_INT_JDE_CheckMatch.groovy
- O_INT_JDE_CleanupMatches.groovy
- O_INT_JDE_CreateCustomer.groovy
- O_INT_JDE_CustomerOrderURL.groovy
- O_INT_JDE_CustomerQuoteURL.groovy
- O_INT_JDE_CustomerReportURL.groovy
- O_INT_JDE_CustomerSalesDocumentURL.groovy
- O_INT_JDE_CustomerSync.groovy
- O_INT_JDE_GetSearchCriteria.groovy
- O_INT_JDE_OpenSalesAnalysisURL.groovy
- O_INT_JDE_PopulateMatches.groovy
- O_INT_JDE_SalesByOpportunityURL.groovy
- O_INT_JDE_SearchCustomer.groovy

Creating the Action

Add the Retry Sync action listed the following table to the Account object, using the procedure Adding Actions and Links to an Object.

Parameter	Value
Display Label	Retry Sync
Type	Action
Description	Leave blank
Source	Script
Method Name	O_INT_JDE_CustomerSync

Creating the Validation Rule

Create a validation rule from the O_INT_JDE_MatchSelectRule.groovy file, using the procedure Creating Validation Rules, and the information from the following table.

Parameter	Value
Rule Name	O_INT_JDE_MatchSelectRule
Error Message	Only one JDE customer record should be marked as selected after manual matching.
Rule Definition	O_INT_JDE_MatchSelectRule.groovy

Creating the Trigger

Create the following triggers using the procedure Creating Triggers for an Object.

- Trigger Type: "Before Update in Database". Groovy File: O_INT_JDE_CustomerSyncTrigger.groovy file.
- Trigger Type: "Before Insert in the Database". Groovy File: O_INT_JDE_CustomerCreateTrigger.groovy file.

Configure the JDE Match Child Object

You now configure the JDE Match child object.

Creating the Object Function

Create object functions from the O_INT_JDE_ForceCreateNewCustomer.groovy file using the procedure in Creating Object Functions.

Adding an Action to the Child Object

Using the information from the following table, add the Create New JDE Customer action to the JDE Match child object, using the procedure in Adding Actions and Links to an Object which appears in the Configuring Standard Objects topic.

Parameter	Value
Display Label	Creating New JDE Customer
Name	Force_Create_New_Customer
Type	Action
Description	Leave blank
Source	Script
Method Name	O_INT_JDE_ForceCreateNewCustomer

Creating Triggers

Create a trigger called "Before Update in Database" from the O_INT_JDE_CleanupMatchesTrigger.groovy file, using the procedure in Creating Triggers for an Object.

Validating and Publishing the Sandbox

Validate and publish the sandbox for the JDE Match object using the procedures in the Publish Unified Sandboxes topic.

Related Topics

- Create and Activate Sandboxes

Configure Account Object Pages

You modify pages for the Account standard object in Application Composer.

Use this procedure to add the subtabs listed the following tables.

1. Click **Navigator**, and select **Application Composer**.
2. Click **Pages**.
3. For **Details Page Layouts**, select **Standard Layout**, then click the **Duplicate Layout** icon.
4. In the Duplicate Layout dialog box, enter the following name for the duplicate layout: Integration Layout, and make sure that **Standard Layout** is listed as the **Source Layout**.
5. Click **Save and Edit**.
6. In the Details Layout: Integration Layout: Buttons and Actions page, click the **Edit** icon (pencil).
7. In the Details Layouts: Integration Layout page, in the Subtabs Region area, select **Default Layout**, then click the **Edit Layout** (pencil) icon.
    a. In the Subtabs Region, click the **Subtab Profile** icon.
    b. In the Summary area, click the Edit (pencil) icon to view the **Configure Detail** form.
    c. In the Configure Detail Form, move **JDE Sync Status** to the Selected Fields list.
    d. Click **Save and Close**.
8. In Available Actions, move **Retry Sync** from the Available Actions list to the Selected Actions list.
9. Click **Save and Close**.
10. In the Subtabs Region area, click the **Add** icon to add a new subtab.
11. In the Create Subtab page, select **Web Content** and click **Next**.
12. In the Create Subtab: Web Content page, enter the information listed in each of the following tables, click **Save and Close** after creating each subtab, and then repeat the previous steps.

    The following table lists the information required to create the JDE Quotes subtab.

Parameter	Value
Display Label	JDE Quotes
Source	URL

Parameter	Value
URL Definition	O_INT_JDE_CustomerQuoteURL()
Display Icon	Choose your preferred icon

The following table lists the information required to create the JDE Orders subtab.

Parameter	Value
Display Label	JDE Orders
Source	URL
URL Definition	O_INT_JDE_CustomerOrderURL()
Display Icon	Choose your preferred icon

The following table lists the information required to create the Open Sales Analysis subtab.

Parameter	Value
Display Label	Open Sales Analysis
Source	URL
URL Definition	O_INT_JDE_OpenSalesAnalysisURL()
Display Icon	Choose your preferred icon

The following table lists the information required to create the Sales By Opportunity subtab.

Parameter	Value
Display Label	Sales By Opportunity

Parameter	Value
Source	URL
URL Definition	O_INT_JDE_SalesByOpportunityURL()
Display Icon	Choose your preferred icon

13. In the Subtabs Region area of the Edit Details Page, click the click the **Add** icon to create a new child object.
14. In the Create Subtab page, select **Child object** and click **Next**.
15. In the Create Subtab: Child or Related Object page, enter the information listed in the following table:

    The following table lists the information required to create the subtab for the child object.

Parameter	Value
Data Object	JDE Match
Display Label	JDE Customer Matches
Display Icon	Choose your preferred icon

16. In the Configure Summary Table area, move **Selected**, **Party Name**, **Party ID**, **Address Line 1**, **City**, **State**, **Postal Code**, **Country**, and **JDE Match ID** to the Selected Fields list.
17. In the Configure Summary Table: Buttons and Actions area, make sure the **Show Edit** check box is selected, and deselect the **Show Create** and **Show Remove** check boxes, and move **Create New JDE Customer** to the Selected Buttons list, then click **Save and Close**.

Configure the Opportunity Standard Object

Create a new sandbox for configuring the Opportunity standard object, using the procedure in the Create and Activate Unified Sandboxes topic.

Adding Fields

Add the fields shown in the following table to the Opportunity standard object, using the procedure Adding Fields to a Standard Object from the Configuring Standard Objects topic.

Parameter	Sales Document Status Field
Field Type	Text

Parameter	Sales Document Status Field
Display Label	Sales Document Status
Display Type	Simple Text Box
Name	O_INT_SalesDocumentStatus
Constraints	Updateable (Constraint Expression: false)
Maximum length	150
Default Value	None

Creating Global Functions

Create functions from the following Groovy files, using the procedure in the Creating Global Functions topic.

- O_INT_EBS_CreateContact.groovy
- O_INT_EBS_IsMatchedContact.groovy
- O_INT_EBS_SearchContact.groovy

Creating Object Functions

Create functions from the following Groovy files, using the procedure in the Creating Object Functions topic.

- O_INT_EBS_CreateQuote.groovy
- O_INT_EBS_OpptyQuoteURL.groovy
- O_INT_EBS_SearchContactAndCreateXref.groovy
- O_INT_EBS_SyncPrimaryContact.groovy

Creating the Action

Add the Create Quote action shown in the following table to the Opportunity object, using the procedure Adding Actions and Links to an Object from the Configuring Standard Objects topic.

Parameter	Value
Display Label	Create Quote
Name	Create_Quote
Type	Action

Parameter	Value
Description	Leave blank.
Source	Script
Method Name	O_INT_EBS_CreateQuote

Creating Triggers

Create a trigger from the O_INT_CleanUpStatusTrigger.groovy file using the procedure Creating Triggers from the Configuring Standard Objects topic .

Configuring Pages

You modify pages for the Opportunity object in the Application Composer. To configure pages for the Opportunity standard object do the following:

1. Click **Navigator**, and select **Application Composer**.
2. Expand the **Opportunity** node from the Standard Objects list.
3. Click **Pages**.
4. In the Details Page Layouts area, select **Standard Layout**, and then click the **Duplicate Layout** icon.
5. In the **Duplicate Layout** dialog box, enter the following name for the duplicate layout: Integration Layout, and make sure that **Standard Layout** is listed as the **Source Layout**.
6. Click **Save and Edit**.
7. In the **Details Layout: Integration Layout** page, click the **Edit** icon (pencil).
8. On the **Details Layout: Integration Layout: Buttons and Actions** page, move **Create Quote** from the Available Buttons list to the Selected Buttons list, and then click **Save and Close**.
9. In the **Details Layout: Integration Layout** page, in the **Subtabs Region** area, click the **Edit** (pencil) icon.
10. Move **Sales Document Status** from the Available Fields list to the Selected Fields list, and then click **Save and Close**.
11. In the **Configure Detail Form: Buttons and Actions** area, move **Create_Quote** to the Selected Buttons area, and then click **Save and Close**.
12. In the **Subtabs Region** area, click the **Add** icon to add a new subtab.
13. In the **Create Subtab** page, select **Web Content** and click **Next**.
14. In the **Create Subtab: Web Content** page, enter the information in the following table, and when finished, click **Save and Close**.

Parameter	Value
Display Label	Quotes
Source	URL
URL Definition	O_INT_EBS_OpptyQuoteURL()
Display icon.	An icon is selected by default. Click **Change Icon** if needed.

Parameter	Value

Validating and Publishing the Sandbox

Validate and publish the sandbox for the Opportunity standard object, using the procedures in the Publish Unified Sandboxes topic.

Related Topics

- Create and Activate Sandboxes
- Publish Sandboxes

Connect Oracle CX Sales to JD Edwards EnterpriseOne

You use the Integration Configuration page to allow Oracle CX Sales to exchange data with JD Edwards EnterpriseOne.

This example uses JDE_9.1 as the name of the JD Edwards EnterpriseOne instance; yours might have a different name. To connect Oracle CX Sales to JD Edwards EnterpriseOne do the following:

1. Click **Navigator**, and select **Application Composer**.
2. On the **Integration Configuration** page, click the **Create** icon.
3. On the **Create Integration** page, enter the information for the first key-value pair in the following table, and then click **Save and Close**.
4. Repeat the previous steps for the remaining key-value pairs listed in the following table.

Remote System	Key	Value
JDE_9.1	host	Host name of your JD Edwards EnterpriseOne instance, for example: myJDE_server.example.com
JDE_9.1	port	HTTPS port of your JD Edwards EnterpriseOne instance, for example: 443
JDE_9.1	protocol	HTTPS
JDE_9.1	jdeQuoteVersion	Enter the version of the Sales Order Entry application (P4210) that is used to enter quotes. For example: ZJDE0003
JDE_9.1	jdeOrderVersion	Enter the version of the Sales Order Entry application (P4210) that is used to enter sales orders. For example: ZJDE0001
JDE_9.1	jdeCustomerVersion	Enter the version of Work with Customer Master application (P03013) that is used to

Remote System	Key	Value
		enter customer information. For example: ZJDE0001
JDE_9.1	enableAutoModeForSingleMatch	Enable this feature (Y/N)
JDE_9.1	oneViewReportIdOpenSales	To create this value, see the Creating One View Report IDs task.
JDE_9.1	oneViewReportIdSalesByOpty	To create this value, see the Creating One View Report IDs task.
JDE_9.1	enforceRevenueLineMapping	Enable this feature (Y/N)

Related Topics

- Creating One View Report IDs: Explained

Creating One View Report IDs: Explained

You must now retrieve One View Report (OVR) report IDs from the JD Edwards EnterpriseOne applications listed in the following task.

For additional information about the One View Reports used to view sales opportunity information, refer to the One View Reporting for Sales Order Management chapter in the JD Edwards EnterpriseOne Applications One View Reporting User Guide. You can access this guide on the One View tab of the JD Edwards EnterpriseOne Applications Documentation Library, located at: http://docs.oracle.com/cd/E16582_01/index.htm. To review the report

ID values for One View Report applications, perform the following task:

1. From the JD Edwards EnterpriseOne application, in the **Fast Path** field, enter **p42272**.
2. Click in the **Item Number** field, then click the **Search** icon, and in the **Item Search & Select** window, enter **210** in the **Item Number** field, and then click the **Find** icon.
3. Click **OneView**, then **Manage Reports**, and then select **Sales By Opportunity** from the drop-down list.
4. In the Layout view click the **Information** (i) button.
5. In the About window, make a note of the ID value in the **One View Report Information** field.

Accessing the Open Sales One View Report ID

To access the Open Sales One View Report ID do the following:

1. From the JD Edwards EnterpriseOne application, in the **Fast Path** field, enter **p42270**.
2. Click in the **Item Number** field, then click the **Search** icon, and in the **Item Search & Select** window, enter **210** in the **Item Number** field, and then click the **Find** icon.
3. Click **OneView** and select **Sales By Opportunity** from the drop-down list.
4. Click **OneView**, then **Manage Reports**, and then select **Open Sales Analysis** from the drop-down list.
5. In the **Layout** view click the **Information** (i) button.

6. In the **About** window, make a note of the ID value in the **One View Report Information** area.

Related Topics

5 Postconfiguration Tasks

Overview of Post-Configuration Tasks

For the integration to work correctly, you only perform post-configuration tasks after configuring both JD Edwards EntepriseOne and Oracle CX Sales.

Note: The Oracle CX Sales and JD Edwards EntepriseOne integration supports mapping items from JD Edwards EntepriseOne to either product groups or product items in Oracle CX Sales. Choose the mappings and steps from the following topic that best suit your implementation of Oracle CX Sales.

Import Products

Perform this task to populate new quotes with quote line items.

For the Oracle CX Sales to JD Edwards EnterpriseOne integration to populate new quotes with quote line items from the revenue lines on associated opportunities, you must create mappings to associated products from the front office Sales application to the back office Enterprise Resource system. Use the following steps to create the mappings.

Configuring Encoding in the JD Edwards EnterpriseOne Application

Prior to creating a catalog, you must configure character encoding in the JD Edwards Unicode Flat File Encoding Configuration program (P93081).

1. In the JD Edwards Unicode Flat File Encoding Configuration program, enter the values listed in the following table:

Field	Value
User Role	Public. For all users. This can be configured for a specific user ID if necessary.
Environment	The name of your environment.
Program ID	UBE name: R4101ZO.
Version	Enter the version name. The default value is ZJDE0001.
Encoding Name	UTF-8.

Field	Value
Status	Active.

Create a JD Edwards EnterpriseOne Catalog

Use this topic to create a JD Edwards EnterpriseOne catalog to enable you to import products.

1. Navigate to **Setup and Maintenance**.
2. Click the **Tasks** icon, and then click the **Search** link.
3. In the Search field, enter: **Manage Product Groups**.
4. In the Search Results pane click the **Manage Product Groups** link
5. In the **Manage Product Groups** view, search for **JDE Catalog** to ensure one is not already created.
6. Click the **Create** icon, and in the **Create Product Group** dialog box, enter the information listed in the following table:

Field	Value
Name	JDE Catalog
Display	JDE Catalog
Effective from Date	The current date
Active	Click the check box to enable
Allow Duplicate Children	Deselect the check box to disable
Root Catalog	Check the check box to enable

7. Click **Save and Close** to publish the new product group.

   You then add the JDE Catalog as a child of the root catalog.
8. In the **Manage Product Groups** search box, search for **Master Rollup**.
9. In the **Product Group** view, click the **Lock** button.
10. Click the **Subgroups** tab, and then click the **Select and Add** icon.
11. Search for **JDE Catalog**, then select it, and then click **OK**.
12. In the **Product Group** view, click **Publish**.
13. In the **Manage Product Groups** view, click **View**, **Columns**, then make sure **Reference Number** is checked.

14. Note the reference number for the JDE Catalog.

    You use the reference number value when preparing the JDE Item export file for import into Oracle CX Sales.

Export JD Edwards EnterpriseOne Items to Oracle CX Sales

Use this topic to export JD Edwards EnterpriseOne items to Oracle CX Sales.

For additional information about the Item Batch Outbound program (R4101ZO) used to create the item XML file, see the Item Master chapter in the JD Edwards EnterpriseOne Applications Business Interface Implementation Guide. You can access this guide on the Cross-Product tab of the JD Edwards EnterpriseOne Applications Documentation Library, located at: http://docs.oracle.com/cd/E16582_01/index.htm.

To import JD Edwards EnterpriseOne items do the following:

1. Export JD Edwards items using the batch process by submitting Item Batch Outbound program (R4101ZO).

   When you submit Item Batch Outbound program (R4101ZO) for exporting items into XML, the printer selection window appears.

   In the printer selection window, then click the **Document Setup** tab, and click the **OSA Interface Name** check box, and enter **XMLPOSA** in the text box beneath the check box.
2. Submit the batch and review the submitted jobs.

   After the job is completed, a status of **Done** is displayed.
3. Select a row, then from the **Row** menu, select **View OSA**.
4. Use the included processing script to prepare the XML file.

   > **Note:** Of the two included scripts, use the script appropriate to your operating system. Refer to the Related Topics section in this topic for a link to integration implementation files (Doc ID 1645923.1) on My Oracle Support.

   - Linux: prepareImportItems.sh
   - Windows: prepareImportItems.ps1
5. Right-click PowerShell and select **Run as administrator**.
6. Execute the command: Get-ExecutionPolicy.

   If the returned value is Restricted, you will receive an error. In this case, note the current value, and execute the command: Set-ExecutionPolicy RemoteSigned.
7. From the \batch_scripts folder, run the following script to retrieve the necessary XML file:
   - Windows:

     ```
     .\prepareImportItems.ps1 .\<jde items xml file> <OSC Jde Catalog Reference Number>
     ```
   - Linux:

     ```
     sh prepareImportItems.sh <jde items xml file> <OSC Jde Catalog Reference Number>
     ```
8. Re-institute the execution policy as: Set-ExecutionPolicy Restricted.

A new XML file is created: impost_<XML_filename>.

Related Topics

Perform Initial Import of JD Edwards EnterpriseOne Items from the XML Import File

You use the standard Oracle CX Sales Import Management process to import JD Edwards EnterpriseOne items into Oracle CX Sales product groups.

> **Note:** This import action will import all JD Edwards EnterpriseOne items to the XML import file. In certain circumstances however, it may only add only the first item as a subgroup to the parent JDE Catalog Product Group. If this occurs after you have performed the steps in this task, perform the tasks listed in Troubleshooting the Import of All Items into the JDE Catalog Product Group.

To perform initial import of JD Edwards EnterpriseOne items from an XML import file, do the following:

1. Navigate to **Setup and Maintenance**.
2. Click the **Tasks** icon, and then click the **Search** link.
3. In the Search field, enter: **Manage File Import Activities**.
4. Click the **Manage File Import Activities** link.
5. In the **Manage Import Activities** view, click the **Create** icon.
6. In the **Create Import Activity: Enter Import Options** view, enter the information from the following table:

Field	Value
Name	JDE Import
Object	Product Group
File Type	XML
File Selection	Specific file
Upload From	Click the Desktop button
File Name	Select the XML file generated by the processing script in Performing JD Edwards EnterpriseOne Item Export.

7. Click **Next**, then enter a new mapping using the details listed in the following table:

Column Header	Target Object	Target Attribute
AllowDupContentFlag	ProductGroupBulkImport	AllowDupContentFlag
Catalog	ProductGroupBulkImport	ProdGroupDescText
Description1_ID10	ProductGroupBulkImport	ProdGroupName
ItemID_ID1	ProductGroupBulkImport	PgRefNumber
ItemProduct_ID4	ProductGroupBulkImport	PgInternalName
Language	ProductGroupBulkImport	Language
SourceLangFlag	ProductGroupBulkImport	SourceLangFlag
SourceObjectType	ProductGroupBulkImport	SourceObjectType
ItemRel.RelActiveFlag	ProductGroupRelationBulkImport	ActiveFlag
ItemRel.ItemID_ID1	ProductGroupRelationBulkImport	PgRelChildRefNumber
ItemRel.RelLanguage	ProductGroupRelationBulkImport	Language
ItemRel.RelSourceLangFlag	ProductGroupRelationBulkImport	SourceLangFlag
ItemRel.RelationTypeCode	ProductGroupRelationBulkImport	RelationTypeCode

8. Click **Next** twice, and then click **Activate**.

   The process begins. To monitor progress, click **Refresh**.
9. Once the process shows a status of Completed, click **Done**.
   a. If there were errors, click the **Completed with Errors** link on the **Status** column and review the errors.
   b. Once you have reviewed and corrected the errors in the import file, run the import operation over again.
10. If necessary, perform the steps in the Troubleshooting the Import of All Items into the JDE Catalog Product Group task.

Related Topics

- Troubleshooting the Import of All Items into the JDE Catalog Product Group: Explained

Import Updates of JD Edwards EnterpriseOne Items from the XML Import File

Use this topic to import updates of your JD Edwards EnterpriseOne items.

1. Navigate to **Setup and Maintenance**.
2. Click the **Tasks** icon, and then click the **Search** link.
3. In the Search field, enter: **Manage File Import Activities**.
4. Click the **Manage File Import Activities** link.
5. In the **Manage Import Activities** view, locate and select the **JDE Import**.
6. Make a note of the Import Mapping name and time stamp of the import file.
7. Click the **Update** button, and then browse for the new import file, select it, and click **OK**.
8. Select the previously selected import mapping again.
9. Click **Next**.
10. Review the mapping to ensure that it matches the table from the Performing Initial Import of JD Edwards EnterpriseOne Items from the XML Import File task.
11. Click **Next**, twice, and then click **Activate**.

    The process begins. To monitor progress, click **Refresh**.
12. Once the process shows a status of Completed, click **Done**.
    a. If there were errors, click the Completed with Errors link on the Status column and review the errors.
    b. Once you have reviewed and corrected the errors in the import file, run the import operation over again.
13. If necessary, perform the steps in the Troubleshooting the Import of All Items into the JDE Catalog Product Group task.

Related Topics

- Perform Initial Import of JD Edwards EnterpriseOne Items from the XML Import File
- Troubleshooting the Import of All Items into the JDE Catalog Product Group: Explained

Troubleshooting the Import of All Items into the JDE Catalog Product Group: Explained

In certain circumstances, following the import process, only the first item from the XML item list appears in the JDE Catalog Product Group subgroup rather than the intended result of all items appearing.

If you see only one item in the subgroup list, use the following task to add all remaining items into the subgroup.

1. Navigate to **Setup and Maintenance**.

2. Click the **Tasks** icon, and then click the **Search** link.
3. In the Search field, enter: **Manage Product Groups**.
4. From the search results, click the **Manage Product Groups** link.
5. In the Manage Product Groups view, search for **JDE Catalog** and click the **Lock** button.
6. In the Product Group: JDE Catalog view, click the **Subgroups** tab, and then click the **Select and Add** button.

   You should see only one product group in the Subgroups list prior to clicking Select and Add.
7. In the Select and Add: Product Groups dialog box, click the **Advanced** button.
8. Click the **Add Fields** button and select **Description**.
9. In the Description field, enter **JDE_Catalog**, and click **Search**.

   The entire list of JD Edwards EnterpriseOne items appears.
10. Select one item in the list, then select all the records (CTRL + A).
11. The status bar displays **Rows Selected equal to All**.
12. Click **OK**.

    There may be a slight delay to allow the import of all the items.
13. Once the dialog box closes, verify that all items appear in the **Subgroups** list.
14. Click the **Publish** button to publish the changes.

6 Validating the Integration

Integration Validation Checklist

To verify that your integration is functioning properly, see the following checklist.

> **Note:** You must ensure that the domain values for attributes in Oracle CX Sales match the domain values for corresponding attributes in JD Edwards EnterpriseOne (or domain values are properly transformed from one system to the other in the integration). For example, if Oracle CX Sales uses the string **California** to represent the US state of California in an address and the JD Edwards EnterpriseOne system uses the two letter code **CA**, then errors may result in the integration. Be sure that the values, transformations, and any unique configurations you have previously applied to your CX Sales and JD Edwards EnterpriseOne systems have been accounted for when implementing this Oracle CX Sales and JD Edwards EnterpriseOne integration.

1. Successfully import JD Edwards EnterpriseOne items.

   For more information, see the Importing JD Edwards EnterpriseOne Items to Oracle CX Sales Catalog topic.
2. Choose one customer record and perform a synchronization operation.

   For more information, see the Customer Data Matching with JD Edwards EnterpriseOne topic, and perform all scenarios listed in the topic.
3. Using the same customer record from Step 2 create an Oracle CX Sales opportunity using imported JD Edwards EnterpriseOne items.
4. Using the opportunity you created in Step 3, create a new quote and a new order.

   For more information, see the Sales Order and Sales Quote Creation in JD Edwards EnterpriseOne topic.
5. Review the newly created quote and order records in the JD Edwards EnterpriseOne UI mashup.
6. Navigate to the customer record, locate the Quotes and Orders JD Edwards UI mashup and verify that the created quotes and orders from the previous steps also appear in the UI.
7. Using the quote and order records created in Step 4, review the JD Edwards EntepriseOne One View Reporting (OVR) reports.

   For more information, see Viewing Reports in JD Edwards EnterpriseOne.

Related Topics

- Import Updates of JD Edwards EnterpriseOne Items from the XML Import File
- How Customer Data is Matched in JD Edwards EnterpriseOne
- How Sales Orders and Sales Quote are Created in JD Edwards EnterpriseOne
- How to View Reports in JD Edwards EnterpriseOne

Configure the Integration with the Security Console

You can use the Security Console to perform unique configurations to your integration, such as restricting access to specific users, particularly to administration roles with the Customer Matching user interface, and so on.

Perform the following tasks in Oracle CX Sales after you have successfully imported JD Edwards EnterpriseOne items.

1. Click **Navigator**, and select **Security Console**.
2. Click the **Create Role** button, and add the information contained in the following table.

> **Note:** When assigning a role, consider assigning a role to a specific matching user. This enables individual users who have been assigned that role to perform customer matching. Assigning the role to an existing role hierarchy enables the group of users contained within the role hierarchy to perform customer matching.

The following table lists the information required to create the Customer Matching Admin role.

Field	Value
Role Name	Customer Matching Admin
Role Code	INT_Customer_Matching_Admin_Role
Role Category	CRM - Job Roles
Description	Custom role for granting the access required to perform customer matching.

After you have created the Customer Matching Admin role using Security Console, the role is exposed in Application Composer. You can then assign privileges which enable the role to view custom objects.

Assigning Privileges to the Role

Use this procedure to assign privileges and define security policies to the new role.

1. Click **Navigator**, and select **Application Composer**.
2. From the **Application** drop-down list, select the relevant application.
3. Expand the **Custom Objects** node, and then expand the relevant object.
4. Click **Security**, and then in the Define Policies: Object_name page, select the appropriate check boxes for the privileges you want the role to have for the object, and then click **Save and Close**.

7 Required Files

Required Files for the Integration

The following Groovy script files are required for the integration.

Global Functions

- O_INT_ApplyFilter.groovy
- O_INT_AddMultiValueCriteriaItem.groovy
- O_INT_Debug.groovy
- O_INT_Error.groovy
- O_INT_FindRowByKey.groovy
- O_INT_GetLogMsg.groovy
- O_INT_GetRecordCount.groovy
- O_INT_GetRecords.groovy
- O_INT_GetSysParam.groovy
- O_INT_Info.groovy
- O_INT_JDE_ConstructErrorPageURL
- O_INT_JDE_GetLogMsg.groovy
- O_INT_JDE_LogWithException
- O_INT_JDE_GetSysParam.groovy
- O_INT_Log.groovy
- O_INT_LogMessage.groovy
- O_INT_Warn.groovy

Batch Scripts

- prepareImportItems.ps1
- prepareImportItems.sh

Integration Configuration Custom Object

Validation Rule

- UniqueKey.groovy

Global Functions

- O_INT_GetIntegConfigParameter.groovy
- O_INT_GetIntegConfigParameters.groovy

XREF Custom Object

Global Functions

- O_INT_CreateXREF.groovy
- O_INT_GetAllXREF.groovy
- O_INT_GetXREF.groovy
- O_INT_UpdateXREF.groovy

JDE Match Child Object

Object Function

- O_INT_JDE_ForceCreateNewCustomer.groovy

Trigger

- O_INT_JDE_CleanupMatchesTrigger.groovy

Account Object

Global Functions

- O_INT_JDE_GetBaseURL.groovy
- O_INT_JDE_GenerateSalesDocumentURL.groovy

Object Functions

- O_INT_JDE_GetSearchCriteria.groovy
- O_INT_JDE_CheckMatch.groovy
- O_INT_JDE_CleanupMatches.groovy
- O_INT_JDE_CreateCustomer.groovy
- O_INT_JDE_CustomerOrderURL.groovy
- O_INT_JDE_CustomerQuoteURL.groovy
- O_INT_JDE_CustomerReportURL.groovy
- O_INT_JDE_CustomerSalesDocumentURL.groovy
- O_INT_JDE_CustomerSync.groovy
- O_INT_JDE_OpenSalesAnalysisURL.groovy
- O_INT_JDE_PopulateMatches.groovy
- O_INT_JDE_SearchCustomer.groovy

- O_INT_JDE_SalesByOpportunityURL.groovy

Trigger

- O_INT_JDE_CustomerSyncTrigger.groovy

Validations

- O_INT_JDE_MatchSelectRule.groovy

Opportunity Standard Object

Object Functions

- O_INT_JDE_CreateOrder
- O_INT_JDE_CreateQuote
- O_INT_JDE_CreateSalesDocument
- O_INT_JDE_OpptyOrderURL.groovy
- O_INT_JDE_OpptyQuoteURL.groovy
- O_INT_JDE_OpptySalesDocumentURL.groovy